Detection of Advanced Bots in Smartphones through User Profiling

by

Vishnu Teja Kilari

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2013 by the
Graduate Supervisory Committee:

Guoliang Xue, Chair
Gail-Joon Ahn
Partha Dasgupta

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

This thesis addresses the ever increasing threat of botnets in the smartphone domain and focuses on the Android platform and the botnets using Online Social Networks (OSNs) as Command and Control (C&C) medium. With any botnet, C&C is one of the components on which the survival of botnet depends. Individual bots use the C&C channel to receive commands and send the data. This thesis develops active host based approach for identifying the presence of bot based on the anomalies in the usage patterns of the user before and after the bot is installed on the user smartphone and alerting the user to the presence of the bot. A profile is constructed for each user based on the regular web usage patterns (achieved by intercepting the http(s) traffic) and implementing machine learning techniques to continuously learn the user's behavior and changes in the behavior and all the while looking for any anomalies in the user behavior above a threshold which will cause the user to be notified of the anomalous traffic.

A prototype bot which uses OSN s as C&C channel is constructed and used for testing. Users are given smartphones(Nexus 4 and Galaxy Nexus) running Application proxy which intercepts http(s) traffic and relay it to a server which uses the traffic and constructs the model for a particular user and look for any signs of anomalies. This approach lays the groundwork for the future host-based counter measures for smartphone botnets using OSN s as C&C channel.

DEDICATION

To my parents and all those people
who made my life special.

ACKNOWLEDGEMENTS

I would like to express my most sincere thanks and gratitude to my advisor Dr. Guoliang Xue, who has guided me throughout my graduate study and providing me the freedom and support to conduct this research. Working with him has been one of the most enriching experience both professionally and personally.

I am also grateful to my thesis committee members, Dr. Partha Dasgupta and Dr. Gail-Joon Ahn for their valuable comments, suggestions and being on my thesis committee.

I would like to thank Lingjun, my colleague, for his invaluable inputs. I am also thankful to Arun at Information Assurance lab for his immense help during the course of my research. I would like to thank entire group working with Dr. Xue who have made it such a great place to work. I am especially grateful to Dejun Yang and Xi Fang who have made me feel home when I first started working with Dr. Xue.

I would like to express my gratitude to Michael Sullivan and all staff members at Hispanic Research Center for their help and support.

I am thankful for my roommates Edberg, Vipin, Ravi, Sandeep, Vignesh and Yashashvi for their encouragement. I am also thankful to all my faculty, friends and seniors at Pondicherry Engineering College for making me who I am today.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Current day to day operations of individuals and enterprises heavily rely on the individual owned smart devices (smart phones and tablets). This growing reliance on smart devices (owing partly to BYOD trend) results in storage of confidential data on the smart devices. While the sensitivity of the data being stored in the smart devices is increasing at alarming rate, user awareness of security in the smart devices is not increasing in comparable rate. The rate of penetration of these smartphones is making these devices a lucrative target for cyber criminals. To put things into perspective, as of August 2013 there are 1.39 billion smartphone users with Android claiming 58 % market share and 84 % of users use smartphones to browse internet exposing them to nefarious hackers. Smart devices are treasure trove for malicious hackers who pilfer the smart devices for valuable data (both private and enterprise).

One of the important tools for this pilfering in malicious hacker's arsenal is botnet. Botnet consists of bots which receives commands through Command and Control channel, executes them and in some cases upload the data back through the same channels. Botnets differ from other classes of malware in the area of tasks performed by the malware. Almost all other classes of malware will be preprogrammed to do specific task(s) at specific time. Botnets can perform a variety of tasks based on the commands received and control they have over the host. This research fits into mobile security domain specifically focusing on the host based detection of bots.

1.2 Problem Statement

Botnets are one of the major problems being encountered today by the cyber community. This is due to huge number of infected machines and the diversity of tasks they can be commanded to use. Another major problem is the difficulty to pinpoint the source of attack which might help the defenders to organize a counter attack. Botnets have been the cause of spam, DDOS attacks as well as the illegal information collection. Sheer number of smartphones and their ubiquitous connectivity to the internet have made the current day smartphones particular target for the botnet masters. Many researchers tried to address the problem of detecting and disassembling botnets by detecting the botnet traffic among the general network traffic at ISP level or network level [2,3,4] and eventually identifying and targeting the command and control (C&C) servers operating a particular botnet. This approach leads to destruction of botnet but leaves the bots and the information collected by bot on each device intact to be exploited in the future. Some researchers considered a scenario in which C&C servers cannot be taken down even when the botnet traffic is identified. The scenario is as follows: Suppose the bot master (one who is behind the botnet) decided to leverage the Online Social Networks (OSNs) as the C&C channel. Since the OSNs cannot be taken down, even if the botnet traffic is detected at the ISP, the C&C channel cannot be broken. Leveraging OSNs as C&C channel might simply involve creating a user and posting a tweet (in case of Twitter) or a post (in case of Facebook). So, even if the user is deleted by the authorities, the bot master will simply create another user. This scenario can be only handled when the detection of bot is done at the host level so that user can be notified of its existence and take remedial actions. More research is required to study and develop the measures for detection of bots at host level.

1.3 Research Objectives

The Command and Control component is vital for existence of any botnet because it is the component that connects the bot master with the bots. Bot master needs to design a resilient, secure and robust way to stay connected with the bot. This results in bot master leveraging the Online Social Networks as the Command and Control channel. This author is unaware of any host based detection methods for detecting the advanced bots that leverage OSNs as the C&C channel at the time of this writing. This thesis aims to provide a ground work for detecting the bots using OSNs as C&C in the smartphones. The research goals are as follows:

- Design a mechanism to detect potential C&C channel selection and communication and bot presence and alert the user to the same
- Implement the mechanism as a proof of concept on Android platform
- Validate the proof of concept using a prototype of advanced android bot

1.4 Approach

The goals of this research are attained by developing a classifier that takes the user web data as input and create a model based on the user web behavior. This classifier further classifies the incoming data as regular usage or anomaly based on the model developed. If an anomaly is discovered, then user is notified of the anomaly and is expected to take remedial action. The classifier implemented is Kernel Logistic Regression. The bot prototype implemented is similar to "mini-duke" bot in terms of Command & Control which leverages OSN s. This bot prototype is implemented on Android platform. The user data belongs to the users operating a Nexus 4 and a Galaxy Nexus smartphones. Fiddler is used as the application proxy to intercept the data. Daily web usage of these two users is captured by the server. After preprocessing it, the resultant data is submitted to the classifier.

1.5 Assumptions and Limitations

Several key assumptions and limitations were accepted in realizing the goals. The assumptions are:

- Smartphone is not infected during the first 15 days of usage (training phase).
- Adversary does not have knowledge of this system
- OSNs are the only channel for Command & Control
- Successful vectors exist for infecting a smartphone with a bot

The limitations are as follows:

- The system only provides awareness regarding the existence of bot in the smart phone but does not provide eradication mechanism
- The proxy is implemented as Application proxy based on inserting a root certificate in the system trusted store of certificates. Due to this, in depth analysis of traffic belonging to applications which use certificate pinning is difficult. But there are very few apps that use certificate pinning.
- The external server is assumed to be a trusted server

1.6 .Organization

The rest of this thesis is organized as follows: Chapter 2 is a literature review summarizing the botnet phenomenon giving overview of botnets, various types of botnets based on Command and Control mechanisms, various botnet and bot detection methods already proposed. In Chapter 3, we discuss advanced botnets in the smartphone platform and briefly discuss about the next generation botnets. Chapter 4 includes the implementation details of the next generation bots and our proof of concept of bot. Chapter 5 presents our detection system, the implementation details and system evaluation results. Chapter 6 summarizes the research, discusses advanced bots in smartphones and suggest areas for extended and future research regarding the detection of these advanced bots.

Chapter 2

LITERATURE REVIEW

In this chapter, we provide a background for different components involved in the proposed host based mechanism to detect the bot that leverages OSN s as the Command & Control channel. First, the functioning of botnets is discussed to understand how they work and then various Command & Control mechanisms used over the time and the counter techniques developed by the researchers to mitigate the botnets are discussed. Next, an explanation of advanced botnet leveraging the Online Social Networks is provided.

2.1 Overview of Botnets

Botnets are one of the widely used malware to inflict various damages on the cyber community like DDOS attacks, spam, spamdexing, click fraud and very recently mining bit coins. The following section describe botnets.

Definition & Components

A bot is defined as a script/code designed to automate some predefined functions. A spider/crawler used by search engines can be termed as a bot because it is designed to automate the functionality of crawling the web and indexing the pages along the way. Bots are used to do automated tasks such as training the game personas in an automated way without human intervention and thus are very useful to save a lot of time. In the time of IRC networks, these bots were used to automate some administrator functions. It was in these IRC networks that bots were first modified to create a network by linking one bot to other leading to botnet.

In modern days, the term “bot” is being used to refer to malware residing in a computer awaiting commands from the bot master (designer of botnet). It is different from other malware because it is not waiting to be executed (virus) or spreading to other machines (worm) but it allows attacker to take full control of the machine. It is also different from

other classes of malware in terms of functions it is deigned to perform. While others have predefined functionality and corresponding code in them, bot can execute a variety of commands as specified by the bot master and thus can wreak a havoc when compared to other classes of malware. Since, it is dynamic in the sense of executing instructions, it is very difficult to identify them based on the functioning. A botnet is collection of bots controlled by an attacker or a group of attackers with ability to communicate and execute commands.

The bot master needs a mechanism to communicate with the botnet to specify the commands bot needs to execute in their host machines and to send any data to the bot master. This mechanism is known as Command and Control channel (C&C). C&C channel is one of the most vital component in a botnet because it is the only means connecting the bot master and the bots. Design and characteristics of this C&C channel determine the resiliency, stealth and robustness of the botnet.

Botnet Lifecycle

Lifecycle of Botnets consists of four phases [5]. 1. Spread 2. Infection 3. Command & Control 4. Attack. Bot master first delivers the malware payload to target machines which then infects the machine. The Command & Control phase begins after the machine has been infected. The attack phase will start after the bot masters has a large enough number of bots under his control. Let us discuss each of these phases in detail.

Spread: Bots can spread through various means such as phishing emails, compromised websites, worms and malicious pages links sent by spam. Phishing emails are emails designed to look like generic emails with embedded html links in them which point to malicious websites or with a downloadable file like pdf, doc, docx etc., When user downloads and opens these files, the files utilize the vulnerabilities of the software to perform malicious tasks. Worms are malware which are programmed to propagate to other machines either by finding vulnerable accounts on the networks or vulnerable operating

systems. The modern day vectors of botnet spread particularly in the case of smartphones are malicious apps from third-party application market places and drive by downloads. The main purpose of this phase is to just deliver the payload that eventually will infect the machine. This infection might be done by unwitting users executing the malware or the malware exploiting the system or software vulnerabilities. There is a huge delay in delivery of security patches to the Android smartphones by the OEMs and the network carriers. This delay from the time of discovery of a vulnerability to the time of issuance of a patch is a powerful motivation for the attackers to attack the Android platform.

Infection: Once the payload is delivered to the target machine, the malware uses various methods to infect the machine. The purpose of this phase is to infect the machine and maintain a presence inside the machine without attracting any attention. In order to achieve this, the malware uses a wide range of techniques like disabling the anti-malware tools, Polymorphism, Code hardening and root-kitting.

Polymorphism: Anti-virus engines normally keep track of the viruses by maintaining virus signature database containing information such as name, code size and code hash. In order to fool the Anti-malware engines, the malware changes its code with each infection. This makes it harder for the malware engines to detect the code.

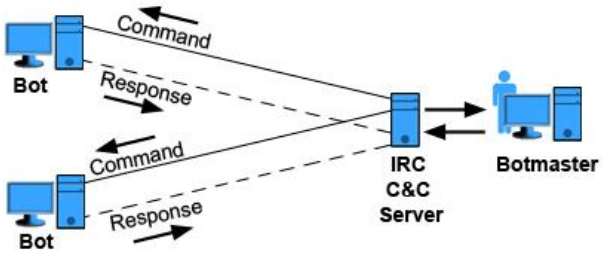
Code hardening: In order to prevent reverse engineering of the bot which might result in revealing the Command & Control channel to others, bots are being designed today to withstand reverse engineering. In order to achieve this, bot masters are using techniques like code obfuscation, encryption and encoding to prevent it from being analyzed.

Rootkits: Rootkits are the most powerful of all these methods because the rootkits are installed with root privileges. Since rootkits are installed with root privileges, they can hide all their activities from the Operating System and thus from the anti-malware engines.

Command & Control: Command & Control phase involves bot communicating with the bot master and receive further commands to be executed on the target machines. The communication between the bots and bot master is done through Command & Control

servers. These servers will issue commands and in some cases receive any data uploaded by the bots as instructed. There are various protocols which are in use by these Command & Control servers and the botnets some times are classified according to the type of protocol used. We will describe the types of botnets in following sections but for now we will discuss the types of protocols in use.

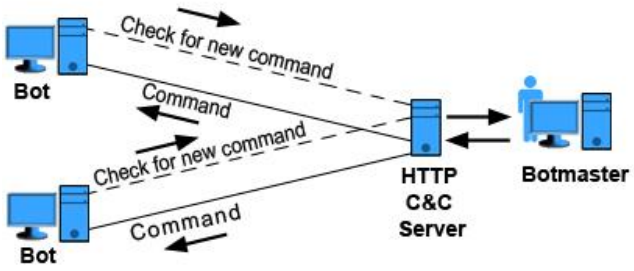
IRC: In the initial stages, Internet Relay Chat (IRC) has been the most common protocol which has been used for Command & Control. Each bot connects to the IRC server mentioned in the body of the bot and awaits commands from the bot master on certain channel in a PUSH style. The disadvantage with this method is that taking down the IRC server or channel will result in dissolution of botnet.



2.1 Bot communication over IRC channel

HTTP: One of the ways of detecting the Command & Control traffic is by observing the open ports in the system and amount of IRC traffic or any other anomalous protocol traffic. In order to have a stealthy C&C channel, bot masters started leveraging the popular HTTP protocol which is used to render the internet pages. Since almost all computers connected to internet have HTTP traffic, detecting the C&C traffic in HTTP traffic is difficult. Instead of being conspicuous by connecting to several HTTP servers and remaining in connected mode and leading to lot of connected ports in the machine which might raise any red flags,

the HTTP bots employed PULL style in which the bots periodically visit certain web servers in order to get new commands or upload any data. This mode of communication will be inconspicuous. This protocol also suffers from central point of failure. If anyone were to decode the servers that were being used as C&C servers, taking down those servers will result in successful disruption of botnet. But there are several advantages by using this protocol as well. Since HTTP traffic is used to mask the C&C traffic, IDS and firewalls will have difficult time spotting it. While it is possible to detect the C&C traffic through whitelisting or blacklisting the websites, it gets even more difficult to stop these kind of botnets if the HTTP servers were actually compromised trusted servers.



2.2 Bot communication over HTTP protocol

Short Message Service: Advent of smartphones have introduced Short Message Service as one of the C&C channels. This is due to the fact that the smartphones are programmable. Even though SMS can be leveraged as C&C channel in feature phones, the bot in the feature phone will have a very limited use such as sending SMS to premium numbers. But, When SMS is leveraged as C&C channel in smartphones, it can serve as a powerful alternative to above mentioned protocols.

Attack: When the bot master has established Command & Control over the sufficient number of bots, he is ready to utilize the full potential of bots under his command.

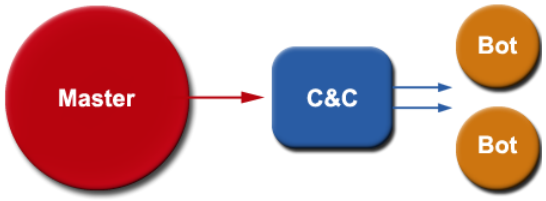
Traditionally, botnets have been used to send spam, phishing mails and perform click fraud as well as massive Distributed Denial of Service (DDOS) attacks against variety of targets such as banking and government networks. Phishing mails will further increase the size of the botnet. Click fraud will result in poisoning of Search Engine results which in turn will cause users to be redirected to malicious sites. DDOS attacks are one of the most expensive cyber-attacks. A DDOS attacks will result in huge monetary and strategic loss and is equally costly to defend. Some botnets perform brute force login attempts on vulnerable network accounts leading to Identity theft and further compromise of the networks.

2.2 Taxonomy of Botnets

Botnets are classified on the basis of the architecture used by the bot master to control the botnet. There are traditionally two types of botnets [6]. They are as follows:

- Centralized botnets and
- Peer-to-peer botnets

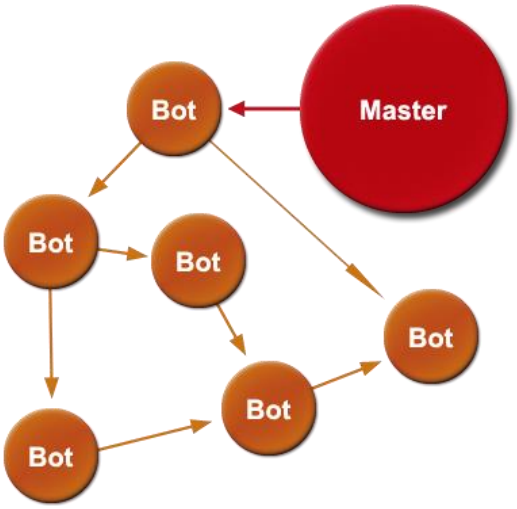
Centralized botnets: In this type of botnets, Command & Control server(s) is (are) centralized. This central server waits for the bots to connect to it and then register the bot, updates them with the commands and keeps track of the status of each bot. The centralized server knows all the bots and is in contact with all the bots in the botnet.



2.3 Centralized botnet architecture

Several advantages exists for botnets with this kind of architecture. It is easy to manage the botnets and commands are dispersed faster and easier. Designing this type of bot is also easier. One of the disadvantages of this architecture is its central point of failure. If bot is reverse engineered and the C&C servers are targeted, the botnet will be disrupted.

Peer-to-peer botnets: In order to overcome the inherent disadvantage of the centralized botnets, peer-to-peer botnets are designed. In this architecture, each bot connects to other infected machines. Each bot is supplied with a list of machines it is supposed to be connected to. This bot is responsible for getting commands from one of them and transmitting them to rest of them. Bot master just needs to have control of one bot to control the botnet. Although disrupting this kind of botnets is difficult, designing and managing this kind of botnets is difficult for the bot master. In order to make the design easier, the bot first contacts centralized server if present and gets a list of bots it is supposed to be connected to and then receives and disseminates the commands based on the peer-to-peer protocol. This will make the botnet resilient because there exists no central server in peer-to-peer to deactivate. Some research has been done to explore mechanisms to identify and disrupt peer-to-peer botnets.



2.4 Peer-to-peer botnet architecture

2.3 Botnet Defense Techniques

Blacklisting: Blacklisting involves prohibiting the machines inside a network to connect to particular domain(s) or IP(s). Command & Control traffic of the botnet is monitored and all the domains and IP addresses it is connecting to are placed in black list. Each time a machine wants to initiate a connection, a look up is performed on the blacklist and if the domain or IP it is connecting to is present in the blacklist, the connection is terminated. If not, then the connection is allowed. This prevents the machines from contacting the known malicious domains which serve as C&C servers. This method is widely used to disrupt HTTP based botnets

Port blocking: This mechanism is widely used to block IRC botnets. Since bots operating IRC protocol should use a network socket to connect to the IRC servers, firewalls are programmed to block well know IRC ports. This will prevent the IRC bots from making connection to their C&C server.

DNS sinkhole: DNS sink holing [7] cuts off the bots from communicating with Command and Control servers by utilizing the Domain Name Service (DNS). DNS is a service used to resolve the human readable names to IP addresses. DNS sinkhole works by spoofing the authoritative DNS servers for malicious hosts and domains. DNS sink holing involves configuring the DNS forwarder to return false IP addresses for known malicious hosts and domains. So, when a bot requests to resolve such host or domain, it is routed to a non-existent address or a benign host from which the bot cannot retrieve any commands. This approach is particularly effective against centralized botnets due to the fact that there is a single point of failure. If the bots cannot resolve the host, then the C&C server does not have any means of disseminating the commands.

Botnet infiltration: The above methods work fine for majority cases of centralized botnets and some minor peer-to-peer botnets. Botnet infiltration is one of the effective ways to disrupt peer-to-peer botnets. This involves exploiting the bootstrapping process of peer-to-peer botnets. In order to connect and receive instructions initially, the bot (peer-to-peer or

centralized) must have some information on mechanisms to establish Command and Control channel and to download the instructions. This information might include IP addresses of bots to be connected to initially, protocols of communication and connection information. Peer-to-peer botnets are vulnerable to unauthenticated publish systems, any bot can publish commands and if the sufficient number of benign bots flood the botnet peer-to-peer network with benign commands, then they overwrite the bot master commands and thus succeed in disrupting the botnet.

Index poisoning: Idea of index poisoning [8] is to introduce massive number of bogus records into the index so that if one of the peer receives a bogus record, the peer would not be able to locate the peer or connect to the wrong peer. If a bot is reverse engineered and bot command related index keys are compromised, then "Index-based" peer-to-peer botnets are vulnerable to this attack. Since there is no central authority to manage the file index, malicious nodes will be able to insert records into the index, and there is no way to authenticate the identity of the node and content of the records.

Sybil defense: A Sybil attack [8] is the forging of multiple identities to subvert the reputation system upon which the peer-to-peer network is built. Peer-to-peer networks normally operate without any kind of authentication or validation and hence any peer can join the network. If index keys were compromised, Sybil nodes can be added to the botnet to re-route or monitor the command related traffic.

2.4 Botnet Design Techniques

In order to cope up with the techniques evolved into disrupting the botnets, bot masters have come up with variety of techniques to make botnet as resilient and as stealthy as possible. We discuss some of the techniques used by the bot master to design the botnet to be resilient.

Fast Fluxing:

Fast fluxing [9] is assigning multiple IP addresses to a domain name (like

www.xample.com). The IP addresses are assigned and de-assigned to the domain name very frequently using round robin algorithm. There are single flux networks and double flux networks in use. Single flux networks involve front end bots to which traffic is redirected to. If the end user browser likes to communicate with the www.xample.com, then actual communication will take place with one of the many infected hosts whose IP addresses are used in round robin fashion to be assigned to the DNS record of the domain www.xample.com. In some cases, these front end bots will communicate with the back end servers to deliver the commands or upload the data. This method will ensure the load distribution across the infected hosts. Double flux networks will involve the above mentioned approach with some changes. In the double flux networks, the DNS name server itself will be a part of the flux scheme with its IP address changing frequently. This name server will contact the back end servers to service the user query of the domain name and will deliver the IP address of an infected host which is a part of the single flux network. There are several advantages with the flux networks. One of them is that the front end hosts are disposable. A security researcher tracking the botnet might locate some of the infected hosts which are front end hosts involved in the fluxing. Thus, the backend server is protected by the layer of disposable front end hosts. Double flux networks increase the resilience of the botnet in the same method adding a extra layer of protection for the backend servers by involving the name servers in the flux scheme.

Domain Generated Algorithm (DGA) based Botnets:

Fast flux technique mentioned above can be otherwise call IP fluxing technique because IP addresses are fluxed to standard domain name. Domain Generation Algorithm (DGA) can be called as Domain fluxing technique. In this technique, the bot will have an algorithm which will take a seed value (like current system date and/or time) and output a list of random names. These random names will serve as domain names to which the bot will attempt a connection. Each bot will execute this DGA periodically and will try to resolve these domain names by sending DNS queries until one of them directs to a IP address of a

Command and Control server. This technique is remarkably resilient to botnet disruption techniques because it combines the advantages of Centralized and peer-to-peer botnets. Even if one or more Command and Control domains are identified and removed, the IP address of the Command and Control server can be assigned to one of the Domain names in the next set of domain names provided by the Domain Generation Algorithm. This technique has some disadvantages like it is time and cost consuming to register domain names to IP addresses of C&C servers every time a domain is taken down. And if the DGA in the bot is reverse engineered, then the past, current and future candidates of the DGA can be computed and blocked to cut off the communication from the bot to C&C servers. This can be made harder by techniques such as code-obfuscation and encryption. But the cost effectiveness is still an issue. Some researchers have suggested a method of detecting DGA based malware without reverse engineering the bot. Since botnets employing this technique require bots to query DNS servers for valid domain names resolving to IP addresses of C&C servers, by monitoring the NXDomain traffic [10] (Non-existent Domains) which will result for the domains that does not resolve to any IP, the researchers aim to employ clustering algorithms to cluster the similar NXDomain traffic generated by bots of the botnet.

Chapter 3

BOTNETS IN SMARTPHONES

Traditionally bot masters leverages various vectors like Spam email, Phishing email, Viruses, Worms and malicious web pages as the vectors for the infections of hosts to become bots. However, as the smartphones evolve and the data being stored on the smartphone becomes sensitive and personal day by day, the malware writers have shifted their focus towards mobile platforms. There are various reasons for this phenomenon. One of them is that current day smartphones are not just personal devices, they are computing machines too. Current day smartphones pack a processor (more generally dual core), separate Graphics co-processor, at least 1 GB RAM and a variety of sensors (and a processor sometimes to monitor them, like motion co-processor in iPhone 5S). The variety of sensors in the current day smartphones provide a unique opportunity to malware writers that traditional computing platform was not able to provide.

The sensors package in the current day smartphone has the ability to collect a variety of personal information like the locations visited by the user of the smartphone from which user location patterns can be deduced (GPS). The other sensors like accelerometer, gyroscope and magnetometer provide data that cannot be gathered from traditional computing system. The data from this sensor rich mobile computing platform made this platform a potential target for malware writers. Another main reason for this platform to become a target for malware writers is that its ubiquitous connectivity with the internet. Smartphones these days are connected to internet through wide variety of means such as 3G/4G, Wi-Fi etc.,

Botnets for smartphones are lucrative for malware writers because of the ubiquitous connectivity, smartphone adoption rate, lack of central security authority and lax security policies being implemented in this nascent platform.

3.1 Infection vectors and Command & Control Channels in Smartphones

The modes of connectivity of the smartphone has given rise to new infection vectors as well as Command and Control channels in the smartphones such as Short Message Service (SMS), Bluetooth, Application market places, drive by downloads and MMS. Short Message Service is one of the most coveted channel as the infection vector and the Command and Control channel due to its ubiquitous nature. It is present by default in every phone and the SMSs are routed through the carrier internal network thus restricting its exposure to tracking and detection. Bluetooth is another new vector for infection and C&C channel. It is generally thought that Bluetooth is not ideal vector C&C channel due to its low range. But recent research conducted to evaluate the Bluetooth as C&C channel [11] in smartphones showed that due to regular habitual patterns followed by humans enable Bluetooth to be a good C&C channel. For example, if a smartphone user use subway at 9 AM and 6 PM then the bot can send and receive commands from other neighbor bots from the co passengers of the subway at 9 AM and 6 PM. Application market places also contribute to the infection of smartphones due to lack of inspection mechanism. Malware writers combine famous and functional apps with malware and upload them to application market places. Google Play store, Apple app store and Amazon app store are some of the major application markets for smartphone applications. Apple app store performs checking on apps for malicious features while Google Play store has recently employed static checking mechanism for apps submitted to it. While these mechanisms prevent some of the malicious apps to be uploaded to the app store, both these app stores are vulnerable to apps that contain malicious code in separate places and combine the code while running of the app. And many countries involves local application market places in which no checking is performed on the apps uploaded. These application market places will serve as main infection vectors along with drive by downloads.

3.2 Advanced Bots in Smartphones

This section deals with advanced botnets [12] in smartphones in terms of infection vectors, resiliency, stealth, primary functionality and the sheer number of devices infected.

Zeus in the Mobile:

This is the mobile component of the famous Zeus botnet known for fraudulent banking transactions by collecting the user banking credentials. As the authentication mechanisms evolved, many organizations shifted to what is popularly called “two factor authentication” mechanism. This mechanism involves the knowledge of some information (password) and the possession of the something (in most cases, a registered mobile device to receive a code which should be inputted as second layer of authentication). This mechanism is based on fundamental security primitives of “what you know” and “what you have” which are building blocks of other authentication systems such as hardware tokens (RSA SecurId), dongles etc., The smartphone implementation of the system has an inherent flaw in it compared to the USB tokens. While the sole functionality of the hardware dongles is to provide the random authentication code to access the second layer, smartphones have various functionalities apart from that. So, when organizations started to utilize smartphones for two-factor authentication, the malware writers began targeting the smartphone platforms for the secondary authentication tokens being sent to the smartphones. The Zeus in the Mobile or Zitmo is developed for this purpose. It infects variety of mobile operating systems, such as Symbian, Windows Mobile, BlackBerry, and Android mainly by social engineering approaches. It sends an infected SMS to victims contain a fake URL to dupe users to download a security certificate which is, in fact, the Zitmo bot. It also intercepts messages which are sent by banks to customers and authenticates illegal transactions by stealing mobile Transaction Authentication Numbers (TAC).

AnserverBot:

AnserverBot is considered one of the most sophisticated malware found in the smartphones. Its sophistication lies in its Command and Control mechanism based on a two-layer mechanism and implemented over public blog. This mechanism makes it resilient to botnet takedown techniques and makes it harder to identify the botmaster who is covered by two layers. Its additional capabilities include detecting and disabling the security solutions in infected device which is normally an attribute of PC botnets. In order to prevent reverse engineering, the AnserverBot periodically checks its signature to verify its integrity in order to protect itself from any type of changes.

DroidDream:

DroidDream was one of a those kind of bots which can be termed intelligent, since it is activated silently and at night (11pm to 8 am) when the usage of the user is minimal so as to prevent any changes in battery usage and CPU processor cycles visible to the users. It was designed to gain root privileges on infected mobiles and install a second application to steal sensitive information and protect itself from removal.

Ikee.B:

Ikee.B is one of the advanced botnets in smartphones due to its infection mechanism. It targets jail broken iPhones and when it infects a smartphone, it scans the IP range of iPhone networks, looking for other vulnerable iPhones in global scale.

3.3 Next Generation Botnets in Smartphones

Design of these next generation botnets involves designing a stealthy command & Control channel, making detection harder and prevent botnet hijacking and take down of botnet as hard as possible even in case of complete reverse engineering of the bot and control of critical resources by the defenders trying to hijack or take down the botnet. These next generation botnets involve leveraging Online Social Networks or public blog services as the Command & Control channel to disseminate commands. Since, these Online Social

Networks and public blog services are neither monetarily inhibiting nor require any kind of identity to be provided and is hard to monitor, they can be effectively used as Command & Control channels. Another advantage of using these channels as C&C channels is camouflage. It is difficult to detect the traffic of these platforms as Command & Control channel traffic due to their pervasiveness and popularity.

In order to counter this type of botnets, we ought to develop the mitigating strategies that try to take down the botnet at its lowest level, individual bot. Taking down a botnet has involved finding the source of Command & Control channel and trying to cut off the Command & Control channel, thus isolation the individual bots without any means of connection with the bot master to accept the commands and thus do any nefarious activities. But this approach might not work with these next generation botnets whose Command & Control channel might be impossible to take down. Generally, large ISP s try to identify the Command & Control channel traffic in the total traffic that passes through them by various means. Many researchers have proposed various mechanisms that try to detect anomalous traffic of Command & Control in normal traffic [13]. These methods involve clustering of similar type of traffic which might be anomalous in nature with respected to general traffic. So, instead of looking at the source of Command & Control channel to take down or hijack the botnet, we look to the existence of bots in the devices.

Chapter 4

METHODOLOGY

In this chapter, we discuss the design details of a next generation botnet, our approach to detect such bots and our implementation details regarding the prototype of such bot and our method.

4.1 Design of Next Generation Botnet

Our next generation botnet design details are as follows. The bot uses centralized C&C topology. The bot contains a hard coded public key and a Username Generation Algorithm [14]. This Username Generation Algorithm is derived from the Domain Generation Algorithm mentioned above. In the domain flux, Domain Generation Algorithm takes a seed (system date and/or time) and generates a list of domains and the bot attempts to connect to each one of them until it connects to a domain. Once it is connected to a domain, it downloads the latest commands and tries to verify their authenticity using the public key hardcoded in it. Username Generation Algorithm differs slightly from it. Username Generation Algorithm generates a number of random names which are appended to a list of Online Social networks, blog posts. For example, let us consider "Werdhfljcm" is one of the names outputted by the Username Generation Algorithm. This username is now appended to a list of Social Online Networks. Let us consider that list of social networks contain 1. www.facebook.com 2. www.twitter.com 3. www.baidu.com etc. Now, the bot appends the username to all these sites resulting in 3 possible candidates for communication. 1. www.facebook.com/Werdhfljcm 2. www.twitter.com/Werdhfljcm 3. www.Baidu.com/Werdhfljcm . The bot checks for presence of these users and if they exist, it will download the latest tweets (in case of twitter) or cover pictures or profile pictures (in case of Facebook) or posts (in case of Baidu). The important reason in downloading the specific information is that it is public. One does not need to login to obtain the above

mentioned information (tweets, cover/profile pictures, posts) from the Online Social Networks or blog posts. After downloading this publicly available information from the user profile of one of the users from the list of users generated by Username Generation Algorithm from one of the Online Social Networks or blogs, the bot proceeds to authenticate the information through the means of the hard coded public key. It is obvious in the case of tweets and posts. In case of pictures, attackers leverage the concept of "Steganography" in hiding the commands in the pictures. Once, the command is authenticated, the bot proceeds to execute the instruction. If the command needs the data to be uploaded, it then can include the instructions for the upload.

The bot master who created the bot needs to create just one user in the users list generated by the Username Generation Algorithm in one of the social networks, and post the commands in that user publicly available information signed with his/her private key.

4.2 Prototype

Our implementation of this next generation bot involves only one Online Social Network (OSN), Twitter. Our username generation algorithm is derived from the following Domain name Generation Algorithm (DGA).

```
for i in range(16):
    year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
    month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
    day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
    domain += chr(((year ^ month ^ day) % 25) + 97)

 return username
```

Pseudo code for Username Generation Algorithm [15]

The bot will append the usernames generated by this algorithm to the OSN (in our case www.twitter.com/) and checks the OSN for the existence of the user (of username generated by DGA) and if user exists, the bot retrieves the first tweet of the user and authenticates the tweet using the public key hardcoded in the bot.

Chapter 5

DETECTION MECHANISM

Our detection mechanism of the above mentioned type next generation bot in the smartphone platform involves modelling the user web behavior and detecting any anomalies that might be caused by the presence of the bot. This detection mechanism involves collecting the user traffic (training data) for a certain amount of time (training time), modelling the user behavior based on the traffic collected using a classifier and then comparing the traffic generated on the subsequent days to the model already constructed on the user behavior. If the traffic does not match with the model already constructed above a threshold, the user will be notified of the anomalous traffic. If the user approves the traffic then this traffic will be considered as benign traffic and will be used to update the model.

User data collection*:*

We have considered two modes of traffic collection. One of them is the application level proxy and the other one being the "SSL hooking". At the end we chose Application Level proxy. The reasons for choosing Application Level proxy and not SSL hooking are detailed below.

SSL Hooking:

SSL in android is implemented using OpenSSL library. OpenSSL is a open source toolkit to implement SSL version. Android makes calls to the OpenSSL library functions to implement SSL. SSL implementation in android works as follows. Suppose a browser executes HTTPS request for www.google.com. In order to encrypt the data (i.e. , www.google.com), the "ssl_write" function of the OpenSSL library is called. The data www.google.com is written to memory and the pointer to that memory is passed to the "ssl write" along with the key negotiated by the protocol and the mode of encryption to be used. During the decryption process, similar method is followed. The encrypted version of the data will be written to a memory location and the library method "ssl_read" will be called

with this memory pointer and the respective key. The “ssl_read” will decrypt the data. So, “SSL hooking” involves hooking the calls to “ssl_read” and/or “ssl_write” functions to read/write the data when it is in clear text.

Run time hooking in windows involves hooking DLL files through various means such as IDT/SSDT hooking. Hooking function calls in Linux is normally done through the “LD_PRELOAD” function. This “LD_PRELOAD” instructs the .so (shared object) files to be loaded from the path “LD_PRELOAD” points to before loading any library. So, if LD_PRELOAD were to point to a directory containing our version of “ssl_read” and/or “ssl_write”, our “ssl_read” and/or “ssl_write” will be executed instead of original functions. Our functions can copy the clear text data and then redirect to the original functions which will perform the intended duty.

According to our knowledge, “SSL hooking” on android involves either building a custom rom or repackaging the applications which are installed on the smartphone. This is due to the fact that Android Operating System does not include “LD_PRELOAD” function. Although Android OS is derived from Linux, many parts of Linux are stripped away to reduce the size of the Operating System. “LD_PRELOAD” is one of them.

We tried to hook the android .so files without repackaging the individual applications and/or without building a custom ROM. But, the Android Operating System would not permit any such hooking as it is viewed as a potential breach. And, the data collected by our ssl_read/ssl_write functions will be huge. And also, the Application proxy provides us with a variety of information which we can make use of when trying to model the user web behavior. So, we used an Application Proxy (Fiddler) to collect the data from the user. This can be done by modifying the network connection settings of a particular network in the Android settings. Application proxy also intercepts HTTPS traffic by installing a root certificate in Android’s system trusted store.

5.1 Modelling the User Behavior

We extract some metrics from the user data to model the user behavior. The metrics are

1. Hostname
2. Application name
3. Language encoding
4. Randomness of the request (entropy)
5. Top Level Domain (TLD)
6. Time of the request

We design the features for our classifier based on the above metrics.

Our modelling mechanism involves two phases. First phase involves modelling the semantics of the user web usage. It consists of first dividing the user requests on per domain basis. For every domain, we keep track of the requests made to that particular domain. We construct a separate model for each domain based on the user behavior. After getting the test data from each user, we divide the data based on domain and create models for each domain. In the testing phase, we divide the test data based on the domain name and match each domain traffic with the model constructed for that domain during the training phase and assign anomaly weights based on the level of matching. We do this for all the domains.

Second phase involves calculating the frequency of requests made per domain per day by the user. During the training phase, we keep track of the time of requests. We divide the day into 24 cycles. We calculate the number of requests made to a domain per cycle and the total number of requests made to a domain per day. We calculate this frequency of traffic per domain for all the days during the training phase and construct a model based on the frequency of the traffic. During the testing phase, we compare the frequency of the traffic per domain per cycle to the already constructed model and assign an anomaly weight to the extent of matching.

Finally, we sum both the anomaly scores and if the total score crosses a threshold, then the domain traffic will be reported to the user along with the apps that are making the

requests to the domain. The user will be prompted to classify the traffic as benign or malicious. If the traffic is classified as benign, then it is used to update the model.

The features used above are very useful in detecting the bot traffic. The significance of each feature is explained below

- *Host name*: host name is used to separate the traffic directed at a particular host from others. This is useful in particular because the bot generates anomalous traffic (C&C traffic) to different domains. These domains are identified by the host name.
- *Application name*: This is a very useful particularly in smart phones. In smartphones, each application will be mostly associated with a domain (in other words, host name). For example, almost all the traffic generated to New York Times (www.nytimes.com) will come from the New York Times application installed on the smartphone. This association of the domain name with the application is useful to differentiate traffic from other applications to this domain. While it is not always true that majority of the traffic to a domain will come from a application, it is true in majority of the cases.
- *Language encoding*: It is a very useful feature along with the Top Level Domain feature to differentiate the domains based on geography. When a language encoding for requests directed to a particular domain changes, it might be because of a change in the user preference of language or anomalous traffic (bot C&C traffic). This feature along with the TLD feature can be used to model the geography of the domain.
- *Randomness*: This feature is calculated based on Shannon entropy of English words. Since the user requests made by the bot are random in nature due to the Domain name Generation Algorithm used to craft the user names, the randomness of these requests will be more than general requests. For example, if user normally requests users like www.twitter.com/BarackObama and suddenly start making requests like www.twitter.com/snsaUe43Ksak, the randomness changes. We

consider two randomness features. Randomness of the string immediate next to the host name and the randomness of the whole user request. Both these will help capture the normal entropy of the user and thus differentiate anomalous traffic from general usage.

- *Top Level Domain* (TLD): Top Level Domains like .com, .net, .org etc., help us model the user behavior more accurately based on the kinds of the domain he/she is visiting. IF the user traffic contains .com or .net normally and suddenly .in or .cn or .ru starts appearing in the user traffic, then it is an indication of the anomaly. This might be due to change in user behavior or presence of bot.
- *Time of the request*: Time is one of the very important metric due to the fact that bot checks for the presence of the user before retrieving his first tweet and authenticating it. This check for the presence of the user will generate a traffic that will be almost anomalous because it varies with user traffic in frequency at that moment in time and also for the whole day. By carefully modelling this anomaly, we have a greater chance of predicting the bot traffic.

The frequency of the requests collected for every hour per domain will help in detection of the bot even if it employs Time space randomization. The bot which employs time space randomization will not make all the requests at one instance. Instead, it will space the requests to C&C server through the time so that any metric which is trying to catch the bot based on Time based similarity will fail. Our model of calculating frequencies of the traffic for every hour will detect the bot even if it employs Time space randomization. Our method is useful even in the case of a bot that employs a dynamic Username Generation Algorithm. A dynamic Username Generation Algorithm does not construct the usernames based on static seed values like date and/or time. A dynamic Username Generation Algorithm will make use of changing seed values like “trending topics in twitter”, “top searches in google”. This kind of algorithm is more powerful because it is immune to reverse

engineering. If the bot does not emply dynamic UGA, and if someone reverse engineers the malware (bot) and found the UGA, it is possibly to disrupt the botnet by registering all those usernames in advance. Even though it is time consuming and tedious, it can still be done. However, a bot employing UGA is not susceptible to reverse engineering because its UGA takes seed values that change constantly over time. Twitter trending topics change frequently and so does the top google searches. Our method of detection works for the bots that employ dynamic UGA because we are bot dependent on the type of UGA used for the generating usernames. As long as the traffic is anomalous, our method will try to differentiate it.

5.2 Evaluation

Our evaluation consists of traffic collected from two users. Our training period consists of 15 days. All the subsequent days are considered as testing data. We train the classifier (Kernel Logistic Regression) based on the data of these two users for a period of 15 days. One user is given a Samsung Galaxy Nexus and the other user is given a LG Nexus 4. Both the smartphones are new and are installed with the Application Proxy before given to the users. The users are Graduate students from Arizona State University. No restrictions are placed on the usage of the smartphones. The users are encouraged to use the smartphones normally. The smartphones are assumed to be clean of any malware for the first 15 days of usage. The application proxy redirects the traffic to the designated server before returning the results to the smartphones. The server is a Intel core i5 dual core machine with 8 GB RAM. The server collects the data from the smartphones and is responsible for building the usage models using Kernel Logistic Regression classifier after 15 days.

The data collected over 15 days for each user is about 120 MB each. There are 156 unique domains being accessed by user 1 and 128 unique domains being accessed by user 2. After the 15 day training period, one of the smartphones is infected with the prototype bot and the traffic is collected and compared to the existing models built on the training data.

5.3 Results

The Kernel Logistic Regression classifier is used for training and testing of the user models. As mentioned above, the usage of the user is modelled as distinct phases. One of which is used for capturing the semantics of user requests and one for the timing and frequency of the requests. The user requests are divided on per domain basis and the kernel logistic regression classifier is used to train the model of the users on per domain basis in these phases. The training data consists of traffic from the users without any infection. Next, the prototype bot is injected into the smartphones and it is configured to try out 100, 50 and 25 usernames per day respectively for 3 days. The traffic of these 3 days is considered as 3 different samples of test data and is inputted to the models already trained individually.

When the bot made 25 UGA generated requests per day, our classifier returned a mismatch of 56.6% for phase I and a mismatch of 59.3 % for the phase II for the domains associated with the UGA. When the bot made 50 UGA generated requests per day, our classifier returned a mismatch of 59 % for Phase I and 65.4 % for Phase II. When the bot made 100 UGA requests per day, our classifier returned 63 % mismatch for Phase I and 78.8 % mismatch for Phase II.

Based on our observations, when the threshold match is set for 55 % for Phase I and 59 % for Phase II, the domains associated with the bot traffic will be reported as mismatch for 99 % of the time. The test data for each day consists of almost 500 requests made per day including the bot generated traffic.

The advantage of our method is less number of false positives. Since, we attempt to estimate the mismatch threshold and not the match threshold, we achieved almost zero rate of false positives. This might lead the system to believe that some requests of the bot as benign but since the bot make much more requests than one, the model is bound to catch one of them and report to the user.

Chapter 6

CONCLUSION AND RECOMMENDATIONS

6.1 Overview

This chapter includes the research efforts discussed in this thesis. The following sections discuss the significance of the work done and recommendations for the future work.

6.2 Significance of Research

Smartphone malware is one of lucrative and dangerous phenomenon in current cyber space. Among them, botnets are one of the widely deployed malware due to the dynamic nature of their use. Advanced botnets targeting Online Social Networks are in particular malicious because of their resilience to botnet take downs.

This research provides a framework to detect the bots present in the smartphones leveraging Online Social Networks as Command & Control channel. The detection is based upon modelling the user's web usage patterns and detecting any anomalous traffic that fall outside normal user web usage. This research develops a prototype bot of the above type and identifies the features needed to model the user behavior.

This research extends the work of detection of bots in the smartphones by successfully modelling the user behavior in web and detecting anomalous traffic.

6.3 Recommendations for Future Research

The design and proof of concept are the first steps towards detection of advanced bots in the smartphone domain. The future research to improve the proposed approach can be as follows:

1. Eliminate the central server. Collect and model the user data in the smartphone. This is possible due to the hardware improvements being done to the smartphones these days. Almost all the smartphones have dual cores and at least 1 GB of RAM. The energy

consumption can be made negligible by building and updating the model when the phone is plugged in for long durations. This will reduce the reliance upon the central server and allays the privacy concerns of the individual users.

2. Continuous learning: Continuous learning mechanism can be implemented which instead of training a model and testing the data against it and updating the model periodically with the users data, will update the model for every request user makes, benign or malign.

REFERENCES

[1] “Global marketing statistics”, 2013, http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats.

[2] Gu, Guofei, Roberto Perdisci, Junjie Zhang, and Wenke Lee. "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection." In USENIX Security Symposium, pp. 139-154. 2008.

[3] Gu, Guofei, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. "Bothunter: Detecting malware infection through ids-driven dialog correlation." InProceedings of 16th USENIX Security Symposium on USENIX Security Symposium, p. 12. USENIX Association, 2007.

[4] Gu, Guofei, Junjie Zhang, and Wenke Lee. "BotSniffer: Detecting botnet command and control channels in network traffic." (2008).

[5] “Understanding and defending against Botnets and stealthy malware”, 2009, http://www.infoq.com/articles/intel-botnets-malware-security.

[6] “The botnet business”, 2008, http://www.securelist.com/en/analysis/204792003/The_botnet_business.

[7] “DNS Sinkhole”, 2010, http://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523?

[8] Wang, Ping, Lei Wu, Baber Aslam, and Cliff Changchun Zou. "A systematic study on peer-to-peer botnets." In Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on, pp. 1-8. IEEE, 2009.

[9] “How FAST-FLUX SERVICE NETWORKS WORK”, 2008, http://www.honeynet.org/node/132

[10] Antonakakis, Manos, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. "From throw-away traffic to bots: detecting the rise of DGA-based malware." In Proceedings of the 21st USENIX Security Symposium. 2012.

[11] Singh, Kapil, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. "Evaluating bluetooth as a medium for botnet command and control." InDetection of Intrusions and Malware, and Vulnerability Assessment, pp. 61-80. Springer Heidelberg, 2010.

[12] “Mobile Botnets: From anticipation to reality”, 2013, http://securityaffairs.co/wordpress/12862/malware/mobile-botnets-from-anticipation-to-reality.html

[13] Zeidanloo, Hossein Rouhani, Mohammad Jorjor Zadeh Shooshtari, Payam Vahdani Amoli, M. Safari, and Mazdak Zamani. "A taxonomy of botnet detection techniques." In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 2, pp. 158-162. IEEE, 2010.

[14] Xiang, Cui, Fang Binxing, Yin Lihua, Liu Xiaoyi, and Zang Tianning. "Andbot: towards advanced mobile botnets." In Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats, pp. 11-11. USENIX Association, 2011.

[15] "Domain Generation Algorithm", 2013, http://en.wikipedia.org/wiki/Domain_generation_algorithm