International Journal of Information & Network Security (IJINS)
Vol.3, No.2, April 2014, pp. 116~127
ISSN: 2089-3299

A Survey On Real World Botnets And Detection Mechanisms

Somayeh Soltani[1], Seyed Amin Hosseini Seno[2], Maryam Nezhadkamali[1] and Rahmat Budirato[3]
[1]Computer Emergency Response Team (CERT), Ferdowsi University of Mashhad
[2] Departement of Computer Engineering, Ferdowsi University of Mashhad
[3]College of Computer Science and Information Technology, Al Baha University, Kingdom of Saudi Arabia

Article Info

Article history:

Received Mar 15th, 2014
Accepted Apr 12th, 2014

Keyword:

Botnet
Drive-by download
Domain flux
Fast flux service network
Rootkit

ABSTRACT

Mitigating the destructive effect of botnets is a concern of security scholars. Though various mechanisms are proposed for botnets detection, real world botnets still survive and do their harmful operations. Botnets have developed new evasion techniques and covert communication channels. Knowing the characteristics of real world botnets helps security researchers in developing more robust detection methods. There are some surveys in the literature that study botnet detection methods; however they do not advert to real world botnets a lot. In this paper, we study various aspects of several real world botnets, i.e. Conficker, Kraken, Rustock, Storm, TDL4, Torpig, Waledac, Zeus and P2P Zeus. Architecture, protocol, type of infection, communication interval, attacks and evasion techniques of these botnets are probed in this paper. Moreover, studies on botnets mitigation and detection techniques that based on fast flux service networks, domain flux, and drive-by download and new trends in botnet communication channels are reviewed.

Corresponding Author:

Second Author,
Departement of Computer Engineering,
Ferdowsi University of Mashhad, Iran
hosseini@um.ac.ir

1. INTRODUCTION

Cyber world encounters serious security threats such as viruses, worms, trojans and botnets. Botnet can be introduced as the biggest chance for cybercriminals and the biggest challenge ahead of security researchers. Botnet is a collection of infected machines worldwide which receive commands from their botmaster and do some illegal actions such as Distributed Denial of Service (DDoS), credential stealing, click fraud, spam sending, bank account and credit card theft and downloading other malwares.

The communication channel between bots and botmaster is named Command and Control (C&C). Botnets have developed their C&C channels since their first appearance; while the early botnets were centralized IRC based, the new ones use peer to peer (P2P) architecture. IRC based botnets are brittle, because firewall can avoid them by filtering related ports. Totally, centralized botnets suffer from single point of failure; if the C&C server(s) is blocked by law enforcement, the bot network is useless. The P2P architecture is a true replacement for C&C communication. Aside from communicating to botmaster, bots communicate with each other, transferring received commands and new updates from botmaster among each other.

Botnets use different evasion techniques to hinder detection. Obfuscating or encrypting the binary code and the transferring messages is one of the most utilized approaches in real world botnets. Fluxing the IP addresses and domain names of the C&C servers are two techniques for impeding the shutdown attempts. Hiding the presence of the malicious code in the infected machine using some sort of rootkit techniques is another method of anti detection.

Various real world botnets come into existence in recent years. Having detailed information about them, makes more robust detection mechanisms and vice versa. For example while much of normal communication between bots in P2P Zeus, P2P Conficker, Kraken and Storm is based on UDP [1-8] detection techniques presented in [9, 10] filter out UDP flows to reduce the big volume of captured traffic. In this paper we focus on some of real world botnets and discuss their architectures, types of attacks and evasion techniques respectively. Moreover various existing detection techniques are described in this survey.

The rest of the paper is organized as follows. Section 2 reviews the related works. An overview of real world botnets is explained in section 3. Different attacks done by real world botnets are illustrated in section 4. Section 5 describes various evasion techniques used by botnets to hinder detection and tracing. Detection techniques are described in section 6. Section 7 explains some new trends in C&C communication and section 8 concludes the paper.

2. RELATED WORK

Several surveys on botnets are proposed by researchers in [11-14]. The technical report in [11] presents a comprehensive survey on botnet measurement and detection techniques. Different passive techniques including packet inspection, analysis of flow records, DNS-based approaches, analysis of spam records, analysis of application log files, honeypots and evaluation of anti-virus software feedback are probed. Active detection techniques including sinkholing, infiltration, DNS cache snooping, detecting fast-flux networks, IRC-based botnets detection and P2P botnets detection are b2ETBT1 0 0 sink of 100 random IP addresses for local network search and another array of 100 IP addresses for global Internet are generated. Then TCP and UDP scanners try to connect to these IP addresses on some precomputed ports. When a peer is discovered, a peer session will be established.

Bots communicate with other entities for several reasons. Bots need to communicate to botmaster to receive commands and binary updates and send sensitive data stolen from the infected machines. In P2P networks, bots exchange information with each other. For synchronization purposes, bots may connect to time servers as in Storm botnet [23] or acquire time from some known web sites like google.com as in Conficker C. These communications usually take place in specific time intervals and make some repetitive patterns which can be used by analysts for detection purposes. Conficker A queries each of domains generated by a DGA algorithm every three hours. In Zeus, a message containing main status information about the zombie such as botID, IP address, bot OS is sent to botmaster every two minutes. Moreover another message containing stolen data is sent to botmaster every ten minutes [24]. Torpig bots contact every twenty minutes to their C&C server to upload the stolen data [25]. Bots in P2P Zeus check the responsiveness of their peers every thirty minutes.

Table 1. Botnets Overview

* Check connectivity

Name	Creation (Detection) Date	Architecture	Protocol	Type of infection	Number of Peers	Communication Interval	Number of Zombies
Conficker A	November 2008	Centralized	HTTP [16, 17]	MS08-067 exploit [16, 17]	-	Every 1 minute*, Every 3 hours** [16, 17]	(All variants) 10.5 million till 2009 [26]
Conficker B	December 2008	Centralized	HTTP [16, 17]	MS08-067 exploit, NetBIOS Share, USB [16, 17]	-	Every 1 minute*, Every 2 hours** [16, 17]	-
Conficker B++	February 2009	Centralized	HTTP [16]	MS08-067 exploit, NetBIOS Share, USB [16]	-	-	-
Conficker C	March 2009	P2P	TCP, UDP, HTTP [3, 4]	Updated from previous variants [3, 27]	scans the Internet looking for other peers [3, 4]	-	-
Conficker E	April 2009	P2P	TCP, UDP, HTTP [27]	MS08-067 exploit, NetBIOS Share, USB [27]	-	-	-
Kraken	Late 2006 [28]	Centralized	UDP, TCP [5, 6]	Social Engineering [5]	-	-	400.000 till 2008 [28]
Rustock	Around 2006	P2P [29]	HTTP	Spam email [30]	-	-	1.3 million [31]
Storm	Mid 2006 [32]	P2P [7, 8, 33, 34]	UDP+ Overnet/ eDonkey [7, 8]	Spam email plus social engineering techniques and client-side vulnerabilities [7, 32, 33]	100 [8], 290 [7] ***	every 10 minutes [7]	between 1 million and 5 million [7]
TDL4	2010	P2P	HTTP, Kad network [15]	Drive-by downloads [35], MS10-092 vulnerability [36]	-	-	Over 4.5 million till 2011 [15, 35]
Torpig		Centralized	HTTP [25]	Drive-by downloads [25]	-	Every 20 minutes [25]	180.000 till 2009 [25]
Waledac	April 2008 [19]	P2P [19, 20]	HTTP [19, 20]	Spam emails using social engineering techniques [19-21] , Drive-by downloads [19]	500 [19]	-	90.000 Till 2010 [37]
Zeus	2006 [24]	Centralized [38]	HTTP [38]	Spam emails using social engineering techniques, Drive-by downloads [22]	-	Every 2 minutes and 10 min [24]	3.6 million (US only) Till 2009 [38]
Zeus (P2P)	September 2011 [1]	P2P [1, 2]	UDP/TCP and HTTP [1, 2]	Spam emails [1]	50 [1]	Every 30 minutes [1]	200.000 [39]

** Query the list of domains generated by DGA
*** Number varies based on Strom version

4. REAL WORLD BOTNET ATTACKS

Botmaster spread its malware to different machines worldwide to perform its malicious activities. Table 2 demonstrates some types of attack have been done by various botnets. Distributed denial of service attacks are designed to overwhelm the communication bandwidth and computational resources of the victim server and stopping it from servicing to its normal clients. Storm uses two types of DoS attack, i.e. TCP syn flood and ICMP ping flood and attacks some anti-spam sites and antivirus web sites [7, 8, 33, 34]. Moreover Rustock, TDL4, Waledac and P2P Zeus have the DDoS code in their binaries [1, 2, 19, 35, 40].

Malware infects a machine by using spam emails and social engineering techniques which later on to infect other systems and to disseminate other types of malware. Kraken and Storm are two famous botnet spammers. Storm in one of its peak day is responsible for 99% of all spam emails seen by a large service provider [41]. One single Kraken bot has sent up to 500,000 pieces of spam in a day [28]. Rustock, TDL4, Waledac and Zeus are some other spammer botnets.

Stealing sensitive information from infected machine is one of the common events takes place by bots. Web site login credentials, cookies, credit cards, banking accounts and passwords are some of important information which is thieved by botnets. Botnets use different methods to steal aforementioned information. Torpig steals bank account and credit card information using man-in-the-browser phishing attacks. Torpig configuration file contains about 300 domains of banks and financial institutions. It also steals a variety of other personal Information. Zeus which is the largest bank theft botnet, hooks network APIs to steal network related information.

Other malicious activities are done by different botnets. Zeus botmaster may occasionally ask its bots to send him a snapshot of the system screen [24]. TDL4 does click fraud which increases the number of clicks on an advertisement on a site and has benefits for owner of the site [35, 36, 42]. It also does bitcoin operations [43]. Waledac downloads and installs a ransomware which is a fake antivirus, warning users about some threats in their systems and inciting them to buy the full version of the antivirus [19, 20]. Zeus installs CryptoLocker ransomware which encrypts some files on user systems and prompts user to send a ransom in order to receive the decryption tool [44].

TDL4 is used for malware dissemination; it downloads other malicious programs to the host computer [15, 35, 36]. Moreover TDL4 searches the system for any competitor’s malware and removes it [15, 35]. Though having a lot of zombies, Conficker does not perform any serious attack yet. Only Conficker E includes Waledac which used for spam sending [27].

Table 2. Botnet Attacks

Name	DDoS Attacks	Spam Sending	Password theft
Conficker	-	-	-
Kraken	-	Yes [5, 6, 28]	-
Rustock	Yes [40]	Yes [45]	steal sensitive information [46]
Storm	Yes [7, 8, 33, 34]	Yes [7, 32, 33]	-
TDL4	Yes [35]	Yes [35]	Yes [35]
Torpig	-	-	Yes [25]
Waledac	Yes [19]	Yes [19, 20]	capture login information [19, 21]
Zeus	-	Yes [22]	Steal passwords, credentials and banking information [24]
Zeus (P2P)	Yes [1, 2]	-	steal crypto certificates, steal cookies [1], Steal banking information [2]

5. REAL WORLD BOTNET EVASION TECHNIQUES

Botnets use various techniques to elude tracing. Table 3 shows some of these techniques including IP address flux, domain flux, binary obfuscation or encryption, encrypted or obfuscated communication and rootkit.

IP address fluxing or Fast Flux Service Network (FFSN) is an evasion technique which is used in some of recent botnets such as Waledac and Storm [7, 20, 34]. FFSN maps a single fully qualified domain name to different set of IP addresses. These IP addresses belong to infected systems scattered in the world. Botmaster chooses some of its zombies as front-end servers, setting their IP addresses in the response of DNS queries in A records. One can see these A records by means of Dig tool which is part of BIND name server software [47]. Intercepting the C&C channel by these proxy servers, the real hidden server is protected from shutdown attempts. To achieve a secure scheme, a short Time-to-Live (TTL) is considered for each DNS record and a new set of zombies act as front-end servers.

Domain flux is another evasion technique used wildly in recent botnet such as Kraken, Conficker, Torpig, TDL4 and P2P Zeus [1-3, 6, 25, 42]. Botmaster changes domain name of C&C server periodically in order to elude tracking and shutdown attempts. Bots have a Domain Generation Algorithm (DGA) embedded in their code, which using a seed generates a random domain list. Then bot tries to connect to each of these domains by sending DNS query. Just one (or a few) of these domains which is recently registered by botmaster answers the query with IP address of the domain. Other queries are responded with an NXDomain response.

P2P Zeus generates 1,000 domains per week. Domain generation algorithm first creates a MD5 hash over concatenation of the year, month, day and domain index, which is used to generate domain names. The top-level domain (TLD) is selected from one of the six top-level domains and concatenated to the domain name [1, 2]. Conficker A and B generate 250 domain names per day and use UTC date for seed. Conficker A selects a TLD randomly from five famous TLD and appended to the domain name. The number of TLDs increases to eight in Conficker B [16].

In response to collaboration of Microsoft, ICANN and several security team to disable Conficker's domains [48], Conficker C generates 50,000 random domains per day. The TLD concatenated to each domain is selected randomly from 110 TLDs. However each bot just sends DNS query to 500 of these domains. If none of these 500 domains are registered by botmaster, bot sleeps for 24 hours, and then will generate a new list of 50,000 domains [3]. Kraken uses a random word generator that constructs English-language alike words containing vowels and consonants properly. The generated word is appended with a suffix chosen randomly from a list of common nouns, verbs and adjective and adverb suffixes [49].

To harden the analysis of the binary file, botnets use some kind of obfuscation or encryption. Moreover to enable validating the code, digital signature is appended to it. The binary file in Conficker A is encrypted using the symmetric stream cipher RC4 with a password which is constructed by SHA-1 hash of the binary file. Then this encrypted file is signed by RSA algorithm. The signature is appended to the encrypted binary. Bots after receiving the binary file will validate it to ensure that it is signed by Conficker authors [3, 16]. In Conficker B, authors use a new presented hashing algorithm, i.e. MD-6 instead of SHA-1 [3]. Binary files in P2P Zeus are signed with an RSA-2048 signature of the MD5 hash of the plaintext data and are encrypted with RC4 algorithm using a hardcoded key plus XOR encryption [1].

Encrypting the communication messages hinders detection of botnets. Bots of Torpig periodically contact the C&C server to upload stolen data. A simple obfuscation mechanism using XOR and base64 encoding protects this communication [25]. P2P Zeus has various types of messages, some of which are protected using different encryption and signing techniques. For example, TCP data request is encrypted with RC4 using the identifier of the recipient as the key. Botmaster may choose one bot as proxy by issuing the proxy announcement message which is signed by RSA-2048 [1].

Botnets use rootkit techniques to hide its presence on a system, steal sensitive data from the system and disable security products. User mode API hookings including IAT hooking and inline hooking are the simplest rootkit techniques. IAT hooking takes place by exchanging one (or more) entry of import address table of a binary with the address of malicious code. For example to hide malware related files and directories in Windows Explorer, malware can change the IAT entry of NtQueryDirectoryFile in explorer.exe process and replace it with malicious code which filter out the malware files and directories. Inline hooking takes place by directly changing the API function. For the above example it is sufficient for the malware to patch code of NtQueryDirectoryFile in Ntdll.dll and divert its execution somewhere in the code to malicious code [50, 51].

User mode API hooking though simple, involves injecting a DLL to every process that need the changes and may not work for some API functions. Kernel mode rootkits are more robust but need more precise programming; any bug found in the code yields system crash. One simple kernel mode hooking involves exchanging the SSDT table entries with the malicious code. This hooking needs more programming efforts compared to user mode API hooking. All user modes and kernel modes API hooking can be detected by rootkit revealers like Gmer [52].

Runtime patching the native APIs is another rootkit technique which is detected harder. Though Microsoft protects some data structures like SSDT from changing in 64-bit operating systems using Kernel Patch Protection (KPP) or PatchGuard technique [53], malware can bypass this protection technique. Direct Kernel Object Manipulation (DKOM) is the most sophisticated rootkit method used by some malware. Even if online rootkit revealer like Gmer [52] cannot detect this type of rootkit, offline digital forensic tools like Volatility [54] are powerful enough to detect this rootkit [51].

Real world botnets use some kind of rootkit to hide their processes and resources or disable security products or steal information. Zeus hooks some network related APIs to steal information before sending through the network [24]. Torpig uses a rootkit which replace the system's Master Boot Record (MBR) and executes at boot time [25]. Storm performs kernel mode hooking to hide malware files and drivers [7, 8, 32, 34].

To remain undetected, real world botnets do some self defenses. Conficker Disables security products, Disables AutoUpdate, Blocks DNS lookups related to security applications, deactivates safeboot mode and disables Windows' firewall protection of certain high-order UDP and TCP ports [3, 16, 17]. Storm disables security products and windows file protection. It can detect debuggers and virtual machines and accordingly falls into an infinite loop [7, 34]. TDL4 bypasses PatchGuard and the Windows code integrity mechanism and performs anti debugging checks [36, 55]. P2P Zeus injects its code into other processes' memory, therefore hides its network activities [2].

Table 3. Botnet Evasion Techniques

Botnet Name	Fast Flux Service Network	Domain Flux	Binary Encryption, Obfuscation and signature	Encrypted Communication	Rootkit (API Hooking)
Conficker	No	Yes [3, 16]	RC4, MD6, SHA-1, RSA [3, 16]	Yes [3]	Yes (User Mode API Hooking) [3, 16, 17]
Kraken	No	Yes [6, 49]	Yes [5, 28]	Yes [5]	Yes [5]
Rustock	No	No	No	TLS [56]	Yes (Kernel mode rootkit) [46]
Storm	Yes [7, 34]	No	TEA [34]	-	Kernel rootkit for hiding malware files and drivers [7, 8, 32, 34]
TDL4	Yes [15]	Yes [42]	RC4 [55]	Base62, proprietary encryption using XOR [15]	an MBR based rootkit named Alureon (32bit/64bit rootkit) [15]
Torpig	No	Yes [25]	-	XOR, base64 encoding [25]	an MBR based rootkit named Mebroot [25]
Waledac	Yes [20]	No	Bzip2,AES [19, 20]	Bzip2, AES-128-CBC, Base64, RSA keys [20]	-
Zeus	No	No	XOR, RC4 [57]	XOR, RC4 [24, 38, 57]	User mode API hooking to steal sensitive information [24]
Zeus (P2P)	No	Yes [1, 2]	RSA-2048, RC4, XOR, MD5 [1]	RC4, RSA-2048, MD5, XOR, zlib [1]	-

6. DETECTION MECHANISMS

In this section various techniques for detection of IP and domain flux, drive-by download attacks and presence of bots in a single host or in a monitored network are examined.

6.1 Detecting fast flux service networks

The first study that introduces the FFSN is [58]. Holz et al. [59] probe the matter and propose a metric for detection of FFSN. According to their investigation, about 30% of domains advertised in spam are FFSNs. They compare FFSN with two other techniques, Round Robin DNS (RRDNS) [60] and Content Distribution Network (CDN). RRDNS is a load balancing technique used by large websites which returns a list of A records in response to DNS queries. The response list is cycled in a round-robin manner for each query. CDN is an advance load balancing method which distributes data among different far apart servers. While servers in a CDN scatter around the world, the servers in a RRDNS are located in a same place. These two techniques are similar to FFSN in the number of A records in DNS responses. Also the CDN technique is similar to FFSN because of its low TTLs.

[59] presents some distinguishing parameters to detect FFSNs from benign domains: 1) nA, the number of unique A records returned in all DNS queries. 2) nNS, the number of nameserver (NS) records in one single query. 3) nASN, the number of unique ASNs for all A records. They represent a metric named fluxiness ($\varphi = n_A / n_{single}$) to distinguish FFSNs from CDNs. n_{single} is the number of A records returned in a single DNS query. For benign domains without the usage of CDNs, this metric is equal to one. But for CDNs and FFSNs, the fluxiness metric is greater than one.

[59] also proposes another metric based on above parameters named flux-score using a weight vector w and a bias term b: $f(x) = w^T x = w_1 \cdot n_A + w_2 \cdot n_{ASN} + w_3 \cdot n_{NS}$

A flux-score f(x)>b indicates an instance of a FFSN and f(x)<b correspond to benign domains.

Another study in this field is FluXOR [61], a system to monitor and detect FFSNs. FluXOR extracts nine distinguishing features from domains which categorized in three groups: 1) features relating to domain name, 2) features relating to availability of the networks and 3) features relating to heterogeneity of the agents.

FluXOR is a system with three components: 1) Collectors which collect suspicious hostnames from emails, 2) Monitors which extract some distinguishing features from domains. Once a domain is tagged as FFSN, the monitor tries to find the proxy servers (i.e. botnet zombies) by acting like a zombie and querying the DNS records. 3) A detector which combines the aforementioned features using a naïve Bayesian classifier. The study shows 7.8% of hostnames collected from spam emails monitored by FluXOR for a month were associated with FFSNs.

Flux-score [59] and FluXOR [61] use some temporal metrics such as the number of total A records or the number of autonomous systems for detection of FFSNs. Temporal metrics though accurate, need considerable time (i.e. the TTL of A record) to decide about the domain, which delay the detection. Meanwhile the bot harder dictates many of his malevolent orders to his zombies and uses some other techniques like domain flux to exchange the domain to a new registered one. Researches in [62-64] propose some real time detection schemes.

Spatial Snapshot Fast-Flux Detection (SSFD) [62] is a detection scheme which maps IP addresses in DNS responses to geographical coordinates using hostip.info [65]. SSFD defines two spatial measures, spatial uniform distribution estimation and spatial service relationship evaluation, for detecting FFSNs. Spatial uniform distribution estimation is an entropy-based function which estimates the uniform geographic distribution of proxy servers (i.e. zombies). Because this measure cannot differentiate CDNs from FFSNs, the service relationship measure is presented. The authors of [62] acclaim that the real time SSFD system is more effective and efficient than flux-score based detection system [59].

In [64], a Genetic-based ReAl-time DEtection (GRADE) system is proposed. GRADE defines two new measures, the Entropy of Domains of Preceding Nodes (E-DPNs) and the Standard Deviation of Round Trip Time (SD-RTT) between the analyzer system and all A record host. Analyzing the results of the traceroute command, GRADE estimates the heterogeneity of the domains of the preceding nodes of all A record hosts. The preceding domains of A records associated with CDNs and RRDNSs will exhibit a high degree of homogeneity. But the preceding domain of A records relating to FFSNs will show considerable heterogeneity. For benign domains using CDN technique, the SD-RTT metric is small. But for FFSNs because of the scatter model of proxy servers, the SD-RTT is a large value.

GRADE uses four metrics nA, nASN, E-DPN and SD-RTT and plugs these metrics into a linear decision function. To determine the best set of weights for the decision function, a genetic algorithm is employed. Experimental results show that GRADE classifies domains in a few second and with high accuracy. The authors of [64] claim that GRADE achieves higher detection accuracy compared to flux-score [59], FFBD [63] and SSFD [62].

6.2 Detecting domain flux

One straightforward solution to detect domains generated algorithmically by a botnet, is to reverse engineer the malware executable which is resource and time intensive and is not always feasible [49, 66]. Antonakakis et al. [66] present a detection system called Pleiades based on the number of NXDomain responses created in querying the DGA-generated domains. They test their system in a large ISP network and cluster NXDomains based on their similar syntactic features and the number of overlapping compromised machines that query them. Then (if possible) they assign these clusters to models of known botnet DGAs.

Malicious automatically generated domain names are detected in [67] using Stateful-SBB. A dataset of benign and malicious domain names is used in training and test phase. Different classification algorithms including Naïve-Bayes, C4.5, AdaBoost, SBB and Stateful-SBB are applied to identify malicious domain names. Since classifiers other than Stateful-SBB require the data set to be feature-based, 17 different features are extracted for each domain name. The Stateful-SBB classifier and C4.5 classifier provide high accuracy on classification. Though Stateful-SBB classifies domains without requiring any a priori knowledge, however other classifiers require a set of features.

6.3 Detecting drive-by download attacks

Drive-By Download (DBD) attack is a growing type of attack which takes place when a user visits a web page containing malicious codes. The Malicious code is silently downloaded to the user system and executed without the user consent. Often a drive-by download attack is done in three phases. 1) Shellcode injection phase: that takes place through a malicious active client-side content (such as JavaScript) within a malicious site or a legal site which is exploited by an attacker, 2) Shellcode execution phase: which uses a vulnerability in the browser or its plug-in components to detour the control flow to the shellcode and execute it and 3) Malware download and install: which silently downloads the malware (bot) code from remote server, store it in the system and execute it on the host operating system.

Researches in [68-72] have probed this type of infection and propose solutions to mitigate it. BLADE [68] for example is an attack-agnostic system that prevents drive-by malware installation. Placing

BLADE as a dynamic loadable driver into the OS can successfully prevent the execution of binaries which are downloaded without user consent.

BLADE consists of five components. Screen Parser monitors kernel windowing events to detect appearance of a download consent dialog (a dialog box for asking permission from user) and notify the Supervisor. To track user interaction with this dialog box (clicking Yes or No button), the Hardware-Event Tracer is invoked by the Supervisor. Hardware-Event Tracer intercepts mouse and keyboard input events to detect those that relate to the download consent dialog. Correlator matches a downloaded file to a tuple (*u*,*p*) where *u* is the URL from which file is downloaded and *p* is the file system path where file is saved. The I/O Redirector establishes a secure zone, a location where all binaries downloaded by the browser and its child processes are placed. Execution of files in this area is prohibited by blocking memory-section synchronizations. Files that are downloaded by the user consent, subsequently moved out of the secure zone.

Authors of [68] have evaluated the proposed scheme for some common browsers against some active malicious sites. BLADE successfully detects and blocks all DBD infections with zero false positives.

Egele et.al. [70] propose a DBD detection scheme which is based on detection of shellcode in JavaScript strings. Binary representation of shellcode is typically assigned to JavaScript string variables in the address space of the browser. To facilitate the execution of shellcode, attacker put multiple instances of the shellcode combined by a NOP sledge in different strings. All strings that are allocated by the JavaScript interpreter are monitored for presence of shellcode using the libemu library [73]. Starting from each character of the string, libemu checks whether there is a sequence of valid instructions in the string. These checks are done before a vulnerability can be abused to divert control flow to the shellcode. If the system finds such a string, the corresponding script is terminated.

The proposed scheme has been implemented by extending Mozilla Firfox and its JavaScript engine, SpiderMonkey [74]. The implemented scheme can detect drive-by downloads that exploit memory corruption vulnerabilities and use JavaScript code for launching the exploit with zero false positive. However, the scheme cannot detect and protect other types of DBDs.

In [69], an anomaly-based approach for detection of drive-by download attacks which use malicious JavaScript code is presented. First some distinguishing features in a DBD attack life cycle are defined. Using these features and some machine-learning techniques, the system can identify anomalous JavaScript code. The features are related to different phases of an attack, namely redirection and cloaking, deobfuscation, environment preparation and exploitation.

Number and target of redirection, browser personality, ratio of string definitions and uses, number of dynamic code executions, length of dynamically evaluated code, number of bytes allocated through string operations, number of likely shellcode strings, number of instantiated plugins, values of parameters in method calls and sequences of method calls are ten features proposed by [69]. The implemented tool called JSAND is available online in [75] that analyzes and gives detailed reports for URLs or files sent by any user. However, the proposed scheme does not prevent or block any attacks.

6.4 Botnet detection

In this section several botnet detection techniques are presented. We study recent works on botnet detection that are not discussed in previous surveys [11, 12].

First, we discuss a work from Huang [76]. This work presents a host-based detection system based on network failure model. The author assumes that network failures is an inherent property of botnet traffic which results from unavailability of C&C server, a peer or the attacked target. Huang’s work shows that network failure patterns of normal, peer-to-peer and botnet traffic are distinguishable and classifies them using the C4.5 decision tree classifier. The system workflow consists of 1) collecting numerous benign, peer-to-peer and bot traces, 2) filtering out non-failure traces, 3) extracting features from failure flows and 4) building the classifier. To filter out non-failure traffic, this work defines possible type of failures, most of which relate to transport protocols, TCP and UDP. For example receiving a TCP reset (RST) or an ICMP unreachable in response to a TCP SYN, is considered as failure. In the feature extraction phase, this work presents thirty four features of failure flows which can be classified into six categories: certain types of failures, average interval between failures, total number of failures, ratio of distinct destination port numbers, ratio of distinct destination IP addresses and average number of failures per destination IP address. The author believes that the proposed model can detect bot hosts with more than 99% accuracy.

The second work is a botnet detection method based on group activities of bots as presented in [23]. The work is an improvement of the authors' previous works [77, 78]. The detection scheme named BotGAD works by capturing group activities from DNS traffic. Though Botnet group activities are more shown in centralized botnets, some P2P botnets may have the same activities. DNS traffic while having a small portion of network traffic, enables botnet detection at its early stages, maybe prior to performing attacks.

Botnet group activities include C&C server or update server lookup as well as victim lookup. In centralized model, bots frequently send some messages to C&C server. For example, HTTP based bots periodically send HTTP requests to server to receive commands from it. To resolve the IP address(es) of the server domain, generally DNS queries are sent. When performing malicious behavior like DDoS attacks, spam distribution and click frauds, DNS queries are sent to find the IP addresses of victims. While group activities are seen in normal communication, they have some distinguishable characteristics in botnets, which are used in [23] for detection mechanism. Pattern Based Network Security Using Semisupervised Learning that intelligently is able to detect botnet activities in a network is introduced in [79]. A Botnet Prevention Strategies for Social Network users is discussed in [80].

Lastly, EFFORT [81] is a host-network cooperated framework for detection of bot process on a system and employs five modules. The first module, Human-process-network correlation, finds a suspicious process which does not have keyboard or mouse events and heavily creates DNS traffics. If a process tagged as suspicious by this module, it will be sent to other modules for further analysis. The second module, Process reputation analysis, determines the reputation of a process mostly by detecting the reputation of the entities (servers/peers) whom the process contact. Reputation information of a domain is collected by 1) detecting some anomaly features in its registration information, 2) investigating its previous records in well-known blacklists and 3) asking a search engine like google. The third module, system resource exposure analysis, monitors resource access activities of a suspicious process. While bots may modify critical registry keys, create a large number of sockets in a short time, access and modify in system folders; these actions are not seen in benign processes. The fourth module, network information trading analysis, determines the information gain/loss for suspicious process. Unlike normal processes which act as clients in network communication and gather information, bots mostly send data (stolen data, massive spam and DDoS packets) to other entity. The fifth module, correlation engine, determines the weights of the decision of each module and makes the final decision whether the process is bot or not.

7. NEW TRENDS IN C&C COMMUNICATION

A new trend in botnet design is to hide the communication channel between bots and botmaster. Stegobot [82] for example is a social network botnet which uses covert channel for communication. First infection of Stegobot is through social-malware attacks. Emails seem to be sent from friends of victim (social phishing) having the malicious code embedded in the attachment or persuading social links which refer to sites having malicious code are some form of social-malware propagations. Once the malware is deployed on a machine, it can be propagated by embedding the malicious payload in any email attachment sent by user.

Communication between bots and botmaster is through social network image sharing behavior of members. Stegobot uses image steganography to hide the presence of communication. The botmaster commands and the bots stolen information are embedded in the images share by the user of the system. Just viewing the image by other members of the social network, transfers the image to their system. If the system is infected by bot, the embedded information is extracted and subsequent operations are done. It is sufficient for botmaster to embed its command in an image. It will be scattered through the social network overlay and reached to its bots.

VoIP may be used as covert C&C channel between bots and botmaster [83] as in MoshiMoshi botnet [84]. Voice network is used to transfer commands. In this way, bots can be controlled individually or as a group. Since voice traffic is not monitored a lot, detection of VoIP botnet is unlikely. Different from IRC-based botnets which can be prevented by blocking related ports, phone calls cannot be avoided.

Using URL Shortening Services (USSes) for hiding C&C channel is proposed in [85]. USSes such as bit.ly [86] and is.gd [87] give a shortened URL or alias of a long URL. Botmaster can stealthy communicate with the bots using aliases generated by USSes. Alias flux, a new technique proposed by Lee et.al. [85], is based on changing the aliases associated with IP addresses of C&C servers. Botmaster periodically register obfuscated IP addresses of C&C servers to USSes. Obfuscated IP address is needed to make it similar to a legitimate URL and to hide the IP address of C&C from USSes. In the case of USSes which support custom aliases (aliases selected by user), an alias generation algorithm is used by both botmaster and bots. Giving the date information to this algorithm, n different aliases is generated which will be registered by botmaster into USSes. Bots that are wanting to communicate with C&C servers, use this algorithm to retrieve custom aliases and then query one of these aliases from USSes. In the case of not supporting custom aliases, alias generation algorithm is replaced whit a list of registered aliases.

The botnet threat may spread to mobile smart phones and generate Mobile botnet phenomenon. Hua et.al. [88] probe this matter and claim that human mobility facilitates the propagation of commands. Two C&C communication based on SMS flooding algorithm and Bluetooth technology is presented. Bot transfers commands by sending SMSs to its neighbor bots sporadically. To achieve stealth, group messaging is avoided. Whenever smart phones are close enough, they transfer commands using Bluetooth.

8. CONCLUSION

Today cyber world encounters various botnets that try to perform malicious activities without the knowledge of bot system owners. Knowing the various aspects of these botnets helps users in detecting abnormal behavior of their systems. Moreover if security researchers who develop botnet detection mechanism be aware of real world botnet characteristics, they can offer more accurate products. In this paper important aspects of some real world botnets are probed.

Botnets use different methods to infect systems worldwide. Different attack vectors used by botnets are explained. Detecting drive-by download attacks, one of the popular attack vectors is deliberated in this work. Various evasion techniques used by botnets to delay and hinder shutdown attempts are described. Besides, some solutions to detect these evasion techniques are reviewed. New technologies used by botnets in transferring commands are introduced which may attract researchers in the field.

REFERENCES

[1] Andriesse D, Bos H. "An Analysis of the Zeus Peer-to-Peer Protocol". Report. 2013.

[2] "ZeuS-P2P monitoring and analysis". Report. CERT Polska, 2013.

[3] Porras P, Saidi H, Yegneswaran V. "Conficker C Analysis". Report. SRI International, 2009.

[4] Porras P, Saidi H, Yegneswaran V. "Conficker C P2P Protocol and Implementation". Report. SRI International, 2009.

[5] Mushtaq A. "Kraken Botnet – A detailed analysis" 2008. Available from: http://www.fireeye.com/blog/technical/botnet-activities-research/2008/04/kraken-botnet-1.html.

[6] Royal P. "Analysis of the Kraken Botnet": Damballa; 2009. Available from: https://www.damballa.com/downloads/d_pubs/KrakenWhitepaper.pdf.

[7] Porras P, Saidi H, Yegneswaran V. "A Multi-perspective Analysis of the Storm (Peacomm) Worm". Report. SIR International, 2007.

[8] Stewart J. "Storm Worm DDoS Attack" 2007. Available from: http://web.archive.org/web/20071013133954/http://secureworks.com/research/threats/view.html?threat=storm-worm.

[9] Chang S, Daniels TE, editors. "P2P Botnet Detection using Behavior Clustering & Statistical Tests". Proceedings of the 2nd ACM workshop on Security and artificial intelligence (AISec '09); 2009.

[10] Yu X, Dong X, Yu G, Qin Y, Yue D, Zhao Y. "Online Botnet Detection Based on Incremental Discrete Fourier Transform". JOURNAL OF NETWORKS. 2010;5.

[11] Plohmann D, Gerhards-Padilla E, Leder F. "Botnets: Detection, Measurement, Disinfection & Defence". Report. The European Network and Information Security Agency (ENISA), 2011.

[12] Silva SSC, Silva RMP, Pinto RCG, Salles RM. "Botnets: A survey". Computer Networks. 2013;57:378–403.

[13] Liu J, Xiao Y, Ghaboosi K, Deng H, Zhang J, editors. "Botnet: classification, attacks, detection, tracing, and preventive measures". Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC'09); 2009.

[14] Zhang L, Yu S, Wu D, Watters P, editors. "A Survey on Latest Botnet Attack and Defense". Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM '11); 2011.

[15] Golovanov S, Soumenkov I. "TDL4 – Top Bot" 2011. Available from: http://www.securelist.com/en/analysis/204792180/TDL4_Top_Bot.

[16] Porras P, Saidi H, Yegneswaran V. "An Analysis of Conficker's Logic and Rendezvous Points". Report. SRI International, 2009.

[17] Porras P, Saidi H, Yegneswaran V, editors. "A Foray into Conficker's Logic and Rendezvous Points". Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET'09); 2009.

[18] "MS08-067: Vulnerability in Server service could allow remote code execution". Available from: http://support.microsoft.com/kb/958644.

[19] Tenebro G. "W32.Waledac Threat Analysis": Symantec. Available from: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/W32_Waledac.pdf.

[20] Baltazar J, Costoya J, Flores R. "Infiltrating Waledac Botnet's Covert Operations: Effective Social Engineering, Encrypted Http2p Communications, And Fast-Fluxing Networks". Report. Trend Micro.

[21] Pilici S. "Remove TrojanDownloader: Win32/Waledac.AK (Virus Removal Guide)" 2013. Available from: http://malwaretips.com/blogs/trojan-downloader-win32-waledac-ak-removal/.

[22] Nahorney B, Falliere N. "Trojan.Zbot": Symantec. Available from: http://www.symantec.com/security_response/writeup.jsp?docid=2010-011016-3514-99.

[23] Choi H, Lee H. "Identifying botnets by capturing group activities in DNS traffic". Computer Networks. 2012;56:20-33.

[24] Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer R, et al., editors. "Your botnet is my botnet: analysis of a botnet takeover". Proceedings of the 16th ACM conference on Computer and communications security (CCS '09); 2009.
[25] Riccardi M, Pietro RD, Palanques M, Vila JA. "Titans’ revenge: Detecting Zeus via its own flaws". Computer Networks. 2013;57:422-35.
[26] "Calculating the Size of the Downadup Outbreak" 2009. Available from: http://www.f-secure.com/weblog/archives/00001584.html.
[27] "Conficker Working Group: Lessons Learned" 2010. Available from: http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf.
[28] Higgins KJ. "New Massive Botnet Twice the Size of Storm" 2008. Available from: http://www.darkreading.com/attacks-breaches/new-massive-botnet-twice-the-size-of-storm/d/d-id/1129410?
[29] "Defeating Rustock In the Courts". Available from: http://www.microsoft.com/security/sir/story/default.aspx#!rustock_defeating.
[30] SPAMfighter News. "New Rustock Botnet Trying to Expand Itself" 2008. Available from: http://www.spamfighter.com/News-10711-New-Rustock-Botnet-Trying-to-Expand-Itself.htm.
[31] Greene J. "Microsoft hands Rustock botnet case over to FBI" 2011. Available from: http://www.cnet.com/news/microsoft-hands-rustock-botnet-case-over-to-fbi/.
[32] Landesman M. "Storm Botnet". Available from: http://antivirus.about.com/od/virusdescriptions/p/stormbotnet.htm.
[33] Holz T, Steiner M, Dahl F, Biersack E, Freiling F, editors. "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on StormWorm". Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats Article No 9 (LEET'08); 2008.
[34] Boldewin F. "Peacomm.C - Cracking the nutshell.zip" 2007. Available from: http://www.reconstructer.org/.
[35] Rouse M. "TDL-4 (TDSS or Alureon)" 2011. Available from: http://searchsecurity.techtarget.com/definition/TDL-4-TDSS-or-Alureon.
[36] ESET Team. "TDSS part 1: The x64 Dollar Question" 2011. Available from: http://resources.infosecinstitute.com/tdss4-part-1/.
[37] Goodin D. "Waledac botnet 'decimated' by MS takedown" 2010. Available from: http://www.theregister.co.uk/2010/03/16/waledac_takedown_success/.
[38] Binsalleeh H, Ormerod T, Boukhtouta A, Sinha P, Youssef A, Debbabi M, et al. "On the Analysis of the Zeus Botnet Crimeware Toolkit". Eighth Annual International Conference on Privacy Security and Trust (PST)2010.
[39] Rossow C, Andriesse D, Werner T, editors. "SoK: P2PWNED — Modeling and Evaluating the Resilience of Peer-to-Peer Botnets". In Proceedings of the 34th IEEE Symposium on Security and Privacy, IEEE S&P 2013; 2013.
[40] Stewart J. "Rustock DDoS Attack". Available from: http://www.joestewart.org/rustock-ddos.html.
[41] Gaudin S. "Storm Worm Erupts Into Worst Virus Attack In 2 Years" 2007. Available from: http://www.informationweek.com/storm-worm-erupts-into-worst-virus-attack-in-2-years/d/d-id/1057418?
[42] Antonakakis M, Demar J, Stevens K, Dagon D. "Unveiling the Network Criminal Infrastructure of TDSS/TDL4 DGAv14: A case study on a new TDSS/TDL4 variant". Available from: https://www.damballa.com/downloads/r_pubs/Damballa_tdss_tdl4_case_study_public.pdf.
[43] Paz RD. "TDL4 Worm Component Employs Bitcoin Mining" 2011. Available from: http://blog.trendmicro.com/trendlabs-security-intelligence/the-worm-tdl4-and-botcoin-miners/.
[44] Alintanahin K. "CryptoLocker: Its Spam and ZeuS/ZBOT Connection" 2013. Available from: http://blog.trendmicro.com/trendlabs-security-intelligence/cryptolocker-its-spam-and-zeuszbot-connection/.
[45] "Biggest spammer? The Rustock botnet" 2008. Available from: http://www.securityinfowatch.com/press_release/10549446/biggest-spammer-the-rustock-botnet.
[46] "Backdoor.Rustock". Available from: http://www.symantec.com/security_response/writeup.jsp?docid=2006-011309-5412-99&tabid=2.
[47] "BIND The most widely used Name Server Software". Available from: https://www.isc.org/downloads/bind/.
[48] "Microsoft Collaborates With Industry to Disrupt Conficker Worm" 2009. Available from: http://www.icann.org/en/news/announcements/announcement-2-12feb09-en.htm.
[49] Yadav S, Reddy AKK, Reddy ALN, Ranjan S, editors. "Detecting Algorithimically Generated Malicious Domain Names". Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10); 2010.
[50] Blunden B. "The Rootkit Arsenal Escape and Evasion in the Dark Corners of the System": Wordware Publishing, Inc.; 2009.
[51] Ligh MH, Adair S, Hartstein B, Richard M. "Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code": Wiley Publishing, Inc.; 2011.
[52] "GMER- Rootkit Detector and Remover". Available from: http://www.gmer.net/.
[53] "Kernel patch protection: frequently asked questions" 2007. Available from: http://msdn.microsoft.com/en-us/library/windows/hardware/dn613955%28v=vs.85%29.aspx.
[54] "Volatility an advanced memory forensics framework". Available from: https://code.google.com/p/volatility/.
[55] Rusakov V. "TDSS. TDL-4" 2011. Available from: https://www.securelist.com/en/analysis/204792157/TDSS_TDL_4.
[56] "Beware Botnet's Return, Security Firms Warn". Available from: http://www.pcworld.com/article/192668/beware_botnets_return_security_firms_warn.html.
[57] Wyke J. "What is Zeus?". Report. Sophos, 2011.

[58] Salusky W, Danford R. "Know Your Enemy: Fast-Flux Service Networks" 2007. Available from: http://www.honeynet.org/papers/ff.
[59] Holz T, Gorecki C, Rieck K, Freiling FC, editors. "Measuring and Detecting Fast-Flux Service Networks". Proceeding of the 15th Annual Network & Distributed System Security Symposium (NDSS'08); 2008.
[60] Brisco T. "RFC 1794: DNS support for load balancing" 1995. Available from: http://www.ietf.org/rfc/rfc1794.txt.
[61] Passerini E, Paleari R, Martignoni L, Bruschi D, editors. "FluXOR: detecting and monitoring fast-flux service networks". Proceedings of the 5th Conference on Detection of Instrusions and Malware & Vulnerability Assessment (DIMVA'08); 2008.
[62] Huang S-Y, Mao C-H, Lee H-M, editors. "Fast-flux Service Network Detection Based on Spatial Snapshot Mechanism for Delay-free Detection". Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10).
[63] Hsu C-H, Huang C-Y, Chen K-T, editors. "Fast-flux bot detection in real time". Proceedings of the 13th international conference on Recent Advances in Intrusion Detection (RAID'10); 2010.
[64] Lin H-T, Lin Y-Y, Chiang J-W. "Genetic-based Real-time Fast-Flux Service Networks Detection". Computer Networks. 2013;57:501-13.
[65] "HOSTIP Project. My IP Address Lookup and GeoTargeting Community Geotarget IP Project – what country, city IP addresses map to". Available from: http://www.hostip.info/.
[66] Antonakakis M, Perdisci R, Nadji Y, editors. "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware". Proceedings of the 21st USENIX conference on Security symposium (Security'12); 2012.
[67] Haddadi F, Kayacik HG, Zincir-Heywood AN, Heywood MI, editors. "Malicious Automatically Generated Domain Name Detection Using Stateful-SBB". Proceedings of the 16th European conference on Applications of Evolutionary Computation (EvoApplications'13); 2013.
[68] Lu L, Yegneswaran V, Porras P, Lee W, editors. "BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections". Proceedings of the 17th ACM conference on Computer and communications security (CCS'10); 2010.
[69] Cova M, Kruegel C, Vigna G, editors. "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code". Proceedings of the 19th international conference on World wide web (WWW'10); 2010.
[70] Egele M, Wurzinger P, Kruegel C, Kirda E, editors. "Defending Browsers against Drive-by Downloads: Mitigating Heap-spraying Code Injection Attacks". Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'09); 2009.
[71] Zhang J, Seifert C, Stokes JW, Lee W, editors. "ARROW: GenerAting SignatuRes to Detect DRive-By DOWnloads". Proceedings of the 20th international conference on World wide web (WWW '11); 2011.
[72] Xu K, Yao D, Ma Q, Crowell A. "Detecting Infection Onset With Behavior-Based Policies". 5th International Conference on Network and System Security (NSS '11)2011. p. 57-64
[73] "libemu – x86 Shellcode Emulation". Available from: http://libemu.carnivore.it/.
[74] "SpiderMonkey". Available from: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey.
[75] "Wepawet". Available from: http://wepawet.cs.ucsb.edu/.
[76] Huang C-Y. "Effective bot host detection based on network failure models". Computer Networks. 2013;57:514–25.
[77] Choi H, Lee H, Lee H, Kim H, editors. "Botnet Detection by monitoring group activities in dns traffic". Proceedings of the IEEE International Conference on Computer and Information Technology (CIT '07); 2007.
[78] Choi H, Lee H, Kim H, editors. "BotGAD: detecting botnets by capturing group activities in network traffic". Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE '09); 2009.
[79] Vinod K Pachghare, Vaibhav K Khatavkar, Parag A Kulkarni, Pattern Based Network Security Using Semisupervised Learning, International Journal of Information and Network Security (IJINS),Vol 1 No 3, 2012:228-234.
[80] Nishikant C Dhande, Botnet Prevention Strategies for Social Network users: Cases and Remedies International Journal of Informatics and Communication Technology (IJ-ICT), Vol 2 No 1, 2013:46-50.
[81] Shin S, Xu Z, Gu G. "EFFORT: A new host–network cooperated framework for efficient and effective bot malware detection". Computer Networks. 2013;57:2628–42.
[82] Nagaraja S, Houmansadr A, Piyawongwisal P, Singh V, editors. "Stegobot: a covert social network botnet". Proceedings of the 13th international conference on Information hiding (IH '11); 2011.
[83] Grant NM, Shaw JI. "Chapter 8 VoIP Bots ". "Unified Communications Forensics": Syngress.
[84] "moshimoshi Open-source VoIP Bot". Available from: http://code.google.com/p/moshimoshi/.
[85] Lee S, Kim J. "Fluxing botnet command and control channels with URL shortening services". Computer Communications. 2013;36:320–32.
[86] "bitly". Available from: https://bitly.com/.
[87] "is.gd". Available from: http://is.gd/.
[88] Hua J, Sakurai K. "Botnet command and control based on Short Message Service and human mobility". Computer Networks. 2013;57:579–97.