Botnet-Based Attacks and Defence Mechanisms

Dilara Acarali and Muttukrishnan Rajarajan

Abstract This chapter considers the threat posed by botnets and the impact of botnet-based attacks on both private domains and the global digital infrastructure. Botnets are widely employed by cyber-criminals for a variety of malicious activities and are frequently observed as a component within large-scale organised cyber-crime campaigns. In addition to this, botnets are a varied and evolving threat, bound to grow in parallel with our increasing dependence on digital services and the Internet, as well as the adoption of upcoming technologies like the Internet-of-Things. Botnets can be considered as attacks in-and-of themselves, as well as platforms for future attacks. With this as the foundational perspective, this study examines how a botnet is defined and classified, how it is built and used, the characteristics of a botnet attack, and the factors contributing towards its success. We then analyse how a botnet provides other attack capabilities for the cyber-criminal. This is supplemented with a discussion of how the threat is adapting to new technologies, followed by a short survey of some outstanding problems to be considered in future research.

1 Introduction

Botnets are a growing problem in our increasingly digital world, underpinning malicious activities such as identify theft, financial fraud, corporate espionage, and even cyber-terrorism. They come in a variety of shapes and flavours, can be deployed on a multitude of platforms, and can vary in functionality and scope, making it difficult to find a single, standardised solution. To provide a general framework for discussion, botnet activities can be considered in two distinct parts

D. Acarali (✉) · M. Rajarajan
School of Mathematics, Computer Science and Engineering, City, University of London, London, UK
e-mail: dilara.acarali@city.ac.uk; r.muttukrishnan@city.ac.uk

M. Conti et al. (eds.), *Versatile Cybersecurity*, Advances in Information Security 72, https://doi.org/10.1007/978-3-319-97643-3_6

consisting of the initial penetration and infection of a target, followed by the exploitation of that target. To build a foundation for the rest of the study, this section explains how we define botnets and what kind of behaviours we can expect to observe from them.

1.1 Definition of a Botnet and Basic Functionality

At the most basic level, a botnet is defined as a collection of bot clients connected to each other via a control network through which they can send and receive updates and information. The word "bot" (shortened from "robot") refers to a piece of software which is written to carry out some automated tasks on behalf of the user. Legitimate services frequently use bots as part of their routine processes. For example, consumption bots on Twitter curate content for individual users based on their tastes [1]. The overall purpose of a bot is to free the human user from tasks which need to be performed constantly and consistently over time. Malicious bots function on the same principles. The cyber-criminal behind a botnet campaign is commonly referred to as a botmaster or botherder.

Although there are variations from case to case, the key components of a botnet can be summarised as follows: the bot client program, the infection vector, the communication protocol, the control architecture, the control servers, and the botmaster. The infection vector is the means by which the bot client is inserted into a victim system (e.g. worms, viruses or Trojan software). This tends to be defined as part of the malware design. Opposite the client, the command and control servers (referred to as C&C or C2 servers) are distribution and collection points for instructions and information. The communication protocol and control architecture in combination define how messages will be sent and received. Finally, the botmaster is the human brain and the driver behind the botnet, and their intentions or aims determine how it will be used.

Similarly, we can abstract the activities of a botnet into a generalised lifecycle (Fig. 1), which may begin with initial conception and design [2] or at the point of the first successful infection [3]. For practical purposes, we will focus on the second scenario here. Once executed, the malware performs its initial setup, and then enters the rally stage. This is where a previously isolated bot attempts to connect to the C&C network. Once this connection is achieved, the cycle moves into the secondary injection stage, in which relevant binaries and tools are gathered and installed on the host [3]. The bot is now ready for use and will be provided with commands via the C&C network. When the relevant command comes through, the cycle moves into the attack phase [3]. Note that when the attack ends, the botnet will probably fall back into waiting mode, ready for the next round of activity.

A botnet's main functionality comes from its ability to exploit the resources (e.g. memory, RAM, bandwidth, or data) of existing domains and devices to carry out large-scale tasks. Denial-of-Service (DoS) attacks and spam campaigns are both examples. To achieve this, the bot malware needs to propagate to as many

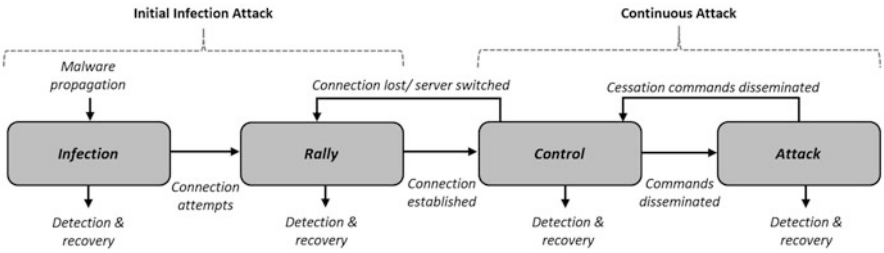

Fig. 1 Botnet lifecycle based on stages proposed by [3] with alignment to the 2-part attack

networked devices as possible, thereby increasing this pool of exploitable resources. Additionally, a bot provides a back-door into the victim system, thereby allowing a botmaster to gain access to potentially sensitive data and environments. It is the flexibility and range of botnets that make them an attractive option for cyber-criminals.

These functionalities are also what distinguish botnet attacks from others. The attack is not a one-off situation, but rather a continuous one. As long as a piece of bot malware remains within a network (and its C&C servers are online), that network should be considered under attack. The difficulty in dealing with this scenario is that there may be no visible symptoms whilst bots operate intermittently or stay dormant until their next update. In propagation modelling, the analogy applied to malware is that of an infectious disease. Bot malware can be thought of as a specialised strain that weakens a body's immune system, making it more vulnerable to future health problems or other illnesses. The network's functionality and integrity is compromised by the presence of this infection.

1.2 Botnet Classifications

Botnets can be classified in terms of their command dissemination style, broadly defined as either push- or pull-based (Fig. 2a). Commands originate from the botmaster, who plants them into the C&C network. C&C servers may then "push" messages out for the clients to receive. This was the typical approach of earlier botnets [4], but has also been used in more recent theoretical mobile setups [5]. In contrast, pull-based spread involves each client individually polling the C&C server to receive an update. Perimeter defences like firewalls contain default rules to block all unknown incoming web traffic. The pull-based approach is effective in getting around this issue, thanks to the client (who is internal to the network perimeter) initiating the connection. Examples of botnets which use pull-based methods are Storm [6], Zeus [7], and Asprox [8]. Meanwhile, GameOver Zeus is reported to use a hybrid push/pull approach for contingency [9].

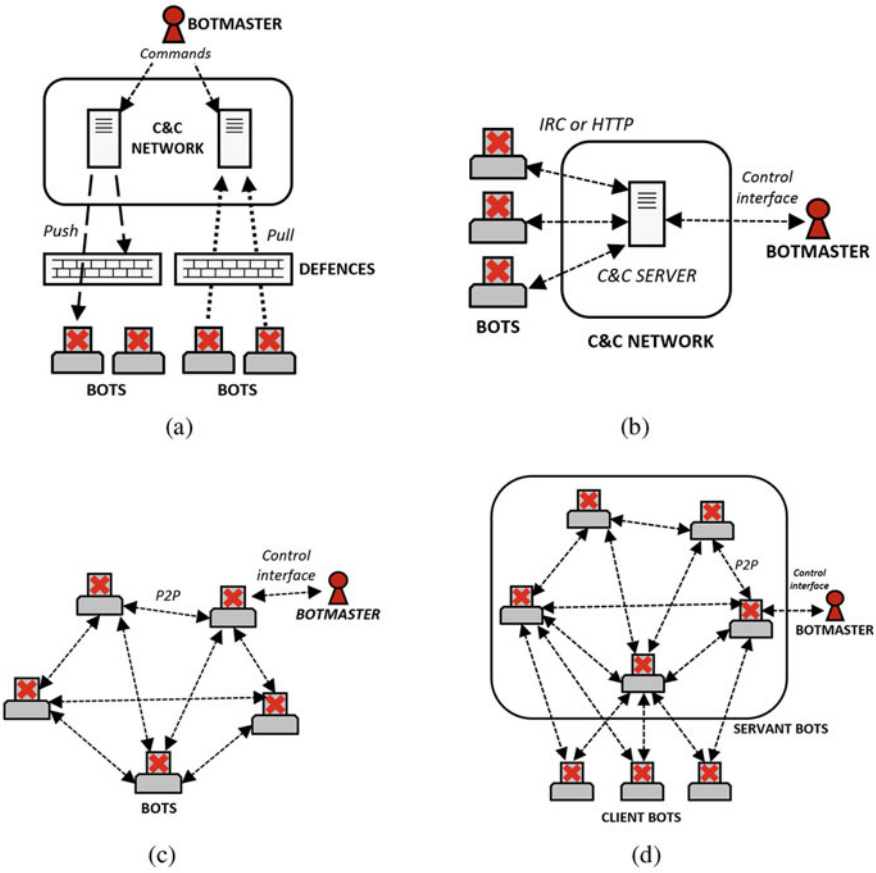

Fig. 2 Botnet classifications, including command dissemination style and architectures. (**a**) Push vs. pull-based dissemination (**b**) Centralised botnet (IRC- or HTTP-based) (**c**) Decentralised botnet (P2P-based) (**d**) Hybrid botnet (combined P2P- and HTTP-based) [10]

Botnets may also be classified by architecture, which refers to the way their client-server relationships are organised. Early botnets used solely centralised architectures (Fig. 2b), where many clients connect to a few central servers. This approach is simple and straightforward, but also vulnerable thanks to the single point-of-failure in its design [11]. Silva et al. [3] lists early examples of centralised botnets including EggDrop (first detected in 1993), GTBot (1998), SDBot and Agobot (2002), and Spybot (2003). The alternative is a decentralised architecture (Fig. 2c), where many clients connect to many servers, and the same devices may act as both clients and servers depending on the situational requirement [11]. This clearly eliminates the single point-of-failure, making the botnet more robust, discrete and resistant to takedown attempts [11].

Hybrid architectures have emerged more recently (Fig. 2d), featuring both centralised and decentralised elements to make the botnet easy to maintain and sturdy against adversaries [11]. This can also be used to obfuscate C&C networks behind layers of clients, making it harder to identify or trace them. For example, [14] reported that Waledac uses a decentralised hierarchical topology made up of spammer nodes, repeater nodes, and 3 layers of servers, combining centralised top-level C&Cs with a decentralised client node layout.

The C&C categories described so far tend to align with the use of particular protocols, providing another angle for classification. Centralised architectures use IRC or HTTP. For instance, EggDrop, AgoBot, and the Chuck Norris botnet are all IRC-based [3, 17]. However, this protocol has fallen out of favour after being targeted by network defenders and was replaced with HTTP, widely used in modern botnets including Rustock [12], Clickbot.A [13], Waledac [14], Koobface [16], Asprox [8], and Dirt Jumper [18]. HTTP-based C&C communication is effective in enabling bot flows to blend in with benign web traffic, and is often combined with pull-based dissemination to overcome defences. Meanwhile, P2P protocols are obvious choices for decentralised architectures, allowing bots to act as peers to exchange data symmetrically within overlay networks. Examples of P2P-based botnets are Storm and Nugache [6], Sinit, Phatbot, SpamThru, and Peacomm [22]. Custom protocols may also be used, like Mariposa's UDP-based Iserdo Transport Protocol [15].

The final classification method is based on malware family, which can be defined in two ways. In the first, botnets belonging to the same family are built using separate instances of the same malware [23]. For example, two botmasters may have access to the same binaries, and launch their own separate campaigns [23]. This is relevant where malware code is known to be highly accessible via underground forums or the black market [7]. The second definition is a group of separate malware codes, which appear to have a single progenitor. Zeus is a famous example of this. After first appearing in 2007, the malware was made public in 2011 [24] leading to the appearance of many derivatives, including GameOver Zeus, Citadel, and Shylock [24]. This is significant to our understanding of botnet evolution.

To summarise, Table 1 lists some botnets along with their classifications.

1.3 The Two-Part Attack

The botnet lifecycle establishes separate stages for spreading, rallying, awaiting commands, and performing attacks [3]. In this study, we further abstract these stages into two parts; the initial compromise and the continuous threat (Fig. 1). The first refers to the process of infiltration and infection, whilst the second refers to the cycle of attacks and waiting periods. From a defensive perspective, our intention is to frame the botnet with two opportunities for mitigation, where missing the opportunity to stop the infection opens the pathway to many future attacks.

Table 1 Summary of classification methods for a selection of botnets

	Classification method		
Botnet	Push/Push	Architecture	Protocol
Storm [6]	Pull	Decentralised	P2P
Nugache [6]	Pull/Push	Mixed	P2P/IRC
Rustock [12]	Pull	Centralised	HTTP
Clickbot.A [13]	Pull	Centralised	HTTP
Waledac [14]	Pull	Decentralised	P2P
Mariposa [15]	Pull	Centralised	UDP
Koobface [16]	Pull	Centralised	HTTP
Chuck Norris [17]	Push	Centralised	IRC
Asprox [8]	Pull	Centralised	HTTP
Zeus [7]	Pull	Centralised	HTTP
Dirt Jumper [18]	Pull	Centralised	HTTP
GameOver Zeus [9]	Pull/Push	Decentralised	P2P
Kelihos [19]	Pull	Decentralised	P2P
Citadel [20]	Pull	Centralised	HTTP
ZeroAcess [21]	Pull	Decentralised	P2P

The initial compromise attack is designed to discreetly implant malware into a node of the target network. Bots take great care to stay hidden at this stage, as the ultimate success of the botnet depends on it. Examples of obfuscation methods used are packing (Storm [6], Citadel [20]), encryption (Citadel [20], Mariposa [15]), rootkits (Nugache [6]), and registry amendments (Nugache [6], Dirt Jumper [18], Citadel [20]). These may also double as methods to weaken the host, making it more malleable. Successful detection and mitigation of this propagation process reduces the later impact of a botnet presence in the network.

A person with an infectious disease can become more vulnerable to further health issues, becoming a health-hazard to others in the process. The concept of the continuous threat is similar to this, as a compromised network (with even a single bot) is now in a weakened state with an increased chance of receiving new infections from unrelated malware and of propagating binaries on to others. The continuous threat also encompasses the possibility of the compromised network being used as a weapon in attacks against third-parties.

There are several benefits to considering botnet threats in this manner. Firstly, the distinction allows us to approach the two attacks separately and in a way befitting of their characteristics. We cannot approach single-host infections the same way we approach a DDoS attack, for example. However, both are botnet-related issues that need to be addressed as such. Secondly, it provides two opportunities for dealing with the problem. If the original infection can be resolved, no further action is needed. If not, there is still an opportunity for damage limitation, provided that the correct understanding and defensive strategies are in place.

2 Domain Impact

Despite the level of compromise, if the malware is well-designed, network users and analysts may be none the wiser. Bot presence may only become apparent once the botnet is offensively engaged. This section explores some of the ways in which an infected network may be directly impacted by the its unwelcome guests.

Maintaining stealth is crucial to a successful bot infection. Hence, the executed bot binary makes various changes to the host system. For example, Nugache installs a rootkit driver to System32 of Windows operating systems, which is then loaded into the kernel [6]. Both Nugache and Dirt Jumper make modifications to the system registry [6, 18]. These changes have an overall weakening effect on the host (and consequently, the wider network), perhaps re-opening previously dealt-with vulnerabilities, causing other malicious processes to go unnoticed, or introducing new exploits.

During the rally stage, bots register with the C&C and authenticate themselves, usually providing information on the host. This allows the botmaster to gain an overview of the variation of nodes recruited. These messages may contain details of hardware, operating system version, and patch levels – all potentially valuable in planning further attacks [8]. Additionally, many bots including Shylock, SpyEye, and Tinba contain keylogging and screen-capture modules used to steal user information [25]. Botmasters may sell this and network details on to other adversaries for future exploitation. Furthermore, captured user credentials enable a botmaster to personally manoeuvre into secure environments in the network.

Another impact of bot infection is the draining of a victim's resources, including physical memory, RAM, processing power, and bandwidth. Botnets are designed to use the collective power of many machines to reach some end goal. An example is the launch of DDoS attacks. Whilst the target may be a third-party, the continuous stream of messages sent out by bots can negatively affect the source network too, especially if multiple infections are present. In mobile botnets, C&C traffic may cause devices to hit data usage limits resulting in additional fees [26]. Alternatively, users of infected nodes may notice their devices running more slowly as bot processes operate in the background.

Discussion so far has focused on instances of a single botnet malware, but the presence of multiple unique malware infections is also possible. This could result in competitive behaviour for the finite resources available, and to the doubling of traffic generated. If analysts then become suspicious, this is a hinderance to both botnets. An example of possible competitive behaviour is the disabling or removal of one malware by another when the two exist on the same host [27]. Alternatively, botnets may collaborate with each other, providing access to core sets of nodes, as reported by [23], who observed separate botnets belonging to the same families with large areas of population overlap.

Case studies reveal that bots often spread laterally through the network resulting in localised clusters of infected hosts. This propagation may be driven by automated worms which scan the local network or via viruses and Trojans delivered through

Table 2 Methods used by botnets as part of initial and continuous attacks

Techniques	Used by
Initial compromise stage	
Binary obfuscation	Chuck Norris [17], Citadel [20], SpyEye [25]
Client testing	Citadel [20], Mariposa [15], Miner [28]
File system manipulation	Zeus [7], DirtJumper [18], Citadel [20]
Registry manipulation	Nugache [6], Rustock [12], Waledac [14]
Hardcoded C&Cs	Asprox [8], DirtJumper [18], Miner [28]
Fast-fluxing	Mirai [29], Waledac [14], Asprox [8]
DGA/domain-fluxing	Carberp [25], GameOver Zeus [25], Kraken [30]
Continuous threat stage	
Trojans	Storm [6], Nugache [6], Koobface [16]
Social engineering	Asprox [8], Chuck Norris [17], Miner [28]
Drive-by-downloads	JiFake [31], Plankton [31], Asprox [8]
DDoS	DirtJumper [18], Mirai [29], Kelihos [19]
Data theft	Zeus [25], SpyEye [25], Citadel [25]
Web-injects/fake pages	Tinba [25], Bugat [25], Asprox [8]
Click-fraud	Clickbot.A [32], ZeroAccess [21]

messaging channels (e.g. internal email) as spam. Mariposa reportedly checks for P2P applications and if found, copies itself to the shared folder to propagate [15]. When a bot infection is detected, it should always be assumed that there are others in the vicinity Table 2 summarises behaviours observed in various botnets.

In summary, a botnet infection can weaken a domain's defence against other threats, expose the network structure, and diminish productivity by causing slow connections and interrupted services. As even minor infections can quickly grow, botnet mitigation should be a top priority for all organisations, large and small.

3 Launching the Attack

The way in which an attack is initiated provides details as to its design, function, and purpose. Here we discuss the launch stages of both the initial compromise and continuous threat, demonstrating both the short-term and long-term effect of a botnet on the target domain.

3.1 The Initial Threat: Compromise Attack

During the infection stage, botnets expand through the propagation activities of individual bots. The characteristics of this process are largely defined by the chosen propagation vector, defined as the bot binary delivery method. Choice of

vector depends on the connectivity capabilities of targets – the more channels, the more possibilities. Malware may be delivered through LANs, the Internet, WiFi, Bluetooth, and even USB devices. Bot binaries piggyback on infectious malware, the choice of which has an impact on the rate and speed of spread. For example, botnets using worm-based propagation may expand faster than those using Trojans. This is because worms are self-activated and self-replicating (e.g. Chuck Norris botnet [17]), whereas Trojans require user interaction to execute and share them (e.g. GameOver Zeus [9]).

Once the bot binary is executed, it goes through a setup process. If the binary is highly obfuscated, it first needs to be unpacked and decrypted (e.g. Mariposa [15]). Next, it will create new directories in system folders and make copies of itself or important files. This allows the binary to launch from within the system folders, giving it the illusion of being part of the OS [6, 7]. The Zeus bot also copies the MAC details of nearby system files to further blend in [7]. The registry is edited to manipulate system processes (e.g. Dirt Jumper [18] and Citadel [20]), and some bots like Nugache install rootkits in the operating system kernel to help them stay hidden [6]. The Citadel bot also goes to the additional step of removing its binary files after installation [20]. These details can vary between families and implementations, making detection difficult without a priori knowledge.

After installation, the bot must rally to the C&C network to register and authenticate itself. This often includes the use of a unique ID (e.g. iKee.B [32] and Clickbot.A [13]) as well as details on the host system. For instance, Mirai bots send the IP, port, and login credentials of the victims [29], whilst Miner bots report on connectivity and speed testing results [28]. These details give the botmaster an overview of node capabilities and statuses, which may then be used to devise attacks or adapt propagation strategies. Bot binaries may be hardcoded with lists of C&C domain names or IPs [18] that are queried sequentially until one can be resolved. However, these C&Cs are easily detected by defenders through binary analysis. Botnets may employ expendable proxies (allowing bots to register and pull-down instructions to access the main C&Cs) to get around this. The rally process typically generates a large amount of erroneous DNS traffic, especially if servers have been migrated or taken offline. This characteristic can be used for detection [30].

Fluxing may be used to minimise detection risk or to protect the C&C network from exposure. Fast-fluxing is where DNS records are periodically updated to rapidly change the IP that a domain name resolves to, thereby obfuscating the true location of C&C servers. Waledac uses this approach, employing its upper tier nodes to act as DNS servers [14]. Double-fluxing adds the cycling of authoritative name server IPs, as observed in Asprox, which uses layered double-fluxing to create a "hydra-flux" setup [8]. Alternatively, botmasters may replace hardcoded lists with a domain name generation algorithm (DGA). Bots use DGAs to generate sets of possible domains, and test each one until a resolving IP is found. This is known as domain-fluxing. DGAs are pseudo-random, taking a seed value shared by the botmaster to ensure that matching domains can be achieved [30]. This behaviour was observed in Zeus, Conficker, and Kraken [9, 30].

The botmaster now has access to system resources, visibility of some users, and the ability to move stealthily through the network. The stage is set for follow-up attacks.

3.2 The Continuous Threat: Follow-Up Attacks

The continuous threat can be considered as either (a). the activities of a previously-initialised botnet (at macro scale), or (b). the activities of a single activated bot (at micro scale). Whichever is chosen, a variety of attack options are available to the botmaster at this stage.

One option is a Denial-of-Service (DoS) attack, where a large number of connection requests are sent to a target server with the intention of overwhelming its capacity. Distributed-DoS (DDoS) consists of requests generated by many devices, thereby increasing the total force of the attack [33]. Dirt Jumper reportedly has 4 different types of DDoS capability, including HTTP flood made up of standard HTTP requests, synchronous flood consisting of batches of 150 requests, downloads flood designed to consume bandwidth, and POST flood aimed at overloading the processing capacity of servers [18]. Large-scale DDoS attacks were also reported for Mirai in 2016 [33]. Alternatively, botnets may deliver ransomware, which once installed 'locks' the device, blocking user access. The authors then demand a ransom to unlock the system [34]. These types of attacks are designed to cause disruption to services.

User data is a precious commodity, and can be farmed via client bots. A keylogger records the keystrokes of a user, whilst screen capture tools record an image of the desktop. Harvested data is then sent back to the C&C. Examples of this behaviour have been observed in Carberp, Tinba, and Bugat [25]. Another method of data theft is a man-in-the-browser attack, where a web browser is hijacked with script injections allowing form details submitted by the user to be intercepted by the malware [25]. Bots may also manipulate browsers to redirect users to malicious replicas of legitimate web pages for social engineering.

Click-fraud is the manipulation of the online ad system to generate revenue for botmasters. Search engines maintain ad networks, agreeing with advertisers to host their content and serving it to users when certain keywords are triggered in search queries. Clicking an ad takes users to a landing page for which search engines receive payments-per-click. Third-parties (known as syndicates) are also allowed to display ads, receiving a cut of the earnings for the traffic they generate. Syndicates may then employ sub-syndicates, and so on, to extend ad visibility. Click-fraud exploits this system by generating fake traffic to the landing page. Clickbot.A sets up its own search engines (called doorways) which are registered as syndicates [13]. Bots are provided with a list of keywords to query, receiving ad URLs to be clicked [13]. ZeroAccess has a built-in auto-clicking module which runs in the web browser, opening hidden windows to access C&C-provided URLs (which are periodically updated) [21]. An additional module also redirects users themselves to ad landing pages when they perform searches [21].

4 Reasons for Attack Success

Botnets operate in phases with attacks formed of sequential steps. This means that many factors may have impact on attack success along the way. In this section, we identify and discuss some of these factors.

4.1 Pre-Existing Vulnerabilities

Exploitation of software is one of the most common approaches used by malware to infect or attack systems. ZeroAccess uses the BlackHole exploit kit for propagation [21], whilst [31] reports that some Android bots use root exploits for privilege escalation. In the underground community, exploitable areas of code are actively sought out and shared. Therefore, continuous use of old software leaves devices, and subsequently networks, vulnerable as flaws and loopholes become known amongst cyber-criminals. Delaying or disabling of scheduled updates can increase the chances of contracting a botnet infection. However, even with defensive measures in place, all software inevitably contains vulnerabilities that, despite extensive testing and review, can go unnoticed. Therefore, networks should contain layers of defence including scheduled backups, perimeter defences (where applicable), and failover systems.

Probability of attack success is greatly impacted by the behaviour of users. Some botnets rely on user negligence to spread. For example, iKee.B and Mirai both used default login credentials to gain access to target devices [29, 32]. Others count on users lacking the vigilance or skills needed to differentiate between legitimate and malicious applications and services. For instance, Clickbot.A uses Trojans disguised as games [13], Asprox uses a fake Flash Player and anti-virus package [8], and Koobface sends malicious URLs from fake Twitter/Facebook accounts [16]. Each of these approaches require user interaction to reach their goal. From an enterprise perspective, employees may circumvent security policies and guidelines by using company devices for personal tasks (expanding the attack surface), connecting unsecured devices to the network via USB (used by Mariposa for propagation [15]), or not using encryption for sensitive data.

4.2 Malware Variability

Malware increases its chances of success when it manages to deviate enough from known attacks. This is because defences tend to be heavily based on previous experiences. At the initial infection stage, propagation vectors vary wildly. Some vectors observed in the past include drive-by-downloads (Asprox [8]), spam mail (Waledac and Storm [14], Dirt Jumper [18]), instant messaging (SpyEye and

Citadel [25], Mariposa [15]), social networks (Kelihos [19], Koobface [16]), and social engineering (Asprox [8]). This level of variation makes it difficult to have a 'one-size-fits-all' defensive approach, which is further exasperated by the constant emergence of day-0 threats. As a result, (at least) some attacks inevitably slip through the net.

Variation is also born out of customisation and adaptations of existing malware. For example, the original Zeus source code was used as a foundation for the development of new threats, including Citadel and ICE IX [9]. Despite being derivatives of Zeus, these botnets developed into distinct threats in their own right. The same malware was later developed further to become GameOver Zeus, which incorporates the use of P2P, making it more resilient than earlier iterations [9]. Bot malware also evolves in response to defensive measures. An example of this is the use of IRC in early botnets. When defenders started to block the IRC protocol, botmasters began using decentralised networks like P2P, and more inconspicuous, harder-to-block protocols like HTTP.

4.3 Discrete Operation

We have already discussed the measures taken by bot binaries to obfuscate their presence. In addition to this, active bots may avoid engaging in superfluous activities in order to draw less attention to themselves. By remaining dormant, the botnet may improve its longevity, allowing more bots to be cultivated in preparation for a larger attack [35]. Alternatively, propagation may be selective rather than randomised so that only specific domains are targeted and the botnet size is deliberately kept small to avoid detection. Similarly, it may be beneficial to the botmaster to avoid creating active nodes beyond a given time threshold to reduce the botnet's footprint [36]. These characteristics help the malware to stay under the radar.

In a typical enterprise network, traffic is generated by hundreds or thousands of systems, resulting in millions of daily flows. Botmasters may exploit this by blending C&C traffic in with normal user behaviours. For example, the implementation of HTTP as a C&C protocol allows polling bots to circumvent firewalls whilst mimicking benign web traffic. Furthermore, traffic-based detection tends to focus on anomalies or patterns suggestive of unauthorised automated processes. Botnets may therefore evolve to include functions designed to distort these patterns. For example, periodicity is a popular flow-based detection metric, measured as frequency, duration, or intervals lengths [37]. Randomised delays can be added between events to break up periodic patterns in propagation or polling traffic, as demonstrated by [38].

4.4 The Black Market

The current state of the black market is major driver behind the success of today's botnets. In the early days, simple malware threats were thought to be written by amateur individuals, largely for their own entertainment. By contrast, malware development and distribution is now a growing underground business [39]. A modern cyber-criminal does not require prior experience or technical expertise. Sophisticated bot malware and related exploit tools are readily available for purchase, including bot binaries, SQL injection kits and browser exploits [40], to name just a few.

Malware authors themselves stand to make massive profits from this. For example, in 2015 the complete Zeus toolkit reportedly cost between $1,500 and $20,000 depending on the version [24]. Complex business models have also emerged, including pay-per-install, pay-per-use [41], and hacking-as-a-service [40], allowing customers to outsource malware distribution or rent existing botnets for a period of time to suit their needs. With such a high degree of availability and a range of options, it is fair to suggest that any malicious group or individual could feasibly launch their own botnet-based cyber-crime campaign.

5 Existing Solutions and Classifications

Botnet solutions can be broadly classified into two groups; proactive and pre-emptive (Fig. 3). Proactive approaches encompass detection-related tasks, including bot identification, mitigation, removal, and takedowns (which may also be considered pre-emptive in stopping future attacks). Generally, these strategies deal with existing threats. Meanwhile, pre-emptive approaches include modelling and simulation aimed at exploring botnet characteristics for a better understanding of behaviours for predictive purposes. In this section, we provide an overview of both.

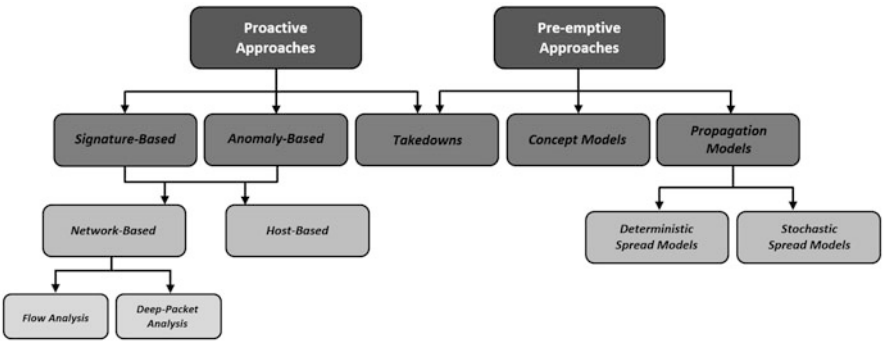

Fig. 3 Classification hierarchy for botnet solutions

5.1 Proactive Approaches

At the highest-level, detection systems can be classified as either signature- or anomaly-based. Signature-based refers to the use of markers which are known a priori to be indicators of malicious activities in the host or network. These 'signatures' are stored within a database, and the detection mechanism is triggered if matching activity is observed. Due to its reliance on heuristics, signature-based detection is highly effective against familiar threats but not for day-0 cases. Rishi [42], a signature-based tool, checks TCP packets containing certain IRC commands for strings that match its collection of known bot keywords.

In contrast, anomaly-based detection generates a baseline of normal activity for the system, triggering when it detects significant deviations from this norm. This is advantageous because unfamiliar threats can also be defended against. However, baselines must be sufficiently representative of actual behaviour, which can be difficult if it is highly random. Unrepresentative baselines can lead to both false positives and false negatives, potentially exasperated by the presence of benign and malicious activities with similar characteristics. An example is BotSniffer [4]. It monitors IRC/HTTP traffic at the network perimeter, which is then checked for group bot-to-C&C communication patterns. Similar tools include BotMiner [43] and BotProbe [44].

Anomaly- and signature-based approaches may be combined so that known threats are handled quickly, or so newly-discovered anomalies are recorded as signatures for future use. Examples of such hybrid schemes are BotHunter [45] and BotCop [46].

Detection systems can be further classified by the scope of their observational areas, as either host- or network-based. Host-based approaches monitor the system-level activities of a single device, including API calls, use of the registry, and generation of outbound traffic [47]. This is advantageous in observing specific infection processes or dormant bots who communicate infrequently. However, this approach alone may be insufficient when there are many connected hosts to secure. An example of host-based detection schemes is BotTracer, which targets automated bot installation and rally processes by isolating them via a virtual machine it runs and monitors on the host [47].

A network-based system observes the traffic generated between multiple hosts, and is placed strategically at ingress/egress points to capture the most relevant or risky traffic. Suspicious behaviours that may be flagged include failed connection attempts [48], failed DNS requests [30], web connections to blacklisted sites [49], encrypted payloads in outgoing messages [50], and use of randomised domain names [49]. Network-wide monitoring provides an overview of all activity in the observation space, and is beneficial in detecting behavioural patterns, fluctuations and group actions [43]. The main disadvantage is the sheer volume of data generated daily, requiring a greater amount of resources to collect and process.

This approach can be divided into the flow and deep-packet analysis sub-categories. Traffic data is often collected as NetFlow logs. A flow is the aggregation

of packets exchanged between two endpoints in a specific conversation, and consists of a vector of features (including timestamps, transport protocol, ports, IP addresses, and a range of statistics) which provide a summary of that exchange. Flow analysis is therefore a shallow but fast examination of traffic, using features like duration, length, size, and frequency for statistical analysis. Bots generate traffic automatically in response to commands, resulting in more systematic flows than that of normal users. Therefore, detection using flow analysis involves searching for automated processes which manifest as repetitive anomalous patterns, or searching for instances of anomalous group behaviour. A drawback of this approach is the number of false positives caused by benign automated traffic picked up erroneously. BotMiner [43] is a flow-based detection scheme. It collects communication (C-plane) and activity (A-plane) traffic and clusters the flows. Then cross-plane correlation is performed to identify suspicious behaviour [43]. BotCop [46] is another example.

The alternative to this traffic aggregation is to use packet captures (PCAPs) which record the full details of every packet exchanged. This is used in deep-packet analysis to examine protocol implementations and payloads, providing richer detail, and is particularly useful when researching new malware or observing custom protocols. However, it is costly in terms of time and processing power, and payload examination is typically unsuccessful when dealing with encrypted traffic. However, [51] developed BlindBox to perform deep-packet analysis on HTTPS traffic by adapting signature-based methods to encrypted payloads.

Flow and deep-packet analysis can be combined, as demonstrated by [52]. They first use statistical analysis to extract sets of suspicious flows. Then, false positives are minimised by performing fine-grained deep analysis on only the packets of those flows.

A takedown is the process of infiltrating and hijacking a botnet's C&C infrastructure to disable communication between servers and nodes [53]. Sinkholing is the most common method for achieving this, where botnet domains are identified and pre-emptively purchased by researchers, who then setup their own servers to receive incoming bot communication [54]. The ethics of sinkholing (including what to do with harvested user data) is an open debate [54]. The first successful takedown was against the Conficker botnet in a collaborative effort between Conficker Working Group (consisting of researchers and industry professionals) and ICANN, who helped to take control of Conficker's domains [53, 55]. These were shared with registrars who blocked them, cutting off the botnet's C&C channels [55]. This was followed by other successful takedowns of Mariposa in 2009 [53], Waldeac in 2010, Rustock and Kelihos in 2011, Zeus and Nitol in 2012, and Batimal and Citadel in 2013 [55].

Figure 4 illustrates how proactive approaches are implemented and which part of the botnet they address. Step 1 highlights a bot-infected client aiming to communicate with a C&C using a hardcoded domain name. The blue dashed line represents the end-to-end communication between the bot and the server. Client devices are equipped with host-based monitors, which observe unusual system calls triggered by the malware. At 2, the request is picked up by the network-based

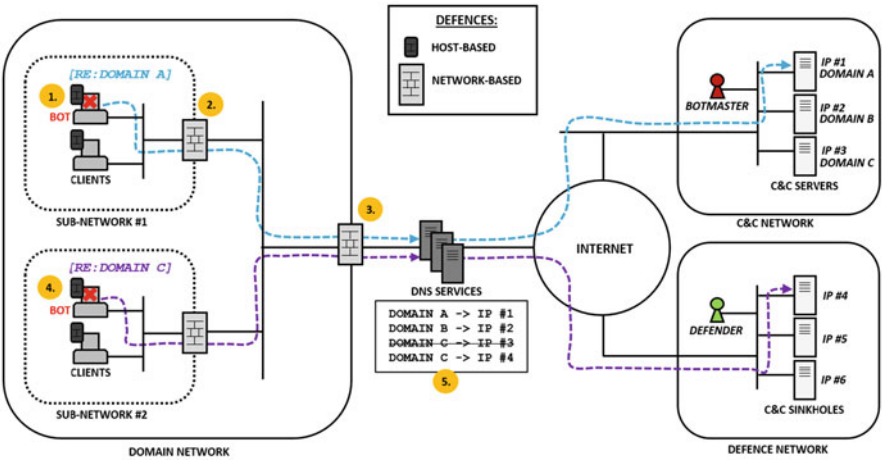

Fig. 4 Example of a bot-infected network illustrating the application of proactive defences

monitor of the local network. This observes all local clients, recording patterns if they exist. Similarly, at 3, the same request hits the network monitor of the wider domain network. The observational scope is expanded and, where multiple bots exist, group behaviours can be identified. Step 4 highlights another bot making a separate connection request. However, this time, the DNS registry has been modified so that the domain resolves to a sinkhole run by defenders. The dashed purple line represents hijacked bot communications in a botnet takedown effort.

5.2 Pre-emptive Approaches

We categorise pre-emptive works as propagation models and concept models. Propagation modelling aims to identify factors contributing to bot spread and the relationships between them, allowing researchers to explore both offensive and defensive possibilities. Meanwhile, concept models are explorations of new architectures or attack paradigms. The aim is to predict how botnets may adapt to current defences or utilise new technologies as a means of prevention. The following sections highlight some examples of pre-emptive research.

5.2.1 Propagation Models

Most botnet propagation studies are based upon state-based, compartmental transition models from the field of epidemiology. These models divide individuals from the total population into states representative of their condition at a given time [56]. Some examples are *S-I-R* (susceptible, infected, recovered),

S-I-S (susceptible-infected-susceptible), and *S-E-I-R* (susceptible-exposed-infected-recovered), though many more variations exist. The transition of individuals between groups is captured by a set of differential equations, where the rate of change for each state is given by a time derivative [56]. In this way, the progress of an infectious disease is tracked through a susceptible population. The results can then be used to identify the best vaccine provision methods. When applied to botnets, the susceptible population is represented by the nodes of the observed domain, the infected population aligns with bot-infected nodes, and the recovered population tends to denote patched or cleaned nodes.

Understanding the propagation characteristics of a bot malware can help predict the speed and reach of spread, potential botnet strength, mitigating factors, and may even provide clues as to the botmaster's intentions. In [57], the authors used sinkholes with known C&C IPs to capture bot traffic. They counted active bot populations (using 3-way TCP handshakes and preventing disconnections to minimise DHCP churn) and found that bot activity appears to be diurnal in nature, probably due to targeted devices being powered down each evening [57]. Based on this they adapted the *S-I-R* model for worm-based propagation to capture this behaviour across time zones. They then used this to make predictions about optimal worm release and patching periods [57].

In another work [27], researchers combined a *S-I-R-S* variation with fast evolutionary game theory to model competitive and cooperative behaviours between botnets. The two interacting botnets are given unique characteristics, where the first can enable/disable local defences, whilst the second can kill-off the first botnet's nodes. A cooperative strategy is where defences are disabled and/or no nodes are terminated. A competitive strategy is the opposite scenario [27]. Based on their tests, the authors derived thresholds for the defensive presence needed to minimise chances of cooperative behaviour. They also reported that cooperation significantly decreased the extinction probabilities of both botnets even at low infection rates, whilst a competitive strategy lead to the first botnet being killed off by the second [27].

Standard epidemic models as described previously are deterministic. In [58], the authors develop the probabilistic *S-I-C* model (susceptible-infected-connected). *I* nodes carry the minimal amount of malware code for compromise, but don't communicate with the C&C until they transition into *C* nodes. Continuous-time Markov Chain modelling is used to determine stochastic transition rates [58]. Their aim was to measure the overall growth of P2P botnets, and to test the effectiveness of various mitigation strategies. Using the model, they studied the impact of sybil attacks (where clean nodes join the botnet to corrupt the C&C infrastructure by sharing false information), and identified a relationship between the number of sybil inserts required to reach a desired mitigation in growth [58].

Meanwhile, [59] abstracted epidemic concepts to build a stochastic model of botnet propagation in mobile networks. Specifically, they aimed to observe the impact of moving nodes and node proximity on the speed of spread. The network consists of user and infrastructure nodes, with transmission range and total bandwidth as parameters [59]. *S* is defined as a population of infected nodes,

representing the mobile botnet. Based on population density and node mobility range, they identified a threshold which when exceeded causes the botnet to grow at a quadratic rate. If not exceeded, the botnet size is predicted to have a finite limit [59].

These examples – which are just a few – demonstrate how propagation models can be used to explore factors which contribute to the growth of botnets in different environments and under different constraints. Simulations allow researchers to experiment with scenarios which may not be practical or safe in real networks, whilst empirical data collected from botnets in the wild can be used to validate models or serve as input data. The main drawback of this approach is that whilst using epidemic models, generalised assumptions are frequently made about homogenous populations and mixing that can make the results somewhat unrealistic. In [60], the authors outline some of the challenges that need to be addressed when applying epidemic modelling to network-based environments.

5.2.2 Concept Models

In [38], the authors propose the delay-tolerant botnet, designed to add random delays to its C&C traffic to disrupt any observable patterns in its communication. It is assumed that the botnet can handle delays because functionality can be maintained even if member bots don't all receive commands at the same time [38]. For randomly selected delay durations, they develop schemes for centralised and decentralised architectures where delays are inserted between bot-initiated connections and data forwarding events, respectively. They also experiment with hybrid architectures using a combined approach. They suggest that the overall delay (i.e. time-taken for all bots to receive a command) must have a logarithmic correlation with botnet size to maintain scalability [38].

In [61], the authors introduced the concept of alias-fluxing, which is the use of URL shortening services to obfuscate C&Cs. This is based on existing fast- or domain-fluxing mechanisms. The C&C network is made up of proxies and the main C&C that the botmaster controls. Using an algorithm, similar to a DGA, the botmaster generates a number of shortened URL strings every period, and registers these with the relevant services for each of the proxy servers. The bots use the same algorithm, or retrieve short URL lists, which they then use when rallying [61]. The authors suggest that this approach circumvents DNS-based detection, and obfuscates traffic where HTTPS is used [61].

Xu et al. [62] explored the use of DNS as a pull-based C&C protocol. Assuming the botmaster has access to a DNS server (setup with short TTL), they defined 2 scenarios; code-word and tunnelled, for unidirectional and bidirectional communication, respectively. In the first, the botnet uses a set of codewords which denote certain functions and requests [62]. Bots make DNS queries for domains containing a specific code-word, to which the DNS server replies with the appropriate data. Meanwhile in tunnelled mode, requests from the bot consist of CNAME queries to which the server responds with encoded CNAME records [62]. They also demonstrate that Markov Chains can be used to generate code-word-containing domain names in such a way as to avoid detection [62].

The authors of [5] built a prototype of a botnet which uses Cloud-to-Device Messaging (C2DM) as a push-based C&C network. This Google service is designed to efficiently incorporate push notifications into apps for Android devices. The botmaster registers an account for the malware package with the C2DM. This account is identified by a unique username which is then distributed with copies of the package to target devices [5]. Clients joining the botnet connect with C2DM servers using the same username, and receive a registration ID which they then share with the C&C server. Then, the C&C server can send its messages to the C2DM service with its authentication details, and the service will forward them on to the bots [5]. Noting the restrictions on the number of push messages per period, the authors suggest that larger botnets may be built from a series of smaller sub-botnets. They also suggest that communication can be further obfuscated by using 2 separate C2DM accounts (one for the C&C and one for the bots) [5].

The research outlined here shows how new perspectives and approaches may be explored. By adopting the position of the botmaster, the authors imagine how they may adapt to the changing digital landscape, what avenues might be most profitable, and how to avoid detection. The ideas proposed therefore give researchers the chance to design better, forward-thinking defensive solutions. Combined with the lessons learnt from propagation modelling (as well as historic botnet attacks), defenders aim to pre-empt threats and mitigate risk.

6 New Flavours

In this study so far, we have generally assumed that the targets of botnets are average LAN or WAN networks. However, modern technology has brought about many new scenarios, some of them pertaining to specific or specialised types of network. In this section, we highlight some of these and the issues surrounding them.

6.1 Mobile Botnets

Mobile devices are now ubiquitous. They are always with us, always in use, and always connected. This flexibility, sustained connectivity (on multiple channels), and lack of diurnal behaviour [26] makes mobile a viable botnet platform. Mobile devices use Internet, WiFi (both infrastructure and ad-hoc), SMS/MMS, and Bluetooth to connect with other devices, providing several channels for both attack and propagation. The battery life and processing power of mobile devices have improved, allowing bots to carry out complex tasks. Furthermore, apps like those provided by the Google Android and Apple iOS marketplaces serve a large range of functionality – users interact with many services (e.g. finance, commerce) sharing lots of valuable data. Apps are also an ideal infection vector, as botmasters can circumvent marketplace regulations to use their own Trojans or hijack existing offerings.

The iPhone-based iKee.B is a classic example of mobile botnets [32]. It uses Internet-connectivity to spread, scanning for phones running SSH services. Porras et al. [32] reports 3 distinct methods; scanning of specific IP ranges belonging to Australian and European mobile operators, scanning of a randomly chosen subnet on the Internet (using a time-seeded algorithm), and scanning of the client's current local network. Hence, propagation is a blend of Internet- and mobile-based methods, with device mobility exploited to attack each new local network it joins. The bot then uses default passwords to get shell access and once installed, shares host data (including archives of SMS messages) with C&Cs via HTTP [32]. Unique IDs are used for bots, allowing them to be tracked despite changing IPs across zones [32].

Zhou and Jiang [31] characterise features of Android-based botnets. They identify drive-by-downloads (with multiple redirections), repacking (malware embedded within legitimate apps), and update attacks (malware injected into apps via fake updates) as the main infection vectors. Some examples include Anserver and Plankton who use partial updates for added discretion, and the Jifake bot which triggers redirections via fake QR codes [31]. Meanwhile, data theft is identified as a key attack type, with targets including SMS messages, phone numbers, email addresses, and user accounts for third-party services. Another popular attack is where bots covertly sign-up to corrupt "services" which they poll via SMS, incurring premium-rate charges for the victim. As with iKee.B, HTTP-based C&C is most commonly observed [31].

Current botnets rely on the Internet for their C&C backends, but we can expect this to expand into other available channels. For example, [5] demonstrated the use of push notifications, whilst [63] showed the Bluetooth is also a viable option. As a larger number of service providers offer app-based interfaces, the pool of data to be harvested by bots will also expand. Furthermore, hacked accounts may act as vectors to target those services themselves. As mobile devices are used to interface with social media, the IoT, and the cloud, we could see cross-platform botnets using a mix of attack and propagation vectors. Finally, device mobility makes it difficult to locate propagation sources or to apply defences where there are no set perimeters. Ad-hoc networks may be used to obfuscate bot nodes, making DDoS attack sources difficult to trace [26] and botnets harder to enumerate.

6.2 Social Network Botnets

Social networking is a staple of modern communication and, as a result, has become a target for botnets. Social networks are free with simple registration processes, accessible from anywhere via mobile or desktop devices. This makes them easy to infiltrate. The platforms sustain massive user bases, who frequently submit personal details (which can be stolen) and exchange multimedia messages, giving botmasters ample possibilities for spam and malware delivery. These platforms provide both attack and propagation vectors, as the spreading of information throughout the network is inherent to their design. Additionally, user-to-user relationships are often

ambiguous, with individuals lacking in awareness of potential risks. Bots may therefore easily infiltrate communities. Finally, resiliency and upkeep is handled by the platform/service providers, easing the maintenance of the botnet for the botmaster.

Koobface is a social network botnet, and was studied by [16]. They report that it exploits Twitter and Facebook as gateways to compromise user devices. It targets existing accounts, whilst simultaneously creating new accounts using fake personal details provided by the C&C. CAPTCHAs served during this process are forwarded to bots for real users to solve [16]. Fake accounts spread malicious links as short URLs, which redirect users to web pages encouraging them to download Trojan software. Once infected, bots connect to master C&Cs (who serve spam commands) and upload statistics on their activities. Meanwhile, existing accounts are given keywords to determine the communities they must join or establish friendships in [16]. This demonstrates how propagation can be targeted at particular groups.

The "Bursty Botnet", documented by [64], is similar. It exploits Twitter by creating new accounts which then disseminate spam, even using mentions (messages containing a specific user's Twitter handle) to entice users to click on malicious links. During their study, [64] identified 500,000 members of this botnet, with a total of 2.8 million spam tweets generated. Like Koobface, Bursty uses URL shortening to obfuscate targets. Bursty bots are designed to only tweet several times in a short period after their creation (hence the name), before falling permanently silent [64]. This one-time-use policy is probably a resiliency measure; the botnet should still function even if many nodes were taken offline. Silent bots may also avoid detection, but can be reactivated later [64].

Whilst Twitter and Facebook are currently the main targets, social botnets could expand to other platforms. For example, LinkedIn may be used to identify employer-employee relationships for targeted spam and even corporate espionage. Bot programs are developing to better mimic user behaviour, but may not even be needed. In their design of Stegobot, [65] demonstrated a C&C channel made only of existing social links, without the need for fake accounts or relationships. Furthermore, social networks make revenue from advertising – botnets may expand their functionality to include click-fraud. Lastly, from a socio-political perspective, social networks can be exploited to quickly spread misinformation or target groups for manipulation, as demonstrated by Koobface [16].

6.3 IoT-Based Botnets

Internet-of-Things (IoT) is the concept of adding network connectivity and automation to sensors, specialised devices, and every-day appliances to provide better services. The IoT is an attractive target for botmasters because its deployment vastly increases the population of vulnerable nodes which can be converted into bots. Devices which have traditionally always been offline now come packaged with Internet access (and often lacking sufficient security provisions), expanding

the reach of botnets into sectors/domains which previously wouldn't have been exploitable, e.g. vehicle networks. Some IoT devices like sensors are low-energy consuming by design and hence do not have the processing power for complex security mechanisms. User awareness also plays a role – many may not be aware of built-in IoT functionality when purchasing new electronics.

In 2016, the IoT botnet Mirai was used to launch several high-profile DDoS attacks, most notably targeting a French web hosting service with a massive force of between 1.1 and 1.5 Tbps [33]. At this time, this botnet reportedly consisted of home routers and digital cameras [33], demonstrating how simple devices may be recruited into large botnets with highly potent attacks. Mirai propagates by scanning address spaces for Linux-based platforms [29], and then uses a simple database of 62 default login credentials to gain access [29, 33]. During scanning, [29] reports that the malware is hardcoded to avoid certain domains such as the US Postal Service, the US Department of Defence, and IANA. This is probably a precautionary measure. Bots collect and forward device details to the C&C, which the botmaster accesses over Tor for additional anonymity [29]. Analysis of Mirai's binaries reveal both network layer (SYN floods) and application layer (HTTP floods) DDoS functionality [29].

Mirai could become the IoT version of Zeus [24]; its source code has been made public and since then, variations with new features (e.g. enhanced encryption) have been reported [29]. This shows the level of criminal interest in IoT systems, and reinforces the potential risks involved in rapid deployment. IoT devices could be used to infiltrate particular sectors of industry for surveillance and espionage, as well as theft of sensor data. The range of use cases means there are no standard configurations, protocols, or known patterns on which to base generalised defensive approaches [33]. Furthermore, manual interaction with every IoT device is not scalable [33]. Combined with the limited processing capability of some devices, this means that security could be neglected. IoT is still a young technology which means we currently have limited experience of possible flaws and their implications. Therefore, to prevent significant risks, protocols and devices must be secure-by-design.

6.3.1 A Note on IPv6

Whilst IPv4 is the current dominant addressing protocol, organisations will increasingly have to shift to IPv6 in order to accommodate the massive influx of devices caused by the IoT. IPv6 uses 128-bit addresses (as opposed to the current 32-bit), providing a much larger address space of unique IPs. The robustness of IPv6 security is still largely unknown [66], and by removing the need for NAT and DHCP, it makes current control mechanisms obsolete. This has serious implications for existing botnet defences. This is further exasperated by IPv6's dynamic addressing function, as perimeter defences and blacklists become ineffective [66]. These issues apply to both IoT and non-IoT networks, and must be addressed before wide-scale adoption of the protocol.

6.4 Cloud-Based Botnets

Cloud computing is built on the business paradigm of computational resources as-a-service, always online and accessible via the web. It provides systems of servers and clients with built in redundancy, easy accessibility and remote storage. These characteristics make them ideal platforms to exploit as C&C networks. For instance, Plankton uses C&Cs in the Amazon cloud where its payloads are processed [31]. The centralised architecture means many potential bot clients are accessible (perfect for spam or malware delivery), whilst resource-rich servers can provide high-levels of processing power and storage for the botnet back-end. Web-based interfaces for services may be used to reach new targets, e.g. by redirecting users to fake versions or by intercepting login credentials. Cloud services are also used by large corporations and organisations – the cloud may be attacked by botnets to ultimately cause harm to those organisations.

Chen et al. [67] proposes a botnet with a cloud-based C&C network using push notifications on mobile devices. This approach is similar to [5], but unlike that model, [67]'s design uses a range of cloud services (e.g. Airbop, JPush, and Google Cloud) for a network of multiple push servers. First, new bots register with a local server. Using a transmission delay metric, bots with similar delays are clustered together (assuming this means they currently sit in the same region). C&Cs then use a round-robin schedule to cycle through push servers to disseminate commands, where servers with the least delay appear more frequently on schedules for that region [67]. This distributes command traffic, making it harder to detect, whilst delivering messages efficiently. Chen et al. [67] suggest that botnet functionality can be maintained even if some push servers go offline.

Meanwhile, [68] proposes cloud-based bots that can be used to launch slow-read DDoS attacks. This type of DDoS uses TCP or UDP to setup connections with small windows, forcing servers to keep connections open for extended periods. As a result, fewer back-to-back connections are required, and consequently, fewer bots. As requests are processed slowly, these attacks are also more difficult to detect [68]. Bot malware is not propagated in this setup. Instead, the botmaster uses image files containing all the required malicious functionality, registers with a cloud service and loads the image onto multiple cloud-based virtual machines (a sustainable approach for smaller botnets) [68]. Bot images also speed up the setup process, giving defenders less time to detect the initial compromise before further attacks are launched.

Malware with standard propagation capabilities could spread within the cloud infrastructure to create computationally powerful botnets with unique attack capabilities (dependent on the available cloud services). Pay-as-you-go provision models may be exploited via fraudulent resource consumption, leading to financial loss for users or reduced performance for the provider [69]. The cloud itself could be targeted in DDoS attacks, causing disruption to millions of users, or as a mechanism to infect the domains of cloud customers by piggybacking on services.

6.5 Crypto-Mining Botnets

Cryptocurrency is designed to be anonymous, decentralised, and unregulated; characteristics that directly align with the requirements of botmasters. Botnets can be built using the Bitcoin infrastructure, exploiting its use of P2P to maintain the blockchain as a C&C channel [70]. The built-in anonymity of users makes it harder to identify the botmaster or to enumerate bots across the network, even if individual bots are discovered [70]. This is a vast improvement in resiliency over typical Internet botnets as previously discussed. Anonymisation is supported by the fact that all transactions look identical to the observer, allowing malicious activities to stay hidden. Such activities are also not easily disrupted as direct action against suspected users would result in large-scale disruption to the blockchain as a whole [70]. Lastly, bots can be arranged into crypto-mining pools to generate their own Bitcoins for income.

The so-called Miner botnet was studied by [28], and is made up of 4 tiers with the top 2 forming the C&C infrastructure and bottom 2 containing the infected clients. P2P bots with Internet connectivity at tier 3 run a special Bitcoin mining module in the binary. Tier 4 bots (who lack Internet connectivity) then use the P2P bots as relays to download Bitcoin client software and to join mining pools from a hardcoded list. Coins and wallets are backed-up to tier 3 bots, which in turn upload this data to the C&C in 20 min intervals [28]. When first infecting a victim, Miner reportedly checks video and graphic card drivers, updating them if necessary and running speed tests to determine the host's capabilities [28].

ZombieCoin is a conceptual botnet with a Bitcoin-based C&C architecture, where commands are encoded into transactions between the botmaster and bot clients [70]. A pair of keys are generated by the botmaster, with the public key hidden in the bot binary. When bots join the Bitcoin network, this key and the botmaster's digital signature are used to identify and interpret commands. Ali et al. [70] identify a selection of methods. The first is to use OP_RETURN, a function allowing 80 bytes of ID data to be added to a transaction. Second is the use of unspendable outputs, which allow the addition of 20 bytes of custom data. The third is to use key leakage; a random factor is used twice, enabling the derivation of the signer's private key. Last is the use of subliminal channels where commands are hidden within signature strings [70]. These options show how resourceful botmasters can build complex C&C structures which would be very difficult to detect and dissolve in real life.

Extending their work on ZombieCoin, [70] suggest the use of rendezvous points, disseminated in near real-time to all bots telling them when/where to upload data, as well as transaction chaining where long commands are encoded as the inputs/outputs to an ordered sequence of transactions. Attacks may include selfish mining (where blocks are used to build private chains for mining without competition) and BGP hijacking (where traffic to and from the blockchain is interrupted, partitioning the network) [71]. Another possibility is an eclipse attack where currency is stolen directly from compromised devices by capturing their

communications [71]. Cryptocurrency is increasingly utilised for black market transactions, making it hard to track the exchange of botnet tools and services. Furthermore, blockchain technology has so far been used primarily for cryptocurrencies, but its functionality is expanding into other applications [71], extending the botnet attack plane.

7 Future Developments and Possible Research Directions

Botnets are constantly adapting to changes in the digital landscape. We must match this with our detection and mitigation strategies to respond to the threat effectively. Here, we summarise some outstanding issues and suggest areas for further research.

Propagation is a difficult activity to detect and observe due to the sheer variability of infection types, vectors and mechanisms, all of which are designed to be discreet. Historic traces of past propagation events are not made publicly available due to privacy concerns, and data of this kind is difficult to accurately simulate in lab environments. Therefore, there is a need for improved propagation research, with high-quality, rich datasets and shared testing environments made widely available to researchers so that better predictive models (with better applicability to real-life) may be built.

In addition to this, most existing work in this area is directly based on compartmental epidemiology. Epidemic models are well-established and useful as a foundation for understanding spread dynamics, but several outstanding challenges remain in their application to network environments, including considerations of complexity, heterogeneity, and prediction accuracy [60]. Furthermore, there is a need to expand past epidemiology, and to explore other characteristics of spread that are domain-specific, including different business settings, different types of devices, and other influences on potential infection reach and rate.

Around the world, many institutions and individuals have studied botnets over the years, and categories of distinct detection and mitigation approaches have emerged as a result. As a community, we have also established classifications for botnet types and behaviours. However, there is still no generalised standard for the measurement of botnet characteristics, like size, infectiousness, force of propagation, or various traffic metrics. The same can be said of the measurement and characterisation of vulnerable networks. A widely-accepted standard for determining the degree of vulnerability and variability would help in building better approaches and for more accurate comparisons between detection schemes.

Recently, we have seen a general push in all areas of cyber-security towards the use of artificial intelligence and machine learning. Whilst this comes with its own set of challenges, this approach can be particularly effective when large, accurate and relevant datasets are available to train predictive models. However, the role of the human analyst is still significant in adaptively and flexibly interpreting new, unfamiliar and evolving threats. Therefore, future approaches shouldn't rely solely on machine learning or AI to perfectly protect against botnets, and should instead be a balance between machine and human interpretation. Future systems should be

efficient and flexible, with intuitive visualisation for defenders to interface with data for improved analysis and response.

The research community needs to maintain awareness of the botnet threat when developing and deploying new protocols and technologies. Security must be built-in at the design phase and shouldn't consist primarily of reactionary patches. We previously discussed risks related to mobile technologies, the IoT, and the IPv6 protocol. Other topics for consideration include new protocol versions like HTTP/2 [72] and DNSsec [73], anonymization services like Tor [74], and biometric security [75]. Researchers needs to develop concept models to creatively explore these technologies, imagining new attack scenarios, new targets of interest for adversaries, and potential vulnerabilities to bolster industry standards against future botnets.

In order to achieve the highest possible level of defence, there must be a greater level of global collaboration between researchers, vendors, network infrastructure providers, and law enforcement agencies. This is vital in dealing with threats effectively and efficiently, as demonstrated in numerous successful botnet takedowns [53–55]. Collaboration may be also happen through the sharing of resources and research platforms (e.g. the DETER TestBed [76]). Security needs to extend beyond business and government networks. For example, the Stratosphere Project aims to empower NGOs by providing free botnet defence tools [77]. Users should also be personally empowered to actively maintain their privacy. A possible path to achieving this is open-source software, though there is some debate as to whether this is more or less secure than proprietary solutions [78].

Another area for development is in legislation and policy. As technology has developed in past decades, legislation around the world has struggled to keep up [55]. The rapid dissemination and adoption of new technology results in legal grey areas which can be exploited by criminals, but also by legitimate businesses and states for profit, power, or surveillance. Such areas in need of clear and robust laws include user privacy, collection of user data, and remote admin of private systems. The reactive nature of cyber-crime legislation risks harsh laws that address symptoms rather than causes, and ultimately threaten the future of a free Internet. Botnets are a unique and complex threat, their impact dependent on context and environment. For the future Internet to be secure whilst remaining democratic and free, new legislation (and the underpinning technology) must balance the rights of individuals with the safe and secure provision of services.

Finally, botnet malware has the potential to cause significant damage to national infrastructure. Combined with the possibility of state-sponsored attacks, this has large implications about the future of warfare and conflict. Hence, nations need to invest in securing critical infrastructure. This requires significant research into the defence of highly specialised systems against sustained, sophisticated attacks. With botnet tools highly available to anyone on the Internet, the potential for large-scale cyber-terrorism must be a major consideration.

8 Conclusions

In this chapter, we have outlined the basics of what a botnet is, how it works, possible variations, and how they are categorised. We defined botnet attacks in 2 distinct parts; the initial compromise and the continuous threat. The former encompasses the infection of a domain, including installation, system manipulation, and propagation activities. Identification of the malware at this stage mitigates the development of the botnet. The latter denotes the state of the domain after a successful compromise, where the botnet is ready to be utilised for a variety of attacks. At this stage, detection is a damage limitation exercise. Building on this, we discussed the impact of infections on the host domain, the attack process, and factors contributing to successful deployment.

We then outlined current defensive strategies, categorising them as proactive (practical solutions for securing real networks) and pre-emptive (theoretical explorations for better understanding and prediction). We demonstrated the adaptability of botnets by discussing new types of bot malware on platforms like mobile, social networks, IoT, the cloud, and blockchain. Finally, we considered some unresolved issues, and made suggestions for future research. Botnets are highly versatile, powerful multi-function tools. They are relatively easy to setup and difficult to take down, and can impact any area of digital infrastructure. Therefore, we need to have equally powerful defensive strategies, with a concentrated focus on early detection and a culture of security awareness.

References

1. R. J. Oentaryo, A. Murdopo, P. K. Prasetyo, and E.-P. Lim, "On Profiling Bots in Social Media", in *International Conference on Social Informatics*. Springer, 2016, pp. 92–109.
2. R. A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Survey and Taxonomy of Botnet Research through Life-Cycle," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 45, 2013.
3. S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A Survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.
4. G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in *NDSS*, vol. 8, 2008, pp. 1–18.
5. S. Zhao, P. P. Lee, J. Lui, X. Guan, X. Ma, and J. Tao, "Cloud-Based Push-Styled Mobile Botnets: A Case Study of Exploiting the Cloud to Device Messaging Service," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 119–128.
6. S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the Storm and Nugache Trojans: P2P is Here," *USENIX*, vol. 32, no. 6, pp. 18–27, 2007.
7. H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the Analysis of the Zeus Botnet Crimeware Toolkit," in *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*. IEEE, 2010, pp. 31–38.
8. R. Borgaonkar, "An Analysis of the Asprox Botnet," in *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*. IEEE, 2010, pp. 148–153.

9. D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos, "Highly Resilient Peer-to-Peer Botnets are Here: An Analysis of Gameover Zeus," in *Malicious and Unwanted Software: "The Americas"(MALWARE), 2013 8th International Conference on*. IEEE, 2013, pp. 116–123.
10. P. Wang, S. Sparks, and C. C. Zou, "An Advanced Hybrid Peer-to-Peer Botnet," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, 2010.
11. H. R. Zeidanloo and A. A. Manaf, "Botnet Command and Control Mechanisms," in *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*, vol. 1. IEEE, 2009, pp. 564–568.
12. K. Chiang and L. Lloyd, "A Case Study of the Rustock Rootkit and Spam Bot," *HotBots*, vol. 7, pp. 10–10, 2007.
13. N. Daswani and M. Stoppelman, "The Anatomy of Clickbot. A," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*. USENIX Association, 2007, pp. 11–11.
14. G. Sinclair, C. Nunnery, and B. B. Kang, "The Waledac Protocol: The How and Why," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*. IEEE, 2009, pp. 69–77.
15. P. Sinha, A. Boukhtouta, V. H. Belarde, and M. Debbabi, "Insights from the Analysis of the Mariposa Botnet," in *Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*. IEEE, 2010, pp. 1–9.
16. K. Thomas and D. M. Nicol, "The Koobface Botnet and the Rise of Social Malware," in *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*. IEEE, 2010, pp. 63–70.
17. P. Celeda, R. Krejci, J. Vykopal, and M. Drasar, "Embedded Malware - An Analysis of the Chuck Norris Botnet," in *Computer Network Defense (EC2ND), 2010 European Conference on*. IEEE, 2010, pp. 3–10.
18. M. M. Andrade and N. Vlajic, "Dirt Jumper: A Key Player in Today's Botnet-for-DDos Market," in *Internet Security (WorldCIS), 2012 World Congress on*. IEEE, 2012, pp. 239–244.
19. M. Kerkers, J. J. Santanna, and A. Sperotto, "Characterisation of the Kelihos.B Botnet," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2014, pp. 79–91.
20. A. Rahimian, R. Ziarati, S. Preda, and M. Debbabi, "On the Reverse Engineering of the Citadel Botnet," in *Foundations and Practice of Security*. Springer, 2014, pp. 408–425.
21. P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, V. Paxson, S. Savage, and G. M. Voelker, "Characterizing Large-Scale Click Fraud in ZeroAccess," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 141–152.
22. J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-Peer Botnets: Overview and Case Study," *HotBots*, vol. 7, pp. 1–1, 2007.
23. W. Chang, A. Mohaisen, A. Wang, and S. Chen, "Measuring Botnets in the Wild: Some New Trends," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ACM, 2015, pp. 645–650.
24. R. Layton and A. Azab, "Authorship Analysis of the Zeus Botnet Source Code," in *Cybercrime and Trustworthy Computing Conference (CTC), 2014 Fifth*. IEEE, 2014, pp. 38–43.
25. A. K. Sood, S. Zeadally, and R. J. Enbody, "An Empirical Study of HTTP-based Financial Botnets," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 236–251, 2016.
26. M. Anagnostopoulos, G. Kambourakis, and S. Gritzalis, "New Facets of Mobile Botnet: Architecture and Evaluation," *International Journal of Information Security*, vol. 15, no. 5, pp. 455–473, 2016.
27. L.-P. Song, Z. Jin, and G.-Q. Sun, "Modeling and Analyzing of Botnet Interactions," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 2, pp. 347–358, 2011.
28. D. Plohmann and E. Gerhards-Padilla, "Case Study of the Miner Botnet," in *Cyber Conflict (CYCON), 2012 4th International Conference on*. IEEE, 2012, pp. 1–16.

29. G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai Botnet and the IoT Zombie Armies," in *Military Communications Conference (MILCOM), MILCOM 2017–2017 IEEE*. IEEE, 2017, pp. 267–272.
30. R. Sharifnya and M. Abadi, "A Novel Reputation System to Detect DGA-based Botnets," in *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on*. IEEE, 2013, pp. 417–423.
31. Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 95–109.
32. P. Porras, H. Saidi, and V. Yegneswaran, "An Analysis of the iKee. b iPhone Botnet," in *International Conference on Security and Privacy in Mobile Information and Communication Systems*. Springer, 2010, pp. 141–152.
33. E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.
34. A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 3–24.
35. S.-W. Kim, J.-H. Park, E.-D. Lee, M.-E. Choi, and S.-W. Seo, "Threat Analysis of Incubation Period in Malware Epidemics," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*. IEEE, 2010, pp. 1–5.
36. S. Eshghi, S. Sarkar, and S. S. Venkatesh, "Visibility-Aware Optimal Contagion of Malware Epidemics," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5205–5212, 2017.
37. M. Eslahi, H. Hashim, and N. Tahir, "An Efficient False Alarm Reduction Approach in HTTP-based Botnet Detection," in *Computers & Informatics (ISCI), 2013 IEEE Symposium on*. IEEE, 2013, pp. 201–205.
38. Z. Chen, C. Chen, and Q. Wang, "On the Scalability of Delay-Tolerant Botnets," *International Journal of Security and Networks*, vol. 5, no. 4, pp. 248–258, 2010.
39. D. Bradbury, "The Metamorphosis of Malware Writers," *Computers & Security*, vol. 25, no. 2, pp. 89–90, 2006.
40. G. Ollmann, "Hacking as a Service," *Computer Fraud & Security*, vol. 2008, no. 12, pp. 12–15, 2008.
41. G. Bottazzi and G. Me, "The Botnet Revenue Model," in *Proceedings of the 7th International Conference on Security of Information and Networks*. ACM, 2014, p. 459.
42. J. Goebel and T. Holz, "Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation," *HotBots*, vol. 7, pp. 8–8, 2007.
43. G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection." in *USENIX security symposium*, vol. 5, no. 2, 2008, pp. 139–154.
44. G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active Botnet Probing to Identify Obscure Command and Control Channels," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 241–253.
45. G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," in *USENIX Security Symposium*, vol. 7, 2007, pp. 1–16.
46. W. Lu, M. Tavallaee, G. Rammidi, and A. A. Ghorbani, "BotCop: An Online Botnet Traffic Classifier," in *Communication Networks and Services Research Conference, 2009. CNSR'09. Seventh Annual*. IEEE, 2009, pp. 70–77.
47. L. Liu, S. Chen, G. Yan, and Z. Zhang, "BotTracer: Execution-based Bot-like Malware Detection," *Information Security*, pp. 97–113, 2008.
48. G. K. Venkatesh and R. A. Nadarajan, "HTTP Botnet Detection Using Adaptive Learning Rate Multilayer Feed-Forward Neural Network," in *WISTP*. Springer, 2012, pp. 38–48.
49. S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based Botnet Tracking and Intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 192–211.

50. A. Al-Bataineh and G. White, "Analysis and Detection of Malicious Data Exfiltration in Web Traffic," in *Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on*. IEEE, 2012, pp. 26–31.
51. J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep Packet Inspection Over Encrypted Traffic," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 213–226.
52. J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 124–134.
53. Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee, "Beheading Hydras: Performing Effective Botnet Takedowns," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 121–132.
54. D. Bradbury, "Fighting Botnets with Sinkholes," *Network Security*, vol. 2012, no. 8, pp. 12–15, 2012.
55. J. S. Hiller, "Civil Cyberconflict: Microsoft, Cybercrime, and Botnets," *Santa Clara Computer & High Tech. LJ*, vol. 31, p. 163, 2014.
56. F. Brauer, "Compartmental Models in Epidemiology," in *Mathematical Epidemiology*. Springer, 2008, pp. 19–79.
57. D. Dagon, C. C. Zou, and W. Lee, "Modeling Botnet Propagation Using Time Zones." in *NDSS*, vol. 6, 2006, pp. 2–13.
58. M. Khosroshahy, M. K. M. Ali, and D. Qiu, "The SIC Botnet Lifecycle Model: A Step Beyond Traditional Epidemiological Models," *Computer Networks*, vol. 57, no. 2, pp. 404–421, 2013.
59. Z. Lu, W. Wang, and C. Wang, "On the Evolution and Impact of Mobile Botnets in Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2304–2316, 2016.
60. L. Pellis, F. Ball, S. Bansal, K. Eames, T. House, V. Isham, and P. Trapman, "Eight Challenges for Network Epidemic Models," *Epidemics*, vol. 10, pp. 58–62, 2015.
61. S. Lee and J. Kim, "Fluxing Botnet Command and Control Channels with URL Shortening Services," *Computer Communications*, vol. 36, no. 3, pp. 320–332, 2013.
62. K. Xu, P. Butler, S. Saha, and D. Yao, "DNS for Massive-Scale Command and Control," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 3, pp. 143–153, 2013.
63. K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee, "Evaluating Bluetooth as a Medium for Botnet Command and Control," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2010, pp. 61–80.
64. J. Echeverria and S. Zhou, "Discovery of the Twitter Bursty Botnet," *arXiv preprint arXiv:1709.06740*, 2017.
65. S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: A Covert Social Network Botnet," in *International Workshop on Information Hiding*. Springer, 2011, pp. 299–313.
66. Q. Li, C. Larsen, and T. van der Horst, "IPv6 - A Catalyst and an Evasion Tool for Botnets and Malware Distribution Networks," *Computer*, p. 1, 2012.
67. W. Chen, C. Yin, S. Zhou, and X. Yan, "Cloud-Based Mobile Botnets using Multiple Push Servers," in *Parallel Architectures, Algorithms and Programming (PAAP), 2015 Seventh International Symposium on*. IEEE, 2015, pp. 183–189.
68. S. Shafieian, M. Zulkernine, and A. Haque, "CloudZombie: Launching and Detecting Slow-Read Distributed Denial of Service Attacks from the Cloud," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1733–1740.
69. J. Idziorek, M. F. Tannian, and D. Jacobson, "The Insecurity of Cloud Utility Models," *IT Professional*, vol. 15, no. 2, pp. 22–27, 2013.
70. S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao, "ZombieCoin 2.0: Managing Next-Generation Botnets using Bitcoin," *International Journal of Information Security*, pp. 1–12, 2017.
71. X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A Survey on the Security of Blockchain Systems," *Future Generation Computer Systems*, 2017.

72. IETF, "HTTP/2," last accessed 23th December 2017. [Online]. Available: https://http2.github.io/
73. IANA, "DNSSEC information," last accessed 23th December 2017. [Online]. Available: https://www.iana.org/dnssec
74. "Tor Project," last accessed 23th December 2017. [Online]. Available: https://www.torproject.org/index.html.en
75. S. Liu and M. Silverman, "A Practical Guide to Biometric Security Technology," *IT Professional*, vol. 3, no. 1, pp. 27–32, 2001.
76. "DETER Project," last accessed 23th December 2017. [Online]. Available: https://deter-project.org/
77. "Stratosphere IPS Project," last accessed 23th December 2017. [Online]. Available: https://stratosphereips.org/
78. C. Cowan, "Software Security for Open-Source Systems," *IEEE Security & Privacy*, vol. 99, no. 1, pp. 38–45, 2003.