SMS Botnet Detection for Android Devices through Intent Capture and Modeling

Erik Johnson
Department of Electrical and Computer Engineering
University of Victoria, Victoria, BC, Canada
erikj@uvic.ca

Issa Traore
Department of Electrical and Computer Engineering
University of Victoria, Victoria, BC, Canada
itraore@ece.uvic.ca

Abstract— **Mobile devices are subject to an increased attack surface vector as compared to desktop computing, due to the nature of sensors, radios, and increased peripherals. We investigate in this work mobile botnets with a specific focus on Android, which is the most widely adopted mobile platform, and a prime target for malicious software; 79% of reported malware threats to mobile operating systems are targeted at Android. Our analysis focuses on a short messaging service (SMS) botnet structure and investigates a new detection model using the concept of intents. We show that transparent control can be achieved by a remote endpoint; yet also detected by our proposed intent detection model. Intents are late run-time bindings mechanisms provided to applications in the Android operating system. Intents provide a clear and accurate picture of device behaviour with external sources, due to their design as a late run time binding mechanism in the Android Operating System. We propose an intent logging system to capture sample data, and use this as the basis to design and evaluate our proposed detection scheme.**

Keywords— mobile botnets; mobile security; SMS botnets; intent modeling, android OS; botnet detection.

I. INTRODUCTION

The rapid growth and increasing sophistication of mobile devices has caused a paradigm shift in computing in the past decade. Institutions, consumers, and businesses are moving personal and business information onto mobile phones, tablets, and other mobile devices. As these entities move their data onto mobile devices, security continues to increase in importance.

One of the threats well studied on traditional computing devices is botnets. In contrast, botnets represent a new phenomenon in mobile computing, for which appropriate countermeasures have yet to emerge. Botnets provide a group or an individual a means of controlling remotely a collection of enslaved machines referred to, as bots; the group or individual controlling the botnet is known as the botmaster [4, 5,11]. Botnet communication is carried out by establishing command and control (C&C) channels between the bots, and the command center operated by the botmaster.

In the mobile world, the detection and analysis of botnets pose a relatively new challenge, in comparison to the desktop domain. Mobile devices differ from desktops in sensors and communication capabilities available for exploitation.

Botnets have a landscape in the mobile world that involves less restrictive OS's, and more divergent architectures, including cellular capabilities, external sensors, and traditional IP traffic. Specifically the architecture of the Android OS represents a landscape of nearly 12,000 distinct Android versions in 2013 [6], representing a year over year increase of 400% in the divergent versions of Android detected in 2012.

The focus of this paper is limited to communications in the 2nd generation of mobile network technology (2G), leaving the future generations of technology open for future research. The 2G-network technology represents additional capabilities over the first generation by introducing data services to the mobile world via text and picture messaging. The text-based data communication is commonly referred to as short messaging service (SMS). SMS capabilities on mobile devices represent a separate protocol for communication and exchange designed for mobility. The implementation protocols underlying the 2G standard have a distinct design separate from traditional IP-based layered networks. The customization of 2G networks renders the traditional botnet detection mechanisms ineffective on mobile networks, due to the difference of traffic flow patterns and protocol stacks in SMS and 2G networks. Likewise, 2G communication protocols do not follow a similar flow pattern and packet structure as traditional IP based and HTTP traffic. 2G communication structures are based on a circuit switched network structure, in comparison to the packet switch nature of the IP/HTTP stack on traditional desktops. This circuit vs. packet switched nature changes the nature of botnet detection.

In recent years, SMS-based botnets have appeared as one of the first types of botnets found exclusively in mobile devices, allowing devices to be controlled transparently to users via mobile networks [1, 3].

Our research focuses on the core ability of SMS to facilitate botnets and provide a facility to control, and respond to messaging through a mobile network.

Our hypothesis is that by developing a model of intent-based behaviour on the device, we can detect malicious SMS style botnets. Intents are a late run-time binding facility on Android that allows interaction across applications for the purpose of security and communication. Intents form the cornerstone of interaction across applications.

By analyzing the inflow of SMS messages using intents, we propose a model to detect non-user initiated and malicious SMS messages on an Android operating system (OS) based mobile client. Intents provide us the ability to listen and respond to incoming and outgoing communications using registered listeners, the intended facility as a late runtime binding mechanism in the Android OS. The function of intents in the Android OS is to provide applications a way of invoking and performing actions that would normally require system level permissions, crossing application sandboxes, and performing actions without root privileges [7].

978-1-5090-0092-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SRDSW.2015.21

Our detector involves a mobile listening platform that enables the logging of communication on a mobile device and is able to communicate logs to a central server for further processing. The central server provides the ability to perform further analysis, and amalgamates data from multiple sources to detect and analyze intents.

The rest of the paper is structured as follows. Section 2 summarizes and discusses related work. Section 3 introduces the Android intent model. Section 4 presents the implementation of our intent logger and our experimental dataset. Section 5 presents our detection approach and model. Section 6 presents the experimental evaluation of the approach, and finally Section 7 makes concluding remarks.

II. Related Works

While the mobile malware threat has been on the rise, limited research work has so far been published on mitigating such threat. A prime target of existing mobile malware is SMS. Compromised devices are used as stepping stones to send bulk of unsolicited SMS (i.e. spam) to a large number of mobile device users, at the expense of the compromised devices owners, who must pay the bills. Vural and Venter [1] provided an overview of mobile botnets, and identified a few characteristics that could be used to detect them, specifically, the number of recipients per SMS and the frequency of SMS sent from a device. Their basic assumption is that a high ratio of outgoing to incoming SMS is a sign of a mobile botnet generating spams. They proposed around this concept a mobile botnet detection system by extracting recurring patterns from weekly SMS traffic volume using statistical analysis.

In a follow up work [2], using stylistic features extracted from SMS (e.g. number of capital letters, number of white spaces, number of punctuation characters, and number of digits), the authors designed a SMS botnet detection model using artificial immune system (AIS). A proof of concept implementation was developed for Android platform, and then evaluated experimentally. Specifically 60 valid SMS (i.e. sent by the user) were used for training, while 14 valid and 6 spam SMS were used for testing, achieving in the best case a False Positive Rate (FPR) of 7% and a Detection Rate (DR) of 83%. While these results are encouraging, they are based only on data from a single user and phone, which is not enough to draw meaningful conclusion.

Geng et al. [3] developed a botnet structure that uses a SMS-based heterogeneous multi-tree topology to implement bot C&C channels. In particular, botnet C&C messages are sent by SMS, via encrypted channels for some of the most critical messages. According to the authors the heterogeneous multi-tree topology ensures robustness and scalability of the botnet architecture. SMS-based botnet C&C addresses several challenges inherent to mobile platforms that make traditional botnets (e.g. HTTP, IRC, P2P-based botnets) unsuitable for such environments. Some of these challenges include limited power resource, monetary costs of the communications, constant or unpredictable changes in connectivity and network configuration, and the lack of IP address. The proposed botnet structure overcomes the above challenges by taking advantage adequately of the characteristics of SMS.

We propose in this work a new approach for SMS botnet detection that uses for the first time Android intent model. We also conduct an experimental evaluation using data from 7 different users.

III. Android OS and Intent Model Overview

A. Android Component Architecture

Applications form the ecosystem of available functions on mobile devices. There are 2 distinct types of applications in the Android OS: system and user level applications. System applications generally reside in a read only folder on a device and are a merger of OEM and Android base system applications. User level applications can reside on device or external storage cards and have a reduced permission set in comparison to system level applications.

Android provides a layered architecture. The Application layer provides the layer where traditional “Apps” reside. Using permissions,,,, The permission model focuses on a sandboxed architecture of applications. Applications are sandboxed and only given access to requested and approved device functions. A manifest file is provided that allows an application to request access to certain device functions [7].
When given privileged access either through user granted permission, or through privilege escalation, the user can perform malicious activities unrestrictedly such as sending premium SMS messages costing as high as 10$/message [9], or conducting fraudulent mobile banking transactions [8].

B. Android Intent Model

Intents have been well documented as they form a basis of the Android Interaction model. Detailed information about the API and interaction mechanisms is provided in the Android Standard API documents [10]. Intents are objects that allow late run-time binding between components internally and externally in an application.
Intents, however, can be misused by enabling privilege escalation attacks. A privilege escalation attack in this context exploits the ability for an application to create its proper intents, and expose permissions to outside application via intents, that were not previously desired, or granted by the users on an Android OS device.
Intents are used in conjunction with filters; filters provide the ability to resolve intents. Intents and intent filters provide the ability to broadcast information surrounding events that are pending execution, or previously occurred. Intents originate from system level services or user level applications on an Android phone.
Intents provide a mechanism of capturing and responding to actions and events on the device, without requiring system-level permission for an application. To facilitate the desired operations, intents are structured as follows:

- Component: Optional parameter used for intent resolution.

- Action: Action to be performed or that has taken place that is now being broadcast, e.g., SMS_RECEIVED.
- Data: A URI or entry that displays the data to be acted upon when the intent is resolved, e.g., http:// or sms://.
- Category: A category that defines information about the action to execute. The category varies based on how an application is to be invoked and the context during which it is invoked.
- Extras: A bundle of extra information that is supplied in the intent. For SMS or email, the body or subject of the email will be attached as a bundled extra inside this object.
- Timestamp: provides information surrounding when an intent is activated.

The Android OS runtime supports two main forms of intents, namely, explicit and implicit intents. Explicit intents designate the component for invocation precisely, while implicit intents do not name a target but rely on the intent filters that inspect the content of the intents to find suitable components and targets. The resolution is what provides the facility of late run-time binding by allowing dynamic registration for intents.

C. *Malicious Application Profiles*

Malicious exploits exist in several forms: malicious applications, OS level exploits and traditional browser-based exploits. These exploits have led to the creation of botnets and botnet style SMS relay applications [9]. SMS relay, or spamming type applications function as a bot style application that responds to commands via SMS messaging.

In this section we examine the profile of malicious applications based on the premise that a bot, or relay SMS application is an undesired infection on a mobile client. The profiles of malicious applications take two distinct forms: First, an application, which is installed with user permission, but performs malicious activity; and second, an application, which is installed without user permission – but acts in a same manner as above (i.e. performing malicious activity). Both application types will use the SMS capabilities of an Android client. To access the capabilities of a client to send and receive SMS messages programmatically, these applications must register with the correct permission. The act of registering, and listening allows us to know when this application is invoked, as an intent object will be created and logged in the operating system.

The profile of a malicious SMS relay bot type application is such that a triggered SMS message will send SMS spam, or respond by sending out a reply to the address that it is indeed listening, and can perform further actions.

During SMS communication, a malicious application exhibits the following characteristics when communicating maliciously:

- Transparent Spam Communication: Utilizing the Android SEND_SMS permission, the application requests to send information transparently through the android interface. Once granted, the application can send SMS messages, transparently to device operators.
- SMS Bot Commands: The application exploits the ability to send SMS based information transparently to users. SMS Bot style commands can be received, and interpreted by a bot listening to intents.

To register for permissions to send or receive an SMS via an intent object malicious applications use a manifest file, and an intent filter. The manifest file is parsed by the base operating system, and provides the ability to be invoked on SMS received, or allows an SMS message to be sent programmatically by the system. Of note is the receiver and intent filter which allow the specified object to be notified on an intent being triggered.

To actually receive an intent from the operating system, we need not just the permission, and intent filter outlined above, but also a method able to deal with the intent object.

This system allows an application to receive an SMS message intent by way of the *onReceive* method through subclassing the Android *BroadcastReceiver* class. The application can read the message and issue a command accordingly to a centralized server, or start an SMS attack from the mobile client.

IV. Intent Logging And Datasets

A. *Activity Events*

Based on the characteristics of intents, outlined earlier, we propose to extract the following factors for the detection of SMS-based mobile botnets:

- SMS Received Intent
- SMS Sent Intent
- Pre SMS Intent invoked applications (i.e. SMS Application conversation list)
- Post received SMS Intent Category (i.e. SMS Sent intent or other invocation)
- Time difference SMS received vs. SMS sent intents

Table 1 SMS Send compared to Receive and Component listing

The activity we collect from the device, and report to the server for evaluation takes the form of an intent object. Activity events are collected continuously and submitted to the detector for processing. Each activity event is associated with a timestamp, giving it context relative to the time when the intent occurred. The format of the activity is as follows:

- ID: the identifier of the activity recorded in the database,
- Phone_ID: the identifier of the phone for which the activity is recorded against,

- Action: the action of the intent object,
- Category: the category of the intent,
- Component: the component,
- Details: the details of the bundle,
- Timestamp: the timestamp as logged on device when the intent is received.

This activity entry forms the basis for our data collection. Table 1 shows an example of activity event related to malicious behaviour.

B. Intent Logger and Detector Implementation

Our detection system comprises a client as an application that is loaded onto a non-rooted Android device, as well as a server logging communication to a central data store from the deployed client applications. The significance of a non-rooted device is that it allows us to detect the intents, and behaviour across all Android devices, in contrast to much research in this area that is subject to constraints of rooting, rendering the applications theoretical. We implemented the prototype system on Android 4.1 commercially called *"Jelly Bean"*.

The server is a python, MySQL database backed application in charge of registering phones and logging intent activity, and which implements our detection algorithm. The system works by allowing phones to register with the system, and sends an activity via HTTP API, thereby logging the intent. Our client is a mobile Android application that provides the ability to register with the server and send it data associated with the device. The client application registers for broadcast and content providers on the device to log the activity associated with SMS messages and detect intent messages of interest from the device. This registration of the application provides us the data on intents usage on the Android OS.

C. Dataset

We collected sample data for model design and evaluation. We built our dataset by collecting separately normal activity data and malicious botnet data, and then merging both datasets.

We collected normal activity data by running our system with 7 separate android mobile phones, each comprising a different OEM release of Android, over the course of 3 weeks.

The bot chosen for the experiments is the *nyaruka* Android relay based SMS controllable application. We adapted it for use in our experiments as a bot which can relay, and SPAM clients from a central SMS control.

To determine the profile of our bot, we installed the bot application along with our intent logging system on an Android 4.1 consumer release Virtual Machine. The bot profile is such that a triggered SMS message will send SMS spam, or respond by sending out a reply to the address that it is indeed listening, and can perform further actions as desired by the botmaster.

We collected in total over the 3 weeks 17,977 activity events consisting of 17,882 non-malicious events and 95 malicious events. We then merged the malicious and non-malicious events in a global dataset using the phone ID of each phone collected to distinguish phones.

V. Intent Detection Model

Our detection model focuses around user behaviour captured by logging and modeling intents. Our model captures several characteristics gained from intent based analysis, which are not provided by traditional traffic only based analysis. We are able to gain contextual information with regard to the action of a user sending or receiving SMS messages, which are not available through a strictly SMS based analysis of messages, or traditional HTTP botnet detection proposed in a desktop scenario. By modeling using intents, we gain insight of not just the specific scenario of interest, but of the general user behaviour on a mobile client, allowing a model to be developed depicting both malicious and non-malicious actions performed by a mobile client, in comparison to known behavioral patterns.

A. Intent Analysis

Our model is based on the density of SMS receipt and response communication on a client, using intent based model. As is found in the conventional Botnets [5], we look at the pattern that develops when a control message is sent from a botmaster to a bot. The bot will respond in a way that triggers a temporally spaced response that differs from a user response. We used 19% of the dataset collected above consisting of 3,385 intent activity events for further analysis in order to build our detection model.

Table 2 outlines the activity details of an event prior to an SMS sent intent. From the data in Table 2, we can see that our sample malicious activity occurs frequently before the SMS sent intent event. The specific listing has a frequency of 22-triggered events out of 73 SMS sent events in our initial testing scenario. Using the fact that the malicious application is detected at such a high frequency before the SMS sent event, we attempt to profile it as part of a generalized malicious signature comprising an initial receive, followed by a send.

Table 2. Pre SMS Sent Intent Listings and Metrics

Pre SMS Sent Intent	Malicious	Frequency
com.nyaruka.androidrelay/com.nyaruka.androidrelay.MainActivity	Yes	22
com.android.contacts/com.android.contacts.activities.PeopleActivity	No	12
com.android.mms/com.android.mms.ui.ConversationList	No	33
com.android.camera/com.android.camera.Camera	No	2
com.android.settings/com.android.settings.SettingsTabActivity	No	1
com.android.phone/com.android.phone.OutgoingCallBroadcaster	No	3
	Total	**73**

We make the assertion since in our analysis we also see that an SMS received is triggered after a SMS sent, which forms a signature of a natural conversation, as well as the signature of

a malicious relay or botnet control signature. We attempt to show a correlation between the intent component in reverse DNS style and the *Action* as listed in Table 3. This action registers information with regard to the intents on the device in malicious and normal user behaviour. We see that an action type of SMS received is of special interest for our research as it is related to the malicious control from a botmaster.

Table 3. Post SMS Received Intent Type List and Metrics

Post SMS Received Intent Action	Malicious	Frequency
android.provider.Telephony.SMS_RECEIVED	No	12
android.intent.action.MAIN	No	19
NULL	Yes	9
NULL	No	7
com.android.phone.EmergencyDialer.DIAL	No	1
android.provider.Telephony.SMS_SENT	Yes	6
android.intent.action.CALL_PRIVILEGED	No	2
	Total	**56**

Based on the data in Table 3 we can see that a simple detection scheme based on the *Received Intent Action* alone would yield incorrect results, due to the cross categorization of Null actions as both malicious and non-malicious.

We move our analysis to a metric surrounding the time difference between SMS sent intents and SMS received intents, based on our logged time. We look at when an intent is received in comparison to when a response is issued through the use of intents. Table 3 summarizes the deviations between the intents send and receive times. Table 4 illustrates a clear difference between an automated botnet style response, and a user crafted response. In fact, the data suggests that in the time domain, standard rules can be used to identify a malicious intent response, in comparison to a non-malicious one with the premise that automated style responses are malicious.

Table 4. SMS Send vs. Receive Time Delta Metrics

Time Delta SMS Send vs Receive	Time (s)
Average Malicious	3.03
Average User Response	1055.76
Median Malicious	3.08
Median User Response	603.55
Standard Deviation Malicious	1.91
Standard Deviation User Response	531.17

We take advantage of the metrics by looking at a standard deviation of 1.9s in a malicious response and 531s in a user response. The data shows a wide variation in the user response, while the malicious responses variability is quite low in comparison. With the average of the malicious responses at 3s and the user responses at 1055s, we can see a large gap that allows a logical separation between the responses to the malicious actions, comparatively to the user initiated intents.

B. Algorithm

Based on sample data analysis conducted in the previous section, we designed an algorithm that is capable of detecting malicious behaviour of an SMS based botnet, in comparison to normal user behaviour. Fig. 1 outlines the proposed algorithm.

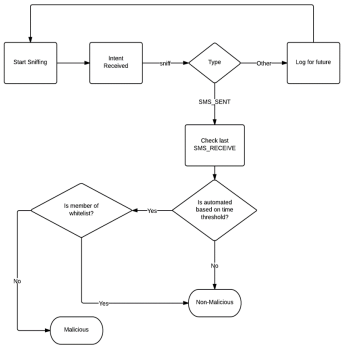

Fig. 1 Intent-based Detection Algorithm

Our algorithm uses a whitelist along with a simple temporal threshold to assess whether SMS activity is automated or not. The whitelist used in the model system consists of a temporal whitelist element. The temporal whitelist deals with actions specifically related to SMS messages such as sending, or viewing SMS messages. This temporal whitelist starts with the following value seeded for our experiment: *com.android.mms/com.android.mms.ui.ConversationList*

The seeded value represents the component observed on Android as the intent used for viewing and sending SMS messages using the SMS messaging application.

The algorithm checks the "Action" type of a received intent to match against the SMS send "Action" type, and focuses on SMS send intent type. Once confirmed as a SMS send, the delay between the SMS send intent and last SMS receive intent is compared against the time threshold. The event will be considered as (intentionally) user generated and classified as non-malicious if the time delta is above the threshold value. Otherwise the event will be considered as automated. The corresponding intent along with all the intents within the time threshold since the last SMS received action will be compared against the whitelist. If any of these intents matches the whitelist, the event will be classified as non-malicious, else it will be considered as malicious.

VI. EXPERIMENTAL EVALUATION

We evaluated the effectiveness of our proposed detector using the remaining portion of our dataset, which consists of 14,592 activity events.

Table 5. Total Intent Events in Test Dataset

Total Events	Total Non-Malicious	Total Malicious
14592	14534	58

Table 4 provides a breakdown of our evaluation dataset in terms of malicious and non-malicious events.

Our algorithm involves a delta time threshold value and a whitelist. The delta time threshold value consists of a time window, which determines if the SMS receive/send pattern was automated in nature. In our experiments, we vary the threshold and assess detection performance accordingly.
We calculate the following performance metrics:

- *False Positive Rate* = # False Positives / #Total Non-malicious Events
- *Detection Rate* = (# Detected Malicious - # False Positive) / # Total Malicious
- *Sensitivity* = # positives / (# positives + # false negatives)

Table 5 lists the obtained performances as the delta time threshold varies. Figure 1 shows the corresponding ROC curve.
The results of our experiments show that we are able to most accurately detect malicious control via SMS on the mobile client when a model threshold of 3.5s is used, while also giving us 0.16% false positive rate and a detection rate of 93%.

Table 6. Performance Metrics as Threshold varies

Threshold (s)	**0.5**	**1**	**1.5**	**2.5**	**3.5**	**5**
False Positive Rate (%)	0.01	0.04	0.04	0.05	0.16	0.07
Detection Rate (%)	0.17	62.07	72.41	84.48	93.33	86.21
Sensitivity (%)	18.64	65.63	75.00	86.15	95.59	88.24

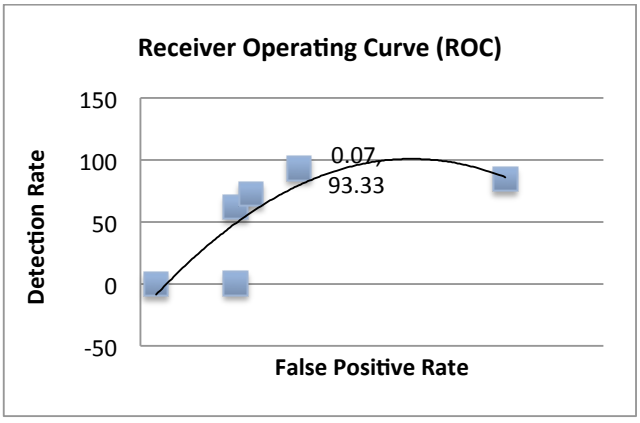

Fig 1.Receiver operating curve of Detection Rate vs. False Positive Rate

We see that a search threshold above 3.5 seconds introduces a greater false positive rate, as well as a detection rate below the 93%, moving in the downward direction towards 86%. Such decrease in performance makes sense since we start to introduce too large a window therefore we start crossing over with other messaging and catching normal user messaging in the system and it would allow the whitelist to cross reference the malicious events.

VII. Conclusion

In this paper, we have described a new detection approach for malicious SMS-based botnet activity on an Android device by modeling intents. We identify key characteristics, and propose an algorithm to detect malicious intents. Our model, and hypothesis were shaped by our experience and through empirical study.
Our model implementation and experiments show that we are able to effectively detect malicious and non-malicious intents, and the activity they represent on an Android device. Experimental evaluation based on data collected from 7 different phones gives a detection rate of 93% and a false positive rate of 0.16%, which are very encouraging.
Our future work consists of improving the detector performance by investigating machine learning techniques and using a more generalized whitelist which focuses on OEM, and core android applications.

References

[1] I. Vural, H. Venter. "Mobile Botnet Detection Using Network Forensics". A.J. Berre et al. (Eds.): FIS 2010, LNCS 6369, pp. 57–67, 2010. Springer-Verlag Berlin Heidelberg 2010.
[2] I. Vural, H. Venter. Detecting Mobile Spam Botnets Using Artificial Immune Systems". G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics VII, IFIP AICT 361, pp. 183–192, 2011. IFIP International Federation for Information Processing 2011.
[3] G. Geng, G. Xu, M. Zhang, Y. Guo, G. Yang, and W. Cui. "The Design of SMS Based Heterogeneous
Mobile Botnet", JOURNAL OF COMPUTERS, VOL. 7, NO. 1, JANUARY 2012, pp. 235-243. Academy Publisher.
[4] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, E. Kirda, *Automatically Generating Models for Botnet Detection*, ESORICS, pages 232–249, 2010
[5] G. Gu, J. Zhang, and W. Lee. *BotSniffer: Detecting botnet command and control channels in network traffic,* in (NDSS'08), 2008
[6] OpenSignal, *Android Fragmentation Report 2013*, July 2013, http://opensignal.com/reports/fragmentation-2013/
[7] Jeff Six, *An In-Depth Introduction to the Android Permission Model,* App Sec Dep 2012
[8] 2015. [Online]. Available: http://krebsonsecurity.com/wp-content/uploads/2013/08/Versafe-SOC-Mobile-attacks-summary-1.pdf. [Accessed: 14- Jul- 2015].
[9] R. Smith, *Dragon Lady: Lookout.com, 'Lookout', 2015. [Online].Available:https://www.lookout.com/resources/reports/dragon-lady. [Accessed: 14- Jul- 2015].*
[10] Developer.android.com, 'Intent | Android Developers', 2015.[Online]. Available:
http://developer.android.com/reference/android/content/Intent.html.
[Accessed: 14- Jul- 2015].
[11] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani and D. Garant, "Botnet Detection based on Traffic Behavior Analysis and Flow Intervals", *Computer & Security,* Elsevier, vol. 39, 2013, pp. 2-16.
[12]Android Operating System, http://ecee.colorado.edu/ecen3000/lecture/introduction.pdf
[Accessed: 14- Jul- 2015].