Pertanika J. Sci. & Technol. 26 (1): 37 - 70 (2018)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Review Article

Android Botnets: A Serious Threat to Android Devices

Shahid Anwar[1]*, Mohamad Fadli Zolkipli[1], Zakira Inayat[2,3], Julius Odili[1], Mushtaq Ali[1] and Jasni Mohamad Zain[4]

[1]Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, 26300 UMP, Gambang, Malaysia
[2]Department of Computer Science, University of Engineering and Technology Peshawar, Peshawar 2500, Pakistan
[3]Center for Mobile Cloud Computing Research, University of Malaya, 50603 UM, Kuala Lumpur, Malaysia
[4]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 UiTM, Shah Alam, Selangor, Malaysia

ABSTRACT

Android devices have gained a lot of attention in the last few decades due to several reasons including ease of use, effectiveness, availability and games, among others. To take advantage of Android devices, mobile users have begun installing an increasingly substantial number of Android applications on their devices. Rapid growth in many Android devices and applications has led to security and privacy issues. It has, for instance, opened the way for malicious applications to be installed on the Android devices while downloading different applications for different purposes. This has caused malicious applications to execute illegal operations on the devices that result in malfunction outputs. Android botnets are one of these malfunctions. This paper presents Android botnets in various aspects including their security, architecture, infection vectors and techniques. This paper also evaluates Android botnets by categorising them according to behaviour. Furthermore, it investigates the Android botnets with respect to Android device threats. Finally, we investigate different Android botnet detection techniques in depth with respect to the existing solutions deployed to mitigate Android botnets.

Keywords: Android botnets, malware, detection techniques, DDoS attacks, mobile security

ARTICLE INFO

Article history:
Received: 06 December 2016
Accepted: 27 September 2017

E-mail addresses:
shahidanwar.safi@gmail.com (Shahid Anwar),
fadli@ump.edu.my (Mohamad Fadli Zolkipli),
zakirainayat@uetpeshawar.edu.pk (Zakira Inayat),
odili_julest@yahoo.com (Julius Odili),
sunnygul1@gmail.com (Mushtaq Ali),
jasni@tmsk.uitm.edu.my (Jasni Mohamad Zain)
*Corresponding Author

ISSN: 0128-7680

INTRODUCTION

Mobile devices (mobile devices/android devices are used interchangeably in this article) are gaining popularity in the 21st century (Narudin, Feizollah, Anuar, & Gani,

2016). These devices offer a host of advanced capabilities and ample storage of large volumes of personal and confidential data (Peng, Yu, & Yang, 2014). Nowadays, most mobile devices offer more computing capabilities and memory storage than many personal computers did a few years back. According to former Android boss Andy Rubin, "There should be nothing that users can access on their desktop that they cannot access on their cellphone" (Rubin, 2008). Any mobile device has three core features, such as Applications, Storage and Connectivity. These key features make Android devices an attractive tool for malware writers to attack organsation/ individual devices. While protecting stored data on these devices is crucial against today's threats, most mobile devices use the Android operating system due to its open nature. Android provides a full set of software for Android devices including operating system, middleware and key Android applications (Sears, 2007). The open nature of Android devices make these devices an attractive source for cybercriminals and over the years there has been a number of threats faced by mobile devices such as spyware, botnet, vulnerable applications, privacy threats, drive-by-download, phishing scams, malware, network exploits, browser exploits and Wi-Fi sniffing (Fossi et al., 2011; Inayat, Gani, Anuar, Khan, & Anwar, 2016), Botnet is one of the most dangerous threats faced by mobile devices recently.

Malware is used to damage Internet-connected devices and gather sensitive information from individuals or it uses spyware for accessing the most private information on the infected device (Sharma, Chawla, & Gajrani, 2016). Spyware gathers all this information specifically for advertising purposes (Sheta, Zaki, El Salam, & Hadad, 2015). Privacy threats can be caused by those Android applications that may not be malicious by nature but use sensitive information obtained illegally from unsuspecting Android users. Vulnerable applications are those that contain deficiencies that may cause malicious attacks and malicious activities. Phishing scams are those that use the victim's device emails for sending the virus infected links to the Internet-connected devices (Naraine, 2012). In drive-by-download, the infected devices download an application when they access a website. While the browser-exploits benefits from the vulnerabilities in mobile device web browsers or applications launched by the browser such as flash player, PDF reader and much more, in network exploits, cybercriminals take advantage of Android operating system flaws for criminal activities (Naser, Zolkipli, Majid, & Anwar, 2014). When the data are transferred from one device to another connected by Wi-Fi as many applications do not use proper security rules, this results in data obstruction known as Wi-Fi sniffing. In this article, we focus on the Android botnet.

A botnet (Robot Network) is a type of malware that enables the infected devices to perform criminal activities according to the botmaster's instructions (Anwar, Zain, Inayat, Haq, Karim, & Jaber, 2016; Naser et al., 2014; Peng et al., 2014). A malicious Android application is installed in a susceptible host that is capable of carrying out a series of different harmful activities to the end user according to the botmaster's instructions. These applications can be downloaded to the victims' devices using different methods. The most common ways to infect a victim's device includes access to the infected websites, drive-by-download, spam emails, viral mechanism and much more (Anwar, Zain, Zolkipli, Inayat, Khan, Anthony, & Chang, 2017; Karim, Shah, Salleh, Arif, Md Noor, & Shamshirband, 2015). Once an end-user's device is infected with malicious software, it receives instructions from the cybercriminal (botmaster) through a command and control server using communication channels. Botmaster is the entity

that performs criminal activities from these bot devices, while a communication channel is the way through which a botmaster can communicate with the C&C server and bots. A bot can be a servant and a client as well at the same time. It can propagate themselves to infect vulnerable hosts (Silva, Silva, Pinto, & Salles, 2013).

To the best of our knowledge, this paper aims to present the Android botnet from first appearance. We aim to guide interested readers and researchers on Android botnets and detection techniques. This paper organises the Android botnet detection techniques with respect to their benefits and limitations; understanding this information can improve Android botnet detection techniques.

The key contributions of this survey paper are:

- It provides up-to-date information on mobile device threats: We provide comprehensive details of the possible threats to mobile devices. We have also categorised these threats in sub-groups according to their nature.
- It provides exhaustive information about Android botnets: We provide in-depth information about Android botnets, their background and timeline.
- We provide in-depth information about Android botnet detection techniques: This paper presents detailed information about Android botnet detection techniques. We explain these techniques regarding their benefits and limitations. These limitations are also explained in more detail in table form.
- We introduce future research challenges: We suggest potential research areas for Android botnet detection techniques and we highlight the challenges present in Android botnet detection techniques as well.

Classification of Mobile Device Threats

There are diverse types of threat to mobile devices that may badly affect mobile devices, such as viruses and spyware that can infect personal computers (PC). These threats can be divided into four broad categories: Application-Level, Web-Level, Network-Level and Physical-Level (see Figure 1).

Application-Level Threats

Application-level threats are based on the Applications, which are the core feature of every mobile device. These threats appear to be the most widely discussed threats in the literature, which presents application-level threats as the most widely discussed threat. Since the applications that run on these mobile devices are available from third-party markets, it is clear that they can be target vectors for mobile device security breaches (Faruki et al., 2015). Malware are Android applications that perform malicious activities can inject malicious codes into mobile device that send unsolicited messages and allow an adversary the ability to remotely control the device.

Malware. Malware is short for ‘malicious software’. This is specifically developed to damage machines on which they are executed or the network on which it communicates (Inayat, Gani,

Anuar, Anwar, & Khurram Khan, 2017; Preda, Christodorescu, Jha, & Debray, 2008). Malware is mostly installed on victims' devices to perform illegal activities without the knowledge of the owner. The range of malware varies; it can be as simple as pop-up advertising or so dangerous that it causes machine invasion or damage. Stealing owner-sensitive credentials and infecting new vulnerable devices are the main targets of malware. The most common malware is found as financial, crypto locker and advertisement malware (Anwar, Zain, Zolkipli, Inayat, Jabir, & Odili, 2015).

Financial malware is developed for scanning mobile devices to gather financial information, while crypto locker malware is used in cyber-criminal activities. According to the Symantec report published in 2013, ransomware evolves regularly in the Android operating system. Compared with other OS on mobile devices, Android is most frequently attacked because of its open nature (Narudin et al., 2016; Odili, Kahar, & Anwar, 2015; Teufl et al., 2013). Ransomware allows cybercriminals to hijack the victim's device, encrypt the victim's private files and then demand a ransom from the victim in order for the files to be released (Anwar et al., 2017). Malicious spyware is considered a significant threat to the confidentiality of mobile devices (Sheta, Zaki, El Salam, & Hadad, 2015). It covertly collects confidential data from the infected device and sends it to the cybercriminal (Botmaster) through the user's Internet connection without the owner's knowledge. These applications mostly contain freeware or shareware, which can be downloaded from third-party markets. Adware is another type of malware. It is a software package that automatically displays related advertisements to the mobile device users based on the user's pattern of web surfing. These advertisements may be present during the installing phase of any Android application, or they are present when an end-user is using these applications. This adware engages in collecting confidential information, frequently by user consent, while stealing this personal information for covert harmful activities.

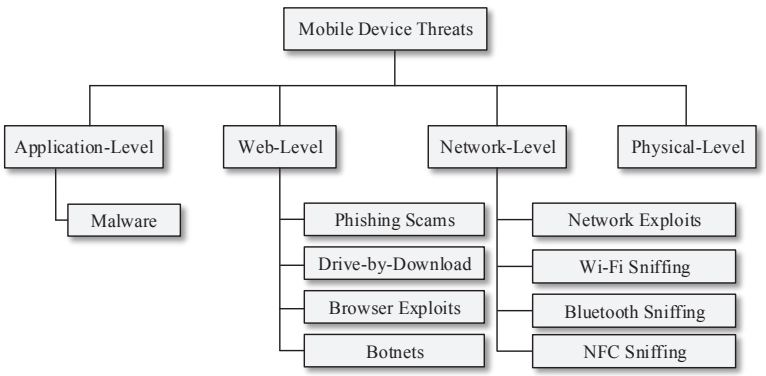

Figure 1. Threats to mobile device security

Web-Level Threats

The security and privacy threats to mobile devices from webs happens normally. The most dangerous web-level threats are phishing scams, drive-by-download and browser exploits. Phishing scams are the key web-level threat, which uses email or other social media apps

to send an unwitting user links to a phishing website designed to trick users into providing sensitive information such as user credentials. Phishing is one of the top seven security threats identified by the Kaspersky lab (Kaspersky, 2015). However, botware is the most dangerous threat to mobile devices nowadays. These are software programmes created to automatically perform specific operations.

Phishing scams. Phishing refers to the criminal action of generating a replica of web pages that exist to fool a mobile user entering private, extremely sensitive credentials, financial or online banking information and passwords (Alta, Loock, & Dabrowski, 2005). Phishing is a technique of attacking to obtain personal information from a mobile device user and is the main cause of various problems encountered by Internet users. This technique can cost the victim financially. Phishing is performed through instant messenger phishing, voice phishing, and flash phishing (Dunne, 2006; Milletary & Center, 2005). After launching a phishing attack on an individual or an organisation, the employees of the organisation handle the customer when he calls after losing his money.

Drive-by-Download. A drive-by-download refers to potentially harmful software code that is installed on a person's computer without the user's permission; the user may not even be aware that the software has been installed. Drive-by-downloads are a form of malware typically found on compromised web pages. By simply 'driving by', or visiting the web page, the drive-by-download begins to download and is then installed in the background on the computer or mobile device without alerting the user (Naraine, 2012).

Browser exploits. This is a malicious code that uses a piece of software or operating vulnerabilities to breach the security of the browser. Browser exploits perform these malicious activities without informing the owner of the device.

Botnets. Short for robot network, botnet, is the network of Internet-connected infected-devices (bots) under the control of a botmaster (cybercriminal) to perform cyber-criminal activities without the knowledge of the device owner (Anwar, Mohamad Zain, Zolkipli, & Inayat, 2014). There are two types of botnet: traditional botnets and mobile botnets. This paper focusses on mobile (Android) botnets. The purpose of Android botnets will most likely be similar to those of existing traditional botnets (e.g. providing means of DoS, DDoS and spam distribution); however, the targets are different (Enck, Ongtang, & McDaniel, 2009). In mobile botnets, the targets are mobile devices.

A common botnet having thousands of infected victims is called a bot (zombie). The botmaster sends instructions to all online bots to send queries to a particular system/server (Mirkovic & Reiher, 2004). By attacking a new victim from thousands of different bots in a botnet, the DoS (DDoS) is distributed. In a DDoS attack, the bot becomes harder to detect and it is difficult for cyber law enforcement to prevent DDoS attacks. Some DDoS attacks include UDP flood attacks, Zero-day DDoS attacks, Sync flood attacks, ICMP flood attacks, Slowloris and Ping of Death (LulzSec, 2011; Zang, Tangpong, Kesidis, & Miller, 2011). DDoS attacks are performed using diverse types of tool, such as agent- and IRC-based tools. These

attacks can be detected through screening of the time interval of requests and bandwidth size. Some DDoS attacks, such as Zero-day attacks, are unknown or new and thus, have no patch yet. The term DDoS is well-known among hackers as dealing with Zero-day vulnerabilities is a common activity.

Network-Level Threats

Any mobile device has three core features: applications, storage and connectivity. Network-level threats can occur due to mobile device connectivity with the cellular/mobile networks, local wireless networks or near field-communication (NFC). Network exploits, Wi-Fi sniffing, Bluetooth and NFC are the main types of network-level threats.

Network exploits. Network exploits take advantage of flaws in the mobile operating system or other software that operates on local or cellular networks, such as an International Mobile Subscriber Identity (IMSI) catcher. Once connected, they can intercept data connections and find a way to inject malicious software on users' phones without their knowledge.

Wi-Fi sniffing. Wi-Fi sniffing seizes data when they are traveling between the device and the Wi-Fi access point. Most Android applications do not use proper security measures while sending unencrypted data across the network. A cybercriminal can easily read the data as they travel. Public sites such as coffee shops, restaurants and bookstores may have WPA2, but it is likely that anyone with the password can decrypt your packets.

Bluetooth. People who leave BT on all the time leave themselves vulnerable to pairing from nefarious devices and the uploading of spyware. Blue jacking is an older-style attack in which a Bluetooth enabled device that is active is used by someone else. Blue jacking refers to the sending of unsolicited data (vCards etc.) to open Bluetooth listeners in the area. It has more recently been used for marketing, but many more modern smartphones are less vulnerable to Bluetooth stack exploits. This can lead to phishing attempts and the spread of malware or viruses.

Near field communication (NFC). Advanced mobile devices contain near field communication (NFC) as a medium for communication. NFC is a newly developed wireless technology that provides communication between two mobile devices, both of which must contain NFC tags using short-range radio waves. NFC enables the exchange of images, apps and other data between two devices without first pairing them. For this purpose, both devices use a feature that Google calls Android Beam (Sauter, 2013), while Beam is Android's trademark for NFC when the protocol is used for device-to-device communication. In the NFC communication, only two devices can communicate, such as the initiator and target. The initiator sends data, while the target receives them; both devices are active during the communication, consuming their own battery power. NFC provides extra opportunities to the attacker to compromise NFC-enabled devices, such as Wi-Fi and Bluetooth. So far, mobile threats are still mainly aimed at consumers rather than at enterprises.

Physical-Level Threats

Physical-level threats are more important than other mentioned threats. Since mobile devices are small, portable and valuable, it makes their physical security more important. Stealing and misplacing devices are the common issue among users of these devices. These devices are valuable not only because they are resold in the black market, but more importantly, because they contain sensitive organisational and personal data. Most mobile device users use their phone for banking, social communication and much more, while end-user are always connected to these accounts, which makes the stolen or misplaced phone more vulnerable for criminal activities. Furthermore, a lost or stolen device can be used to gain access to secret data stored on it.

Android Botnets as a Universal Threat for Mobile Device Users

The dramatic increment in the number of mobile device users has attracted cybercriminals to develop malicious applications (Narudin et al., 2016). In addition, mobile devices contain more sensitive information about the owners; this tends to be taken lightly by security organisations and individuals. A mobile device can be misused in many ways. The botnet is one of the most successful methods by which a mobile device can be misused for malicious activities against organisations or individuals.

Android Botnet as a Threat for Organisations

Android botnets are mostly used for organised economic fraud. Today, world economies must deal with a broad range of botnets that have caused a considerable amount of damage. About USD7.1 million was estimated lost due to click fraud performed by botnets using DDoS attacks in 2007 (Plohmann, Gerhards-Padilla, & Leder, 2011). It is very hard to detect click fraud, as these target legitimate users while they are surfing websites. According to a published report, on every USD3 million spent on digital advertisement, USD1million is spent on click fraud. Another statistics report showed that digital advertising hit the highest level of fraud in 2015, which was estimated at USD27.5 billion (Slefo, 2015). According to Kaspersky's monitoring results, 35,000 malicious mobile programmess were found at the end of 201. These malicious programmes steal sensitive data from the end-user devices, consume account balances while running digital advertisements, which is pushed by these malicious programmes (Christian & Maria, 2013). Kaspersky released security threat statistics for the year 2015, in which they blocked 0.8 billion attacks and used a list of 6.5 million unique host, to from web resources located in various countries around the world (LAB, 2015).

Android Botnet as a Threat for Individuals

Mobile devices offer advanced capabilities, with more storage capacity that can store organisational and confidential data of end users (Peng et al., 2014). Furthermore, advanced mobile devices offer more computing capabilities than many of personal computers offered a few years back. A mobile device has three main features, such as Android applications, storage, and connectivity with internet or cellular network (Rubin, 2008). Once a mobile device becomes

part of the Android botnet, botmasters attempt to take complete control of it to enable the botnet creator to perform illegal activities without the knowledge of the end user. After taking control of the mobile device, a botmaster can access everything from the compromised device. Furthermore, sending texts and making phone calls to premium numbers can be performed with these compromised devices. Botmaster can access contacts and messages on the compromised devices as well. These botnets take advantage of unpatched Android application updates.

Android Botnet Security

As smartphones are the largest category of Android devices, they have become an essential tool in how people communicate with one another. Each Android device has three key features: Applications, online and storage. Android applications are one of the core features of Android devices. They enable users to play games, read the news, connect with others, check weather conditions, perform online banking, read maps and navigators and perform many other functions. These applications are available from third parties like Google Play Store and Amazon (Silva et al., 2013). It may be a primary feature for many end users.

The core function of the Android device is to enable the user to make calls, take photographs, send text or picture messages and access personal data storage. It also allows the developer to develop richer applications. The developer may also access the user's address book, SMS content, GPS location data, movement data by G-sensor and accelerometer and even information in other applications. The Android does not differentiate between the phone's core applications and third-party applications. However, such applications from third parties can access personal/confidential information in the Android device very easily. These core features of Android devices make these devices an easy target for cybercriminals.

In this modern era, trojanised Android applications are a common infection method of Android devices. This is most often targeted by cybercriminals who use different types of malware. Botnet, a dangerous malware, compromises Android devices such as smartphones, smartware, tablets and notebooks, attempting to get full access to the device and provide control to the botmaster. The data found on Android devices include text messages (SMS/MMS), contacts, call logs, e-mail messages (Gmail, Yahoo), chats, location coordinates using the global positioning system (GPS), photographs, videos, web history, search history, driving directions, Facebook and Twitter information, music collections and other information. These third-party applications provide a simple and easy means of accessing content and services of Android devices. It is important to be aware of how to use these third-party applications safely and securely. Android botnets are able to spread themselves by sending copies to compromised devices.

The criminal activity the Trojan-Ransom.Android-OS.Small family is a multifunctional ransomware Trojan performed by an Android botnet. After connecting to the botnet army, it receives commands from the command and control channel and performs the activities received from them. Once run, it asks for the victim's device's admin rights and loads information about the victim's device to a malicious server. It can be an international mobile equipment identity (IMEI), international mobile subscriber identity (IMSI), device model, brand and phone number or other information.

In addition, the Trojan is registered in the Google cloud messaging (GCM) system. As such, the Trojan can receive commands from both the C&C server and via GCM. With this information, it can perform the commands shown in Table 1.

Table 1
Commands between C&C server and trojan

Command	Description
START	Start the main service of the Trojan
STOP	Stop the main service of the Trojan
RESTART	Restart the main service of the Trojan
URL	Change the C&C address
MESSAGE	Send an SMS to a specified number with a specified text
UPDATE_PATTERNS	Update the rules for processing incoming SMS
UNBLOCK	Disable the device administrator's rights
UPDATE	Download a file from the specified URL and instal it
CONTACTS	Send out a specified SMS to all contacts from the list of contacts
LOCKER_UPDATE	Update the text with the ransom demand
LOCKER_BLOCK	Block the device
LOCKER_UNBLOCK	Unblock the device
CHANGE_GCM_ID	Change the GCM id

The main idea behind botnets is to control interaction in Internet Relay Chat (IRC) chat rooms. They are able to interpret simple commands, provide administration support, offer simple games and other services to chat users and retrieve information about operating systems, logins, email addresses and aliases, in addition to other information (Silva et al., 2013). The first known iKee.B Mobile botnet was found in 2009. It was discovered to be using the Command and Control Server in the iPhone. This botnet is able to propagate itself and to instal third-party applications on the end-user's phone without user information (Peng et al., 2014).

Table 2 shows the timeline of Android botnets with respect to their first appearance in terms of year, platform, instruction, categories and C&C type in addition to other related information.

Table 2
Android Botnet timeline

Year	Name	C&C Type	Botnet Instructions	Criminal Activities by Default	Requires Permission
2010	SMSHowU.A	SMS	Leak location, GPS and maps through SMS	None	N/A
2011	Geinimi.A	HTTP	ON, OFF, ADD or Set or Rem Sender	IMEI, IMSI, SIM, SIM state, Build info, GPS, Board, Brand, CPU type, User, Software version, SIM country, SIM operator	N/A
	DroidKungFu.A	HTTP	Leak location, GPS and maps through SMS	Send sensitive data, execDelete, Exploit known vulnerabilities to gain root, Instal APK, execOpenUrl, execStartApp	N/A

Table 2 *(continue)*

2012	Fjcon.A	HTT Phone	ICCID;	Financial, Propagation of malware	N/A
	Rootsmart	HTTP	action.host start; action. boot; action.shutdown; action.install; action. installed; action.check live;action.download apk;	IMEI, IMSI, cell ID, location area code, mobile network code	N/A
	TigerBot.A	SMS	Change APN; Notify of SIM change; Kill running process	IMEI	N/A
2013	Stealer.B	HTTP and SMS	HTTP: time; sms; send; delete; smscf SMS: ServerKey +001; +002; anything	IMEI, IMSI, contacts	READ_SMS; INTERNET; RECEIVE_BOOT_ COMPLETED; READ_ PHONE_STATE; RECEIVE_SMS; READ_CONTACTS; SEND_SMS; WRITE_EXTERNAL_ STORAGE
	Tascudap.A	HTTP	time; sms; send; delete;smscfSMS: ServerKey + 001; 002; anything	Specify time when trojan should next contact C&C, send SMS, delete SMS from phone, selective SMS hiding, start application, forward received SMS, update	READ_SMS; ACCESS_NETWORK INTERNET; READ_PHONE_STATE; RECEIVE_SMS; READ_CONTACTS; SEND_SMS; WRITE_EXTERNAL_ STORAGA;
	BadNews.A	HTTP	news; showpage; install; showinstall; iconpage; coninstall; newdomen; seconddomen; stop; testpost	Propagation of possible malware; download and instal APK	RECEIVE_BOOT_ COMPLETED; SEND_ SMS; RECEIVE_SMS; INTERNET; ACCEESS_INTERNAL_ MEMORY; ACCESS_EXTENAL_ MEMORY;
	Spamsold.A	SMS	Display same icon on the menu, retain the image same but the name may change, instal APK once clicked	Sends SMS spam messages without the user's consent	INTERNET; CHANGE_ COMPONENT_ ENABLED; RECEIVE_SMS; READ_SMS; SEND_SMS WRITE_SMS; RECEIVE_SMS; RAISED_THREAD_ PRIORITY; READ_ CONTACTS; WRITE_EXTERNAL; RECEIVE_BOOT_ COMPLETED; WAKE_ LOCK;

Table 2 *(continue)*

2014	FrictSpy.E3	HTTP; SMS	Command and Control to execute malware activities such as call records, use camera for pictures and videos, use mic for recording voice	Incoming/Outgoing call; Incoming/Outgoing SMS, GPS location information, URLs that the device user accesses	ACCESS_NETWORK_ STATE; CALL_PHONE; GET_TASKS; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_BOOT_ COMPLETED; RECEIVE_SMS; SEND_SMS; SYSTEM_ALERT_ WINDOW; WAKE_LOCK; WRITE_SMS;
	Geinimi.A	HTTP	ON, OFF, ADD or Set or Rem Sender	User, Software version, IMEI, SIM State, CPU type, SIM country, IMSI, SIM, SIM operator, build info, GPS, Board, Brand	CALL_PHONE; GET_TASKS; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_BOOT_ COMPLETED; RECEIVE_SMS; SEND_SMS; SYSTEM_ALERT_ WINDOW; WAKE_LOCK; WRITE_SMS;
	SpyBubb.A	SMS	Leak location, GPS and maps through SMS; HTTP: time; sms; send; delete; smscf SMS: ServerKey +001; +002; anything	Collect SMS, Call, Fine location, Coarse location, GPS, Device info like IMEI, IMSI etc. Share phone information to vendor site	ACCESS_NETWORK_ STATE; ACCESS_WIFI_STATE; READ_PHONE_STATE; INTERNET; WAKE_LOCK;
2015	Leech.A	HTTP	action.host start; action. boot; action.shutdown; action.install; action. installed; action.check live;action.download apk	Instal itself persistently, run with full privileges, unwanted payment through SMS, spying activities, dynamically load command and control server	ACCESS_NETWORK_ STATE; ACCESS_WIFI_STATE; READ_PHONE_STATE; INTERNET; WAKE_LOCK;
	Tediss	SMS	N/A	Monitor calls, SMS and conversation applications	CALL_PHONE; GET_TASKS; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_BOOT_ COMPLETED; RECEIVE_SMS; SEND_SMS; SYSTEM_ALERT_ WINDOW; WAKE_LOCK; WRITE_SMS;
	WormHole.A	HTTP and SMS	WormHole.A HTTP and SMS	Instal applications without notification; Location information; Add contact items; Monitor list of applications	READ_EXTERNAL_ STORAGE; READ_PHONE_STATE; READ_NETWORK_ STATE; INTERNET; READ_INTERNAL_ STORAGE; WAKE_LOCK; READ_COARS_ LOCATION;

Table 2 *(continue)*

	SilverPush.A	HTTP and SMS	HTTP: time; sms; send; delete; smscf SMS: ServerKey +001; +002; anything; ON, OFF, ADD or Set or Rem Sender	IMEI number; Operating system version; Location; Potentially the identity of the owner; Behaviour of users using TVs; Web browsers; Radios	ACCESS_NETWORK_ STATE; CALL_PHONE; GET_TASKS INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_SMS; SEND_SMS; WRITE_SMS;
2016	MazarBOT.A	SMS	N/A	Sends premium SMS, exfiltrate sensitive information and steal the received SMS messages by setting up a backdoor on device	ACCESS_NETWORK_ STATE; CALL_PHONE; GET_TASKS; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_BOOT_ COMPLETED; RECEIVE_SMS; SEND_SMS; SYSTEM_ALERT_ WINDOW; WAKE_LOCK; WRITE_SMS;
	Morder.A	HTTP and SMS	Command and Control to execute Malware activities such as calls record, use camera for pictures and videos, use mic for recording voice	Track location; Leak contacts to C&C Upload data from SD Card to C&C; Delete or download files in the infected device; Leak phone call history; Take pictures with the camera; Record audio and calls; Execute shell commands	ACCESS_NETWORK_ STATE; CALL_PHONE; GET_TASKS; ACCESS_FINE_ LOCATION; ACCESS_COARS_ LOCATION; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_BOOT_ COMPLETED; RECEIVE_SMS; SEND_SMS; SYSTEM_ALERT_ WINDOW; WAKE_LOCK; WRITE_SMS;
	Smishing.D	SMS	time; sms; send; delete; smscf SMS: ServerKey +001; +002; anything; ON, OFF, ADD or Set or Rem Sender	Detect text messages; Access fraudulent fake bank URL; Steal user's sensitive credential; Password stealing; Additional information stealing	ACCESS_NETWORK_ STATE; CALL_PHONE; GET_TASKS; INTERNET; READ_PHONE_STATE; READ_SMS; RECEIVE_SMS; SEND_SMS; WRITE_SMS;

NA=Not Available, HTTP=Hyper-Text Transfer Protocol, SMS=Short Message Service, PUP=Potential Unwanted Programmes, SD=Secure Digital, C&C=Command & Control Servers, IMEI=International Mobile Equipment Identity, IMSI=International Mobile Subscriber Identity, HTTP=Hyper-Text Transfer Protocol

Components of Android Botnet

A typical Android botnet has four elementary components as shown in Figure 2: bot, botmaster, command and control server and communication channel.

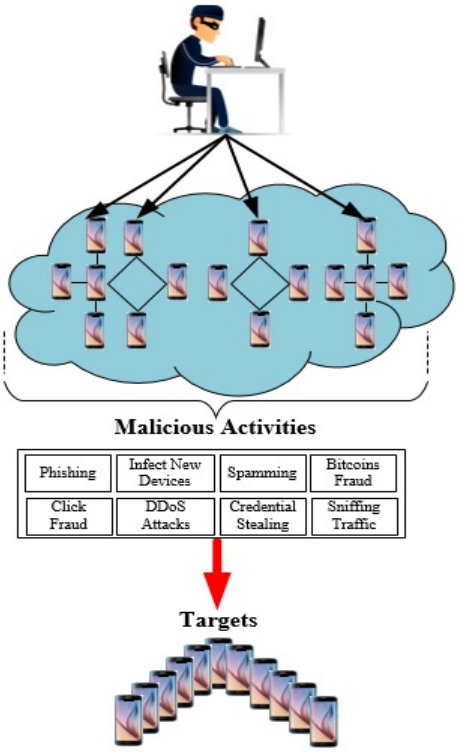

Figure 2. Typical android botnet structure

Bot. A bot is a malicious Android application that is installed in a susceptible host that can perform a series of different harmful actions upon the end user at a cybercriminal's command. This application can be installed to the victim devices in diverse ways. The most common ways include access to the infected websites, drive-by-downloads, spam emails, viral mechanisms and much more (Karim et al., 2015). Once an end-user device is infected with malicious software, it receives commands and controls from the botmaster through the command and control server using communication channels. A bot can be a servant and client at the same time.

Botmaster. The attacker is also known as the botmaster, who maintains and operates the command and control of botnets from remote areas. The botmaster, also known as the bot-herder, is responsible for a variety of malicious activities. Botmasters ensure that errors are fixed and that the bot does not break any of the rules of the channel or server it is logged into. Most botmasters hide their identity via proxies, the onion ring (TOR) and/or shells to disguise their ip address from detection by investigators and law enforcement agents.

Command control server. The term 'command and control' (C&C) is a military concept. Command and Control servers allow a bot entity to take new instructions and malicious capabilities as commanded by a remote individual (botmaster). These servers are used to control botnets, in particular. Command and control in the botnet's Fast-flux Domain Name System (DNS) can be used to make track the control server difficult to do, as the server may change from day to day. These servers may also hop from one DNS domain to another. The Domain Generation Algorithm (DGA) is used presently to create new DNS names for controller servers. A botnet may have different C&C server topologies like Star, Multi-Server, Hierarchical and Random topology.

Communication channels. The botnet communication channel refers to the protocol used by bots and the botmaster to communicate with each other. Bluetooth, Internet Relay Chat (IRC), Hypertext Transfer Protocol (HTTP), peer-to-peer (P2P) and voice over internet protocol (VoIP) servers are used to pass information between bots and the botmaster. The botmaster creates IRC channels on the C&C server, after which the compromised machines will wait for commands to perform malicious activities. An interesting feature of the IRC protocol is the possibility of multicast communication through groups. The IRC channel has some serious limitations like being easy to detect and interrupt. It is rarely used in corporate networks and is usually blocked (Silva et al., 2013).

Due to these limitations of the IRC channel, the HTTP has become the most usable mechanism for implementing command and control communication (Liu, Chen, Yan, & Zhang, 2008). The first Android botnet named SymbOS/Yxes, which appeared in 2009, (Suarez Tangil, Tapiador, Peris-Lopez, & Ribagorda, 2014) targeted the SYMBIAN OS platform using a rudimentary HTTP-based command and control (C&C) channel. Centralised botnets are not more secure as discussed above, so the trend shifted to decentralised botnets. Most of the decentralised botnets are based on a variety of P2P protocols (Jelasity & Bilicki, 2009). Similarly, VoIP is used as the communication channel in vishing (VoIP and phishing) instead of the more usual email technique.

Life Cycle of Android Botnet

Android botnets can come in different structures and sizes, but in general, they go through the same steps as computer botnets (Silva et al., 2013), as shown in Figure 3. An active botnet requires the bot device to complete its life cycle. A typical Android botnet can be developed and maintained in five phases: initial infection, secondary injection, connection, malicious command and control, update and maintenance. In the first phase, initial infection, the end-user device is infected and becomes an active member of the Android botnet. The second phase, secondary injecting, can be carried out by injecting the malicious code into the end-user devices through Bluetooth, drive-by-download, automatic scan, NFC and Wi-Fi (Faruki et al., 2015).

After injecting the code into the victim's device, the bot finds a way to connect to the command and control server. This happens in the connection phase, which is the only phase that may occur several times during the botnet's life cycle (Liu et al., 2008). Once an infected device connects to the Android botnet's command and control server, the botmaster will send

commands through the C&C server using communication channels, while the bots await commands from the botmaster.

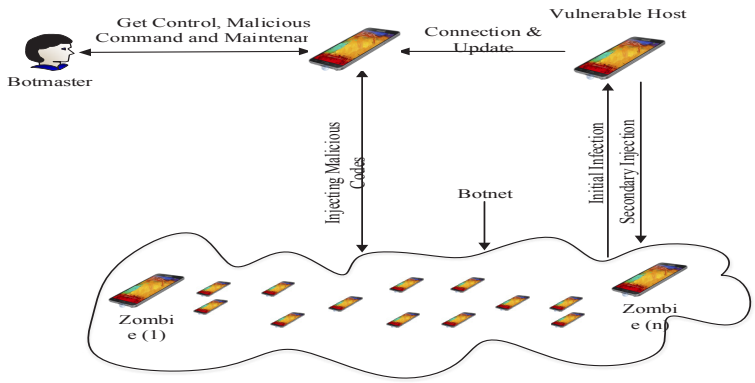

Figure 3. Android botnet life cycle

The last phase of the Android botnet's life cycle is updating and maintenance of the infected devices. Maintenance is important for a botnet to keep his army of infected devices active. The new updates are then sent to these bots many times for many reasons that propagate the different types of criminal activity such as spamming, identity theft, DDoS attacks and much more.

Mobile Device Infection Vectors

There are multiple infection vectors for delivering malicious content to mobile devices. In this survey, we classify infection vectors into four categories: SMS/MMS, Bluetooth, Internet access and file duplication with USB. Cellular services, such as short message service (SMS) and multimedia messaging service (MMS), can be used as attack vectors for smartphones, as shown in Figure 4. For example, SMS/MMS messages can be used to deliver malicious content and to maintain communication with an attacker. For example, ComWar is a worm that browses the host's phonebook and then spreads via SMS/MMS messages (Peng et al., 2014).

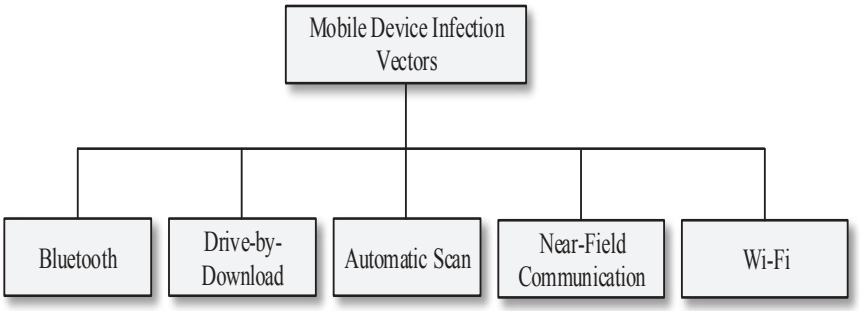

Figure 4. Mobile device infection vectors

Bluetooth

Bluetooth is short-range radio communication protocol used for exchanging data over a limited distance between Bluetooth-enabled devices. Device-to-Device (D2D) malware attacks are performed on the bases of Bluetooth. Once the cybercriminal infects a smartphone with the bot code, it can enable Bluetooth without the knowledge of the owner and target another Bluetooth-enabled device in its range. If the connection is established, the infected device sends the bot code to the targeted device using this Bluetooth vector. This limits the attack vector in some way, such as one-to-one connection, limited distance range and others.

Drive-by-Download

To secure Android devices from botnet attacks, users should only visit reputable websites for downloading application software and other video/audio materials. Botnet infection is possibly acquired by visiting a malicious website. When visited by a smartphone or tablet user, the malicious website forces the user to download the plugin software, which is in fact a malware. If a web page causes the automatic downloading and installation of software without the Android device user's consent, the page is considered malicious. This mechanism, which is also called drive-by-download, allows malware to control Android devices.

Automatic Scan

Automatic scan is performed to infect new victim's devices by compromising and influencing them to be a part of the botnet. In this technique, a new host inside the botnet must be recruited to establish a new botnet through vulnerability scanning (Ianelli & Hackworth, 2005). This goal can be achieved by infecting many hosts, which attempt to identify exploitable vulnerabilities in other new hosts. For example, FTP services suffer buffer overflow exploitation (Lashkari, Ghalebandi, & Moradhaseli, 2011).

Near-Field Communication

Near-Field Communication (NFC) is an advanced wireless technology that allows fast data transfer between two close devices with an enabled NFC setting. NFC is related to mobile payments, such that it has the personal banking information of a user. It has gained popularity among botmasters for spreading malicious commands to compromise other devices because of its fast data transfer capability. In addition, dependence on NFC has induced the C&C channel of botnets to be more challenging (Stevanovic, Revsbech, Pedersen, Sharp, & Jensen, 2012).

Wi-Fi

Wi-Fi has assured compensation over other communication media applicable to Android botnets. The use of open Wi-Fi networks for an Android botnet provides a higher level of stealth and fewer entry barriers than other communication media. Denial of service (DoS) attacks and distributed DoS attacks are threats that can simultaneously inflict devastation on many users. Apart from the aforementioned-infection vectors, smartphones could be compromised

using other methods e.g. the use of USB. If the files used to synchronise smartphones are compromised, malware can also infect smartphones. As a result, attackers can access the host's classified information and instal malicious applications on the smartphone.

Android Botnet Architecture

Personal-computer-based botnet are considered the most compromised platforms for botnet attacks compared to the recently evolved Android botnets due to some limitations, such as limited battery power, limited processing speed, limited internet access and limited memory storage. Android botnets have similar architecture as computer botnets, namely, centralised architecture, decentralised architecture and hybrid architecture. Table 3 shows the advantages and disadvantages of existing Android botnet architecture with respect to map complexity, detection, message latency and survivability.

Table 3
Command-and-Control architecture

Architecture	Centralised	Decentralised	Hybrid
Alias	Star	Peer-to-peer	Random
Map Complexity	Very low	Medium-High	Moderate
Detection	Medium	Low-Medium	High
Message Latency	Very low	Moderate	Moderate-High
Survivability	Low-Medium	Medium	High

Centralised botnet architecture. In centralised botnet architecture, all the bots relate to a central command-and-control server to establish a communication channel with central point as illustrated in Figure 5. In centralised architecture, the botmaster controls and supervises all bots in a botnet from a single C&C server. Botmasters are able to communicate with the bots continuously by sending instructions to them through these central servers (Anwar et al., 2014). As all bots receive commands and report to a C&C server, it is easy for botmasters to manage botnets using centralised architecture. Furthermore, centralised botnet architecture uses two types of topologies, star topology and hierarchical topology, and two types of protocols, Internet Relay Chat (IRC) and Hypertext Transfer Protocol (HTTP) (Khattak, Ramay, Khan, Syed, & Khayam, 2014; Li, Jiang, & Zou, 2009). The design of centralised architecture is less complex compared to other architecture, while message latency and survivability rate are low. This causes low reaction time, easy means of communication and direct feedback (Plohmann et al., 2011).

It also possesses some limitations. For instance, centralised architecture has more maximum failure chances compared with other architecture. If the C&C server fails, then all the botnets may stop working because of the central point of control. Detection of a botmaster is easier compared than if the decentralised and hybrid architecture were used (Bailey, Cooke, Jahanian, Xu, & Karir, 2009; Cooke, Jahanian, & McPherson, 2005; Zang et al., 2011).

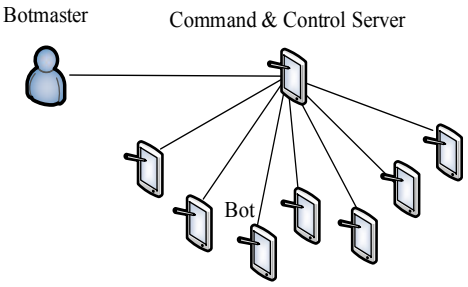

Figure 5. Centralised android botnet architecture

Decentralised botnet architecture. In decentralised botnet architecture, no single responsible entity controls different bots in a botnet. More than one C&C server communicate with various bots as described in Figure 3. Botnets using decentralised architecture are known as decentralised botnets. However, the term peer-to-peer botnet is also commonly used for this type of botnet. Decentralised botnets are more difficult to detect compared with centralised botnets. Figure 6 shows that no specific C&C server exists in decentralised architecture, and all bots act as the C&C server and the client at the same time (Dong, Wu, He, Huang, & Wu, 2008). Decentralised architecture is based on Peer-to-Peer (P2P) protocols. Compared with centralised architecture, the design of P2P architecture is more complex and detection of a botnet with the same architecture is more difficult than detection of other botnets. Message latency and survivability rate are higher than those of the centralised botnet architecture. Failure chances are lower in decentralised architecture than in centralised architecture because if a C&C server fails, then other C&C servers can manage and monitor the botnet (Cooke et al., 2005).

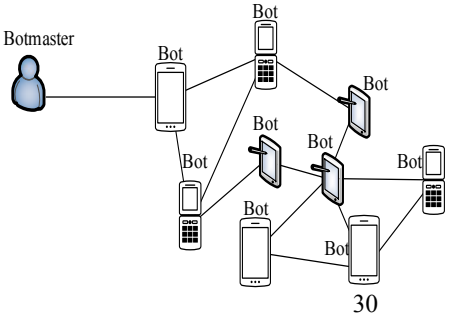

Figure 6. Decentralised architecture

Hybrid botnet architecture. Hybrid architecture is the combination of centralised and decentralised architecture as shown in Figure 7. Hybrid architecture comprises two types of bots, namely, the servant and the client. Bots are connected to the hybrid botnet as a client or a servant. Monitoring and detection of botnets with hybrid architecture is more difficult than detecting those with centralised and decentralised architecture. However, hybrid architecture is less complex in design.

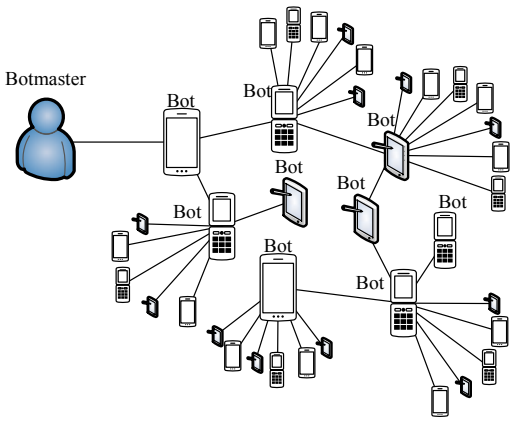

Figure 7. Hybrid architecture

Mobile Botnet Detection Techniques

The challenges faced in mobile-device security are quite similar to those faced in personal-computer security. To solve this problem, researchers have proposed and developed common desktop security solutions for smartphones. Some of the popular security solutions are listed below.

Kirin. Kirin security service is an OS-level protection service that provides enhanced security mechanisms for Android smartphone applications (Enck et al., 2009). This approach performs lightweight certification of applications to mitigate malware at installation time with modification of the Android applications installer. Kirin has different security rules; a well-known combination of permissions is the most important part in these rules. To define these security rules, a detailed understanding of malware and protection techniques is required; it is usually performed by security experts. Furthermore, it prevents access to sensitive information. However, once information enters the application, no additional mediation occurs.

Multi-Agent system. Szymczyk (2009) proposed the Multi-Agent Bot Detection System (MABDS) based on the hybrid approach. It is the combination of multiple agents such as administrative agent, user agent, a central knowledge database, system analysis, honeypots, agent collections and network analysis (Silva et al., 2013). In this technique, each agent observes traffic using different sensors by implementing the Markov chain model to perform dynamic risk assessment (Shameli, Cheriet, & Hamou-Lhadj, 2014). These systems in multifaceted, piercing, real-time domains involve autonomous agents that should act as a team to compete against malware (Castiglione, De Prisco, De Santis, Fiore, & Palmieri, 2014). The slow convergence of new signatures with the knowledge database is the key limitation of this technique. The new signature updates are another limitation of this system (Karim et al., 2014).

SAINT. SAINT (Ongtang, McLaughlin, Enck, & McDaniel, 2009) is a rule-based runtime approach for Android application security that defines application protection at runtime,

depending on the caller and permission constraints. It protects Android applications from one another by their policies during the installation time and runtime interaction. In this case, it allows an application to define which application can access its interfaces and how other applications use those interfaces. This technique has the same limitation as Kirin security services. To define these security rules, a detailed understanding of malware and protection techniques is required, and security experts are required to perform the job.

AASandbox. AASandbox was the first technique to perform both static and dynamic analysis of the Android applications and was proposed by Bläsing, Batyuk, Schmidt, Camtepe and Albayrak (2010). The static analysis scans the Android applications for malicious patterns without installation on the Android platform, while in the dynamic analysis, the Android application is executed in a fully isolated platform called sandbox. It also intervenes and logs low-level interaction with the system for further analysis during application execution. In contrast, both the detection algorithm and sandbox algorithm are implemented in the cloud. AASandbox uses a system called foot-printing approach for detecting suspicious Android applications. In its early days, when AASandbox was proposed, there were no known botnet malware samples available to evaluate this technique, although it seems to be unmaintained nowadays.

Paranoid. Considering various factors of smartphone technology including resources, storage, processing and memory, Paranoid Android malware detection technique was proposed for the first time in mobile technology (Portokalidis, Homburg, Anagnostakis, & Bos, 2010). Paranoid Android is a security model implemented on remote servers (cloud server) to observe the dynamic behaviour of Android applications and to detect zero-day attacks, system call anomaly and antivirus file scanning. Both Crowdroid and Paranoid Android incur a 15-30% overhead for smartphone devices. This particular technique records information that is necessary for application execution and transmits it to a cloud server over an encrypted channel. While a complete replica of the executing application is running parallel on the remote virtual machine, the server can detect the potential malware using this technique. Both the application and its replica are executing parallel to one another, which may cause a lot of space and time complexity. It also converts energy by using 'loose synchronisation', which may cause loss of battery power usage that specifically sends information when the mobile user is using the mobile device.

Crowdroid. Crowdroid is a dynamic approach based on the behaviour of Android applications and was proposed by Burguera, Zurutuza and Nadjm-Tehrani (2011). Crowdroid is a lightweight application available online on Google Play Store, which can be downloaded and installed on Android smartphone devices. It monitors and collects the API calls of apps that are running on mobile devices and sends them to a centralised server after preprocessing. With the application of cluster algorithms, Android applications can be evaluated with this approach. The given approach is also able to detect self-written malware.

DroidRanger. DroidRanger is the combination of two systems based on permissions' behaviour, foot-printing and heuristic-based filtering (Odili, Kahar, Anwar, & Ali, 2017). It was proposed by Zhou, Wang, Zhou and Jiang (2012). This technique applies both the static and dynamic approaches to detect malicious applications in existing Android markets. Permissions-based behaviour foot-printing is used for the detection of known malware, while heuristic-based filtering is used for unknown malware Android applications. Despite the advancements in the detection approaches applied by DroidRanger, the system has some limitations; it requires manual operation for analysing and collecting behaviour of Android applications (Babu Rajesh, Reddy, Himanshu, & Patil, 2015). Manual operation takes more time than other detection techniques (Odili & Kahar, 2015).

Bouncer. Bouncer was proposed in 2012 by Oberheide and Miller (2012). It provides static and dynamic scanning together with Android applications that are automatically performed on the server. Google Play Store uses this technique to scan an Android application before hitting the application market (Penning, Hoffman, Nikolai, & Wang, 2014). Bouncer has the potential to take newly-uploaded applications to the app market. If this application is able to send an SMS to the malicious sites or detect other criminal activities, it classifies that Android application as malware. If not, it classifies it as benign. However, in this advanced era, it seems that cyber attackers have found ways to bypass detection. This technique is better for those who download applications from Google Play Store, while those who download applications from third-party app stores are not protected by this technique.

RobotDroid. RobotDroid is an Android malware detection technique that is based on SVM machine learning classifier algorithm and was proposed by Zhao, Zhang, Ge and Yuan (2012). This technique focusses on the signature of the applications. It has the ability to detect unknown malware like Plankton, DroidDream and Gemini. This framework can be used only for these few types of malware; this is the main limitation of this framework.

DroidScope. DroidScope designed by Yan and Yin (2012) is a fine-grained dynamic binary instrumentation tool for Android. It rebuilds two levels of semantic information: OS and Java. It provides an instrumentation interface that can be used to write plug-ins. It implements API tracing, native instruction tracing, Dalvik instruction tracing and taint tracking plug-ins. DroidScope works entirely on the emulator level and requires no changes to the Android sources. It runs the analysis outside the smartphone software stack and can analyse kernel-level attacks. This system has a big drawback: not able to detect real-time attacks. The second drawback is that it does not cover the subtleties of real devices (Enck et al., 2014).

Table 4 shows the list of Android botnet detection techniques with respect to year, major contribution and limitations.

Table 4
List of Android Botnet detection techniques

Ref	Techniques	Year	Architecture	Key Concept	Major Contribution	Limitation
Enck et al., 2009	Kirin	2009	Static	Based on permissions	Used rules to detect malware in installation time	No access to sensitive information of application; Cannot detect new malware
Szymczy, 2009	Multi-Agent System	2009	Hybrid	Based on permissions and rules	Used agent for traffic analysis	Slow interaction with the knowledge database
Ongtang et al., 2009	SAINT	2009	Static/ Dynamic	Based on rules	Installation time detection of malware	No access to sensitive information of application; Cannot detect new malware
Bläsing et al., 2010	AASandbox	2010	Static/ Dynamic/ Hybrid	Base on signatures and behaviour of the logs	Before instal detection of malware	NA
Portokalidis et al., 2010	Paranoid Android	2010	Dynamic/ Hybrid	Based on behaviour	Dynamic analysis, memory scanners, system call anomaly detection	Consumption of more time, space and power
Burguera et al., 2011	Crowdroid	2011	Dynamic	Based on behaviour	Client APK, behavioural detection	More clients, dynamic analyser
Zhou et al., 2012	DroidRanger	2012	Static/ Dynamic	Permissions-based behaviour	Detection of known and unknown malware, 0-day malware detection	Only a few of all possible execution paths are negotiated within one analysis run
Oberheide & Miller, 2012	Bouncer	2012	Static/ Dynamic	Permissions-based	Detection of unknown malware	Can be easily evaded by cybercriminals
Zhao et al., 2012	Robotdroid	2012	Static	Signature-based	Detection of unknown malware such as Plankton, DroidDream, and Gemini	Detects only specific malware families such as Plankton, DroidDream and Gemini
Yan & Yin, 2012	DroidScope	2012	Dynamic		Dynamic binary instrumentation	Cannot detect real-time botnet attacks
Zhou et al., 2012	DroidMOSS	2012	Dynamic	Permissions-based,	Fuzzy Hashing Technique,	Identifies only repackaged official Android market applications
Alparslan, Karahoca, & Karahoca, 2012	Data Mining	2012	Static	Behaviour-based	utilise auditing programmes to extract and extend features	Cannot detect real-time botnet attacks
Faruki, Ganmoor, Laxmi, Gaur, & Bharmal, 2013	AndroSimilar	2013	Static/ Dynamic	Statistical features	Improbable signature generation, thwarts obfuscation and repackaging	Limited malware DB, more false positives; Cannot detect new malware
Spreitzenbarth, Freiling, Echtler, Schreck, & Hoffmann, 2013	Mobile-Sandbox	2013	Static	Smali, emulator	Both static and dynamic analysis, obfuscation resistance, native API call track, web accessibility	More detection time

Table 4 *(continue)*

Rastogi, Chen, & Enck, 2013	AppsPlayground	2013	Static/ Dynamic	N/A	he-based UI interaction based on contextual exploration	Cannot detect real-time botnet attacks
Reina, Fattori, & Cavallaro, 2013	CopperDroid	2013	Dynamic	Behaviour-based approach	Automatically performs dynamic analysis, reconstructs behaviour of Android	Identifies only repackaged official Android market applications
Gascon, Yamaguchi, Arp, & Rieck, 2013	Embedded call graph	2013	Dynamic	Function call graphs	Obfuscation resistance	Undecidability of static call graph construction; Cannot detect new malware
Abdelrahman, Gelenbe, Görbil, & Oklander, 2013	NEMESYS	2013	Static	Model-based approach	Generate background traffic of network for simulating smaller set of users, learn Random Neural Network,	Limited to a small number of users, space complexity
Roshandel, Arabshahi, & Poovendran, 2013	LIDAR	2013	Static/ Dynamic	Behaviour-based	automatically detects, analyses, protects, remediates	N/A
Moonsamy, Rong, & Liu, 2014	Mini Permission Pattern	2013	Static/ Dynamic	Based on permission	'Used' permission extraction, informative data from contrast permission patterns	Careful analysis of permissions, no repackaging resistance; Cannot detect new malware
Enck et al., 2014	TaintDroid	2014	Static	Behaviour-based	Different APIs, specifically SMS APIs	It does not track implicit control flows due to performance overhead
Suarez-Tangil, Tapiador, Pens-Lopez, & Blasco, 2014	Dendroid Approach	2014	Dynamic	Code Chunks	Unknown malware classification, fast and scalable, dendograms	No obfuscation resistance, large feature vectors; Cannot detect new malware
Dhaya & Poongodi, 2014	N-gram analysis	2014	Static	N-gram CVSS	Produced N-grams signatures	No obfuscation resistance; Cannot detect new malware
Lindorfer, Neugschwandtner, Weichselbaum, Fratantonio, Van Der Veen, & Platzer,, 2014	Andrubis	2014	Static/ Dynamic	Based on behaviour and rules	Static and dynamic analysis on both Dalvik VM and System Level	Dynamic analysis consumes more space; Cannot be used for latest Android applications
Andronio, Zanero, & Maggi, 2015	Heldroid	2015	Static/ Dynamic	Behaviour-based	Static and dynamic analysis	Portability, internationalisation and evasion

DroidMoss. Zhou et al. (2012) proposed DroidMoss in 2012, using the fuzzy hashing technique to effectively localise and detect repackaged and injected applications. This technique detects Android applications in the existing mobile app market that are injected with malicious codes using the repackaging technique. The main feature of the applications used in this technique is the Dalvik opcodes. DroidMOSS calculates fuzzy hashes on each N sequential opcode,

which then applies a measure function on the two applications to realise their similarity quantitatively. The use of DroidMOSS is limited to identifying repackaged official Android market applications.

Data mining. The hardest part of the detection of malicious traffic is to differentiate C&C data flow from normal data flow behaviour. To overcome this limitation, data mining techniques are used to recognise the pattern by extracting the unexpected network patterns (Alparslan et al., 2012). Data mining is the most used machine learning device method for classification, prediction, regression and inference. This technique is extensively used in anomaly detection, especially in establishing generic and heuristic methods (Odili, Kahar, & Noraziah, 2016; Schultz, Eskin, Zadok, & Stolfo, 2001). Data mining approaches detect structures in a wide range of data, such as byte code, and use these structures to detect upcoming malicious occurrences in related data. Researchers such as Gu, Perdisci, Zhang and Lee (2008), Gu, Porras, Yegneswaran, Fong and Lee (2007), Gu, Zhang and Lee (2008), Wang, Huang, Lin and Lin (2011) and Yu, Dong, Yu, Qin, Yue and Zhao (2010) proposed BotMiner, BotHunter, BotSniffer and behaviour-based botnet detection systems based on the data-mining approach. This technique is very effective though it has some limitation as well. In experiments, BotMiner and BotHunter have been able to achieve 99% success rate with 1% false alarm and 99.2% success rate with 0.8% false alarm, respectively (Zhao et al., 2013).

AndroSimilar. AndroSimilar (Faruki et al., 2013) detects Android malware regions of statistical similarity starting from the .dex file. This method employs the similarity digest hashing system on byte-stream-based robust statistical malicious features. Similarly, a digest hashing scheme uses this feature to generate a list of signatures for this app. Here, the feature values between 100 and 990 are selected and the rest are discarded using the Bloom filter. A set of malicious signatures are generated and thus, a database of signatures is created. For testing a sample app, its signature is created in the same way as described above and is matched against a signature database and is considered malware if the similarity score crosses 35% (Sharma et al., 2016). Authors obtain an accuracy of 72.27% using a dataset of 101 malicious applications. Androsimilar performs at file level as an alternative for codes in decompiling; therefore, control of shared library is not protected. Also, porting the approach to constrained memory and a strong database remains a concern.

Mobile-SandBox. Mobile-SandBox is a static and dynamic analysis system that is publicly available. It was proposed by Spreitzenbarth, Schreck, Echtler, Arp and Hoffmann (2015). In this technique, the comparison of applications occurs in different stages: first, it compares the hash value with the VirustTotal database of the running application; second, it extracts the manifest file for permissions, background services, broadcast receivers and intents. This technique also extracts API calls from the Dalvik bytecode; thiss happen frequently in botnets. Mobile-SandBox makes it very easy to submit applications for static and dynamic analysis because of its user interface. A user can easily upload an application for static and dynamic analysis to the Mobile-SandBox by using the user interface. However, in some aspects, Mobile-SandBox seems unable to cope with the submission load.

AppsPlayground. AppsPlayground, based on TaintDroid, is a scalable dynamic analysis system that is used for detection of possible data leaks (Skovoroda & Gamayunov, 2015). Proposed by Rastogi, Chen and Enck (2013), it employs a Javaapp that connects to an emulator running a modified version of the OS and governs app behaviour exploration logic. Simply, the aim of AppsPlayground is to improve the stimulation of apps during dynamic analysis because it also detects dangerous API calls.

This technique also helps to create a more realistic analysis environment. It tries to drive the app along paths that are likely to reveal interesting behaviour through targeted stimulation of UI elements. This approach can be seen as an intelligent enhancement of the Application Exerciser Monkey and the custom stimulation of activity screens. This technique is largely orthogonal, as it focusses on stimulating broadcast receivers, services and common events instead of UI elements. Its main contribution is a heuristic-based intelligent black box (Monkey Exerciser-like) execution approach to explore the app's GUI (Odili, Kahar, Anwar, & Azrag, 2015). This technique can be more useful if combined with the static analysis technique.

CopperDroid. CopperDroid is a dynamic detection system presented by Reina, Fattori and Cavallaro (2013) that is built on top of the quick emulator (QEMU). To the best of our knowledge, this is the first technique that performs system call monitors of the Android applications out-of-the-box through virtual machine introspection (VMI) by reconstructing Dalvik behaviour and monitoring Binder communication (Lindorfer et al., 2014). CopperDroid carried the binder analysis to perform the reconstruction of high level Android-specific behaviour. It is available to the public as a web application that users can use to submit samples.

Embedded call graphs. Embedded call graphs is a static approach proposed by Gascon Yamaguchi, Arp and Reick (2013) in 2013. This technique can be used to find similarities between samples: first,-it-extracts function call graphs and then employs explicit mapping through kernel graphs from map call graphs to-feature-space. Sharma, Chawla and Gajrani (2016) showed that time and space complexity are high and large, respectively. Its key concept is functions call graphs, while obfuscation resistance is the major contribution. Embedded call graphs specially observe assembly-level analysis and support vector-machine implementation. The main disadvantage of this technique is that it cannot decide the static call graph construction.

NEMESYS. NEMESYS is a network model-based security solution that combines learning and modelling for detection of anomalies and attacks in mobile network. It deals with every mobile connection during communication between devices in a network. The motivation behind this approach was the difference between the number of mobile users who are monitored and dealt with in real time. Furthermore, a clear and understandable approach was needed to deal with every unique call. The second consideration in constructing this approach was the computational tools that were developed for anomaly detection that were based on mathematical models. However, NEMESYS has some limitations. For instance, it is limited to a small number of users. Also, this approach is complex and it uses a huge amount of memory.

Layered intrusion detection and remediation. The Layered Intrusion Detection and Remediation (LIDAR) framework focusses on automatic detection, analysis, protection and remediation of security threats. This framework is specially designed to detect intrusion in a multi-dimensional mode that is of network dimension, application dimension and social dimension (Roshandel, Arabshahi, & Poovendran, 2013). This approach contains both local and remote analyses, which causes some drawbacks. For instance, a set of local analyses is performed in the mobile device to detect malware and intrusion, leading to the use of more battery power in addition to time consumption and space complexity, among other issues.

TaintDroid. TaintDroid is a system-wide dynamic taint tracking and analysis system for simultaneous tracking of multiple sources of sensitive data. This technique monitors methods, variables, files and messages during application execution according to data flow (Suarez-Tangil et al., 2014). TaintDroid uses tag chunks to keep track of data in order to find information leakage at runtime. Information flow tracking needs lots of memory. None of these schemes is energy-efficient; hence, they are not suitable for resource-constrained mobile platforms.

Dendroid approach. The dendroid approach is based on text mining and information retrieval techniques (Suarez-Tangil et al., 2014). This technique extracts code chunks (CC) to analyse and classify the code structures in Android malware families. The authors present a simple way to measure the similarity between malicious applications by formulating the modelling process. In the experiments performed, more than 33 families with 1249 malware applications (Sharma et al., 2016) were detected. This approach also provided automatic classification of zero-day malware samples, which is based on the applications-code structure. With respect to time and accuracy, this technique is very fast and accurate, with high scalability. However, this technique features vector growth and new families create issues, while the strategies of obfuscation are not implemented.

N-gram. The N-gram [12] analysis is a probabilistic approach to detect the presence of malware. Reverse engineering tools like DexToJar, Java Decompiler-Graphical User Interface (JD-GUI) and ApkTool are used in this technique to convert executable to source code (high level or low-level language), thereby creating the training dataset. After this, the source code is considered as N-gram signatures. This N-gram model is a popular machine-learning algorithm and it is a type of probabilistic language that predicts the next item in the sequence with given datasets of order (N−1) as in the Markov Model. These signatures are then stored in a Comma Separated Values (CSV) file for the reason that signatures occupy a lot of space.

After this, a Common Vulnerability Scoring System (CVSS) is used to assess the vulnerabilities' severity level in software applications. It is a freeware tool. By applying this tool on the APK file under test, the description of all the vulnerabilities and solutions to mitigate the same is appended to the CSV file. It makes intuitive use of the N-gram machine-learning algorithm to analyse the Android apps. Limitations include the obfuscation techniques not being implemented.

Andrubis. Andrubis is a cloud-based malware detection technique proposed by Lindorfer et al. (2014). This technique combines both static and dynamic analyses on both Dalvik VM and the system level. First, it performs the static analysis by extracting the information including broadcast receivers, requested permissions, activities, services, SDK version and package name from the application manifest and its bytecode. Andrubis uses the modified DroidBox output to generate XML files that contain the analysed results. In the dynamic stage, it executes the application in a complete Android environment; during the execution its action is monitored at both the Dalvik and the system level. Other than this, Andrubis provides a web interface for users to submit Android applications and, so far, it has collected a dataset of over one million Android applications, 40% of which are malware. The only disadvantage of this technique is that it cannot track native codes. API calls that frequently happen in botnet are extracted from the Dalvik code, while the Andrubis is limited to the application's API level 8.

HELDROID. HELDROID is a fully automated behaviour-based approach to recognise known and unknown ransomware and scareware (Andronio et al., 2015). This approach analyses the Android application statically and dynamically as well. By using the static taint analysis approach, it analyses the function calls flow. Its result is more accurate when compared to those of previous apps. Still, it has some limitations. Although they focus on the mobile case, the results shown are far from being accurate. Ransomware is a general problem, but in this approach, it is limited to mobile devices only. HELDROID is based on sentence structure; this needs internationalisation.

CHALLENGES AND FUTURE DIRECTIONS

In this section, we discuss the challenges and future directions based on the findings of our study. Research into Android botnets is still in its initial stages. Therefore, sufficient opportunity exists for the betterment of detection and prevention of botnet attacks. The following challenges will help the researchers, academics and industry players to enhance this field.

Hybrid Approach Towards Android Botnet Detection

There is no way to ignore the rising security threats to mobile devices at this time. Researchers and industry players have proposed different botnet detection techniques. Practically, most of the existing Android botnet detections are either static or dynamic, and can detect known and unknown Android botnets. As can be seen in Table 2, few of the existing detection techniques based on the hybrid approach have a low detection rate and maximum false-alarm rate. This is a disturbing discovery.

Limitation of Mobile Devices

Personal computers are considered a more suitable platform for botnet attacks compared with mobile devices. Mobile devices have certain limitations, such as limited power storage, limited memory storage, limited Internet access and resource constraints. They attract

cybercriminals because of the open environment they operate in as well as their availability, Internet connectivity and storage capacity.

Existing Android botnet detection techniques are based on the estatic and dynamic approaches. These require heavy battery consumption, which is the most crucial challenge in protecting mobile devices (Ali, Zain, Zolkipli, & Badshah, 2014). Furthermore, the dynamic approach of scanning and blocking malicious codes needs a runtime environment. This is a big issue for mobile devices due to their limited battery power.

Internet Service Providers Should Also Provide Security Measurements

One of the biggest challenges for Internet Service Providers (ISP) is to protect mobile device users from the botnet threats as they do not use static addresses. Mobile device users are continuously changing their location; this creates difficulties for ISPs.

Difficulties in Estimating Botnet Size

It is very difficult to estimate the botnet size. To the best of our knowledge, there is no technique that estimates the size of compromised bots in a botnet. With rapid developments in detecting botnet attacks in Android devices, researchers need to find a quantitative methodology to find the number of bots in a botnet (Odili & Kahar, 2016).

CONCLUSION

Android botnets are harmful to Android devices. The popularity of mobile devices has made it a soft target for potential attacks. This survey aimed to find the real threat behind Android botnets. We conducted a comprehensive survey of existing Android botnets and their detection techniques. We categorised the detection techniques according to their detection environment, such as static, dynamic and hybrid detection techniques. Limitations in the existing Android botnet detection techniques as well as their benefits are listed here in an organised way. To the best of our knowledge this is the most current organised survey on Android botnets and their detection techniques. This research will help both academics and industry players.

ACKNOWLEDGEMENT

The authors are grateful to the Faculty of Computer Systems & Software Engineering (FSKKP), Universiti Malaysia Pahang for funding this research study under the Grant GRS140392.

REFERENCES

Abdelrahman, O. H., Gelenbe, E., Görbil, G., & Oklander, B. (2013). Mobile network anomaly detection and mitigation: The NEMESYS approach. *Information Sciences and Systems 2013* (pp. 429–438). Switzerland: Springer.

Ali, M., Zain, J. M., Zolkipli, M. F., & Badshah, G. (2014). Mobile cloud computing & mobile battery augmentation techniques: A survey. In *Research and Development (SCOReD), 2014 IEEE Student Conference* (pp. 1-6). IEEE.

Alparslan, E., Karahoca, A., & Karahoca, D. (2012). BotNet detection: Enhancing analysis by using data mining techniques. In A. Karahoca (Ed.), *Advances in Data Mining Knowledge Discovery and Applications*. INTECH Open Access Publisher.

Alta, V. D. M., Loock, M., & Dabrowski, M. (2005). Characteristics and responsibilities involved in a phishing attack. In *Proceedings of the 4th International Symposium on Information and Communication Technologies* (pp. 249-254). Trinity College Dublin.

Andronio, N., Zanero, S., & Maggi, F. (2015). HelDroid: Dissecting and detecting mobile ransomware. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 382-404). Springer International Publishing.

Anwar, S., Inayat, Z., Zolkipli, M. F., Zain, J. M., Gani, A., Anuar, N. B., . . . & Chang, V. (2017). Cross-VM cache-based side channel attacks and proposed prevention mechanisms: A survey. *Journal of Network and Computer Applications, 93*, 259–279.

Anwar, S., Mohamad Zain, J., Zolkipli, M. F., & Inayat, Z. (2014). A review paper on botnet and botnet detection techniques in cloud computing. In *ISCI 2014 – IEEE Symposium on Computers and Informatics* (pp. 28-29). IEEE.

Anwar, S., Zain, J. M., Inayat, Z., Haq, R. U., Karim, A., & Jaber, A. N. (2016, August, 11–12,). A static approach towards mobile botnet detection. In *2016 3rd International Conference on Electronic Design (ICED)* (pp. 563-567). IEEE.

Anwar, S., Zain, J. M., Zolkipli, M. F., Inayat, Z., Jabir, A. N., & Odili, B. (2015). Response option for attacks detected by intrusion detection system. In *4th International Conference on Software Engineering and Computer System* (pp. 195-200). IEEE.

Anwar, S., Zain, J. M., Zolkipli, M. F., Inayat, Z., Khan, S., Anthony, B., & Chang, V. (2017). From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. *Algorithms, 10*(2), 1–23.

Babu Rajesh, V., Reddy, P., Himanshu, P., & Patil, M. U. (2015). Droidswan: Detecting malicious Android applications based on static feature analysis. *Computer Science and Information Technology* (pp. 163-178). Centre for Development of Advanced Computing, India.

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009). A survey of botnet technology and defenses. In *Conference for Homeland Security, 2009. CATCH'09. Cybersecurity Applications and Technology* (pp. 299-304). IEEE.

Bläsing, T., Batyuk, L., Schmidt, A. D., Camtepe, S. A., & Albayrak, S. (2010). An android application sandbox system for suspicious software detection. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference* (pp. 55-62). IEEE.

Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-Based malware detection system for Android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (pp. 15-26). ACM.

Castiglione, A., De Prisco, R., De Santis, A., Fiore, U., & Palmieri, F. (2014). A botnet-based command and control approach relying on swarm intelligence. *Journal of Network and Computer Applications, 38*, 22–33.

Christian, F., & Maria, G. (2013). *Kaspersky security bulletin 2013*. Retrieved from http://www.securelist.com/en/

Cooke, E., Jahanian, F., & McPherson, D. (2005). The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop* (pp. 6-6). USENIX.

Dhaya, R., & Poongodi, M. (2014). Detecting software vulnerabilities in Android using static analysis. Paper presented at the *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference* (pp. 915-918). IEEE.

Dong, D., Wu, Y., He, L., Huang, G., & Wu, G. (2008). Deep analysis of intending peer-to-peer botnet. *In 2008 Seventh International Conference on Grid and Cooperative Computing* (pp. 407-411). IEEE.

Doorey, A. M. (2016). *Contextualizing privacy concerns within mobile engagement: a comparative investigation of escalating risk among general, e-commerce and health-related use.* (Doctoral dissertation). The University of Texas, Austin.

Dunne, P. (2006). *The art of pishing: How to attract birds by mimicking their calls*. United States of A,erica, USA: Stackpole Books.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., ... & Sheth, A. N. (2014). TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS), 32*(2), 5.

Enck, W., Ongtang, M., & McDaniel, P. (2009). On lightweight mobile phone application certification. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (pp. 235-245). ACM.

Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys and Tutorials, 17*(2), 998–1022.

Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., & Bharmal, A. (2013). Androsimilar: Robust statistical feature signature for android malware detection. In *Proceedings of the 6th International Conference on Security of Information and Networks* (pp. 152-159). ACM.

Fossi, M., Egan, G., Haley, K., Johnson, E., Mack, T., Adams, T., ... & McKinney, D. (2011). Symantec internet security threat report trends for 2010. *Symantec Enterprise Security, 16*, 1-20.

Gascon, H., Yamaguchi, F., Arp, D., & Rieck, K. (2013). Structural detection of android malware using embedded call graphs. In *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security* (pp. 45-54). ACM.

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium* (pp. 139-154). USENIX.

Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007). BotHunter: Detecting malware infection through IDS-driven dialog correlation. Paper presented at the *USENIX Security Symposium* (Vol. 7, pp. 1-16). USENIX.

Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting botnet command and control channels in network traffic. In *NDSS* (Vol. 8, pp. 1-18).

Ianelli, N., & Hackworth, A. (2005). Botnets as a vehicle for online crime. *CERT Coordination Center, 1*(1), 28.

Inayat, Z., Gani, A., Anuar, N. B., Anwar, S., & Khurram Khan, M. (2017). Cloud-Based intrusion detection and response system: Open research issues, and solutions. *Arabian Journal for Science and Engineering, 7*(65), 1–25.

Inayat, Z., Gani, A., Anuar, N. B., Khan, M. K., & Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications, 62*, 53–74.

Jelasity, M., & Bilicki, V. (2009). Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In *2nd USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)* .

Karim, A., Bin Salleh, R., Shiraz, M., Shah, S. A. A., Awan, I., & Anuar, N. B. (2014). Botnet detection techniques: Review, future trends, and issues. *Journal of Zhejiang University-Science C-Computers and Electronics, 15*(11), 943–983. doi:DOI 10.1631/jzus.C1300242

Karim, A., Shah, S. A. A., Salleh, R. B., Arif, M., Md Noor, R., & Shamshirband, S. (2015). Mobile botnet attacks – An emerging threat: Classification, review and open issues. *KSII Transactions on Internet and Information Systems, 9*(4), 1471–1492. doi:10.3837/tiis.2015.04.012

Khattak, S., Ramay, N., Khan, K., Syed, A., & Khayam, S. (2014). A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys and Tutorials, 16*(2), 898–924.

LAB, K. (2015). *Kaspersky security bulletin 2015*. Retrieved February, 2016, from www.kaspersky.com

Lashkari, A. H., Ghalebandi, S. G., & Moradhaseli, M. R. (2011). A wide survey on botnet. *Digital information and communication technology and its applications* (pp. 445–454): Berlin, Heidelberg: Springer.

Li, C., Jiang, W., & Zou, X. (2009). Botnet: Survey and case study. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference* (pp. 1184-1187). IEEE.

Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., Van Der Veen, V., & Platzer, C. (2014). Andrubis-1,000,000 apps later: A view on current android malware behaviors. In *2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)* (pp. 3-17). IEEE.

Liu, L., Chen, S., Yan, G., & Zhang, Z. (2008). Bottracer: Execution-based bot-like malware detection. In T. C. Wu, C. L. Lei, V. Rijmen & D. T. Lee (Eds.), *Information Security* (pp. 97–113). Germany: Springer.

LulzSec. (2011). *Distributed denial of service attack (DDoS) definition*. Retrieved from http://www.incapsula.com/

Milletary, J., & Center, C. C. (2005). *Technical trends in phishing attacks*. *Retrieved December, 1*(2007), 3-3.

Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review, 34*(2), 39–53.

Moonsamy, V., Rong, J., & Liu, S. (2014). Mining permission patterns for contrasting clean and malicious android applications. *Future Generation Computer Systems, 36*, 122–132.

Naraine, R. (2012, February 27). Android drive-by download attack via phishing sms: ZDNet. Zero Day. Retrieved from http://www.zdnet.com/article/android-drive-by-download-attack-via-phishing-sms/

Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing, 20*(1), 343–357.

Naser, A., Zolkipli, M. F., Majid, M., & Anwar, S. (2014). Trusting cloud computing for personal files. In *Information and Communication Technology Convergence (ICTC), 2014 International Conference* (pp. 488-489). IEEE.

Oberheide, J., & Miller, C. (2012). *Dissecting the android bouncer* (pp. 1-62). SummerCon2012, New York.

Odili, J. B., Kahar, M. N. M., Anwar, S., & Ali, M. (2017). Tutorials on African buffalo optimization for solving the travelling salesman problem. *International Journal of Software Engineering and Computer Systems, 3*(3), 120–128.

Odili, J. B., & Kahar, M. N. M. (2015). African buffalo optimization (ABO): A new meta-heuristic algorithm. *Journal of Advanced and Applied Sciences, 3*(3), 101–106.

Odili, J. B., & Kahar, M. N. M. (2016). Solving the traveling salesman's problem using the African buffalo optimization. *Computational Intelligence and Neuroscience, 2016*(2916), 1–12.

Odili, J. B., Kahar, M. N. M., & Anwar, S. (2015). African buffalo optimization: A swarm-intelligence technique. *Procedia Computer Science, 76*, 443–448.

Odili, J. B., Kahar, M. N. M., & Noraziah, A.. (2016). Convergence analysis of the African buffalo optimization algorithm. *International Journal of Simulations: Systems, Science and Technology, 17*(44), 44.41–44.46.

Odili, J. B., Kahar, M. N. M., Anwar, S., & Azrag, M. A. K. (2015). A comparative study of African buffalo optimization and randomized insertion algorithm for asymmetric travelling salesman's problem. In *Software Engineering and Computer Systems (ICSECS), 2015 4th International Conference* (pp. 90-95). IEEE.

Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P. (2009). Semantically rich application-centric security in Android. *Security and Communication Networks, 5*(6), 658-673.

Peng, S., Yu, S., & Yang, A. (2014). Smartphone malware and its propagation modeling: A survey. *IEEE Communications Surveys and Tutorials, 16*(2), 925–941.

Penning, N., Hoffman, M., Nikolai, J., & Wang, Y. (2014). Mobile malware security challeges and cloud-based detection. In *Collaboration Technologies and Systems (CTS), 2014 International Conference* (pp. 181-188). IEEE.

Plohmann, D., Gerhards-Padilla, E., & Leder, F. (2011). Botnets: Detection, measurement, disinfection and defence. *European Network and Information Security Agency (ENISA), 1*(1), 1-153.

Portokalidis, G., Homburg, P., Anagnostakis, K., & Bos, H. (2010). Paranoid Android: Versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference* (pp. 347-356). ACM.

Preda, M. D., Christodorescu, M., Jha, S., & Debray, S. (2008). A semantics-based approach to malware detection. *ACM Transactions on Programming Languages and Systems (TOPLAS), 30*(5), 25.

Rastogi, V., Chen, Y., & Enck, W. (2013). AppsPlayground: Automatic security analysis of smartphone applications. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy* (pp. 209-220). ACM.

Reina, A., Fattori, A., & Cavallaro, L. (2013). *A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors*. EuroSec.

Roshandel, R., Arabshahi, P., & Poovendran, R. (2013). LIDAR: A layered intrusion detection and remediationframework for smartphones. In *Proceedings of the 4th international ACM Sigsoft symposium on Architecting Critical Systems* (pp. 27-32). ACM.

Rubin, A. (2008). Google bets on Android future. Retrieved from http://news.bbc.co.uk/2/hi/technology/7266201.stm

Sauter, D. (2013). *Rapid android development: Build rich, sensor-based applications with processing*. Raleigh, NC: Pragmatic Bookshelf.

Schultz, M. G., Eskin, E., Zadok, E., & Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001 Symposium, S and P 2001. IEEE Symposium* (pp. 38-49). IEEE.

Sears, N. (2007). *Android.* Retrieved from http://www.openhandsetalliance.com/index.html

Shameli, S. A., Cheriet, M., & Hamou-Lhadj, A. (2014). Taxonomy of intrusion risk assessment and response system. *Computers and Security, 45*, 1–16. doi:10.1016/j.cose.2014.04.009

Sharma, M., Chawla, M., & Gajrani, J. (2016). A survey of Android malware detection strategy and techniques. In *Proceedings of International Conference on ICT for Sustainable Development* (pp. 39-51). Springer Singapore.

Sheta, M. A., Zaki, M., El Salam, K. A., & Hadad, E. (2015). Design and implementation of anti spyware system using design patterns. *International Journal of Computer Applications, 123*(2), 9-13.

Silva, R. M., Pinto, R. C., & Salles, R. M. (2013). Botnets: A survey. *Computer Networks, 57*(2), 378–403.

Skovoroda, A., & Gamayunov, D. (2015). Review of the mobile malware detection approaches. In *Parallel, Distributed and Network-Based Processing (PDP) Conference, 2015 23rd Euromicro International Conference* (pp. 600-603). IEEE.

Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., & Hoffmann, J. (2013). Mobile-sandbox: Having a deeper look into android applications. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 1808-1815). ACM.

Spreitzenbarth, M., Schreck, T., Echtler, F., Arp, D., & Hoffmann, J. (2015). Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques. *International Journal of Information Security, 14*(2), 141–153.

Stevanovic, M., Revsbech, K., Pedersen, J. M., Sharp, R., & Jensen, C. D. (2012). A collaborative approach to botnet protection. In *CD-ARES* (pp. 624-638).

Suarez-Tangil, G., Tapiador, J. E., Pens-Lopez, P., & Blasco, J. (2014). DENDROID: A text mining approach to analyzing and classifying code structures in Android malware families. *Expert Systems with Applications, 41*(4), 1104–1117. doi:10.1016/j.eswa.2013.07.106

Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A. (2014). Evolution, detection and analysis of malware for smart devices. *Communications Surveys and Tutorials, IEEE, 16*(2), 961–987.

Szymczyk, M. (2009). Detecting botnets in computer networks using multi-agent technology. In *Dependability of Computer Systems, 2009. DepCos-RELCOMEX'09 Fourth International Conference* (pp. 192-201). IEEE.

Teufl, P., Ferk, M., Fitzek, A., Hein, D., Kraxberger, S., & Orthacker, C. (2013). Malware detection by applying knowledge discovery processes to application metadata on the Android market (Google Play). *Security and Communication Networks, 9*(5), 389-419.

Thompson, B., Morris-King, J., & Cam, H. (2016, November). Controlling risk of data exfiltration in cyber networks due to stealthy propagating malware. *In Military Communications Conference, MILCOM 2016-2016 IEEE* (pp. 479-484). IEEE.

Wang, K., Huang, C. Y., Lin, S. J., & Lin, Y. D. (2011). A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks, 55*(15), 3275–3286. doi:10.1016/j.comnet.2011.05.026

Yan, L. K., & Yin, H. (2012). Droidscope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In *21st USENIX Security Symposium* (USENIX Security 12) (pp. 569-584). USENIX.

Yu, X., Dong, X., Yu, G., Qin, Y., Yue, D., & Zhao, Y. (2010). Online botnet detection based on incremental discrete Fourier transform. *Journal of Networks, 5*(5), 568–576.

Zang, X., Tangpong, A., Kesidis, G., & Miller, D. J. (2011). Botnet detection through fine flow classification. *Unpublished, Report No. CSE11-001*.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers and Security, 39*, 2–16. doi:10.1016/j.cose.2013.04.007

Zhao, M., Zhang, T., Ge, F., & Yuan, Z. (2012). RobotDroid: A lightweight malware detection framework on smartphones. *Journal of Networks, 7*(4), 715–722.

Zhou, Y., Wang, Z., Zhou, W., & Jiang, X. (2012). *Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets.* In *NDSS* (Vol. 25, No. 4, pp. 50-52).