Design of SMS Commanded-and-Controlled and P2P-Structured Mobile Botnets

Yuanyuan Zeng
Perimeter E-Security
Raleigh, North Carolina
yzeng@perimeterusa.com

Kang G. Shin
University of Michigan
Ann Arbor, Michigan
kgshin@eecs.umich.edu

Xin Hu
IBM Research
Hawthorne, New York
huxin@us.ibm.com

ABSTRACT

Botnets are one of the most serious security threats to the Internet and personal computer (PC) users. Although botnets have not yet caused major outbreaks in the mobile world, with the rapidly-growing popularity of smartphones such as Apple's iPhone and Android-based phones that store more personal data and gain more capabilities than earlier-generation handsets, botnets are expected to become a severe threat to smartphones soon. In this paper, we propose the design of a mobile botnet that makes the most of mobile services and is resilient to disruption. The mobile botnet utilizes SMS messages for C&C and a P2P structure as its topology. Our simulation results demonstrate that a modified Kademlia—a structured architecture—is a better choice for the mobile botnet's topology. In addition, we discuss potential countermeasures to defend against this mobile botnet threat.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Invasive Software*

General Terms

Security

Keywords

Smartphone Security, Malware, Mobile Botnets

1. INTRODUCTION

Botnets are one of the most serious security threats to the Internet and the personal computer (PC) world, but they are still rare for the mobile world. Recently, with the rapidly-growing popularity of smartphones, such as the iPhone and Android-based phones, attacks on cellular networks and devices have grown in number and sophistication. A drastic increase in downloading and sharing of third-party applications and user-generated content makes smartphones vulnerable to various types of malware. Smartphone-based banking services have also become popular without protection features comparable to those on PCs, enticing cyber crimes. There are already a number of reports on malicious applications in the Android Market [1]. Although the Android platform requires that applications should be certified before their installation, its control policy is rather loose—allowing developers to sign their own applications—so that attackers can easily get their malware into the Android Market. The iPhone's application store controls its content more tightly, but it fails to contain jailbroken iPhones which can install any application and even run processes in the background. As smartphones are increasingly used to handle more private information with more computing power and capabilities, but without adequate security and privacy protection, attacks targeting mobile devices are becoming more sophisticated. Since the appearance of the first, proof-of-concept mobile worm, Cabir, in 2004, we have witnessed a significant evolution of mobile malware. The early malware performed tasks, such as infecting files, replacing system applications and sending out SMS or MMS messages. One malicious program is usually capable of only one or two functions. Although the number of mobile malware families and their variants has been growing steadily in recent years, their functionalities have remained simple until recently.

SymbOS.Exy.A trojan [2] was discovered in early 2009 and its variant SymbOS.Exy.C resurfaced in July 2009. This mobile worm, which is said to have "botnet-esque" behavior patterns, differs from other mobile malware, because after infection, it connects back to a malicious HTTP server and reports information of the device and its user. The Ikee.B worm [3] that appeared late 2009 targets jailbroken iPhones, and has behavior similar to SymbOS.Exy. Ikee.B also connects to a control server via HTTP, downloads additional components and sends back the user's information. With this remote connection, it is possible for attackers to periodically issue commands to and coordinate the infected devices to launch large-scale attacks. In March 2011, over 50 applications found to contain a type of malware called "DroidDream" were removed from the Android Market. This malware is able to root the infected device and steal sensitive information. It was speculated that the end goal of DroidDream was to create a botnet [4]. In February 2012, RootSmart [5], a malicious application in third party Android markets in China, was reported to create a botnet containing thousands of Android devices. Once started, RootSmart connects to a remote server to send various information of the infected phone and fetches a root exploit from the server to obtain escalated privilege to the phone. The infected phone is configured to send premium SMS messages and use other premium telephony services without users' knowledge, generating profits for the botmaster. Observing the trend of recent mobile malware, we expect that mobile botnets will become a serious threat to smartphones soon.

In this paper, we propose the design of a mobile botnet that makes the most of mobile services and is resilient to disruption. Within this mobile botnet, all C&C communications are done via SMS messages since SMS is available to almost every mobile phone. To hide the identity of the botmaster, there are no central servers dedicated to command dissemination that is easy to be identified and then removed. Instead, we adopt a P2P topology that allows botmasters and bots to publish and search for commands in a P2P fashion, making their detection and disruption much harder. Our contributions are three-fold. First, to the best of our knowledge, we are the first to design mobile botnets with focuses on both C&C protocol and topology by integrating the SMS service and the P2P topology. The main purpose of this work is to shed light on potential botnet threats targeting smartphones. Since current techniques against PC botnets may not be applied directly to mobile botnets, our proposed mobile botnet design makes it possible for security researchers to investigate and develop new countermeasures before mobile botnets become a major threat. Second, we present a method to carefully disguise C&C content in spam-looking SMS messages. Using this approach, the botnet can stealthily transmit C&C messages without being noticed by phone users. Third, we test and compare two P2P architectures that can be used to construct the topology of our mobile botnet on an overlay simulation framework, and finally propose the architecture that best suits mobile botnets.

The remainder of the paper is organized as follows. Section 2 describes the related work. Section 3 details the proof-of-concept design of our mobile botnet. Section 4 presents our simulation and evaluation results. Section 5 discusses potential countermeasures against the mobile botnets. The paper concludes with Section 6.

2. RELATED WORK

The research areas most relevant to our work are P2P-based botnets and botnet C&C evaluation. Wang *et al.* [6] proposed the design of an advanced hybrid P2P botnet that implemented both push and pull C&C mechanisms and studied its resilience. In [7] they conducted a systematic study on P2P botnets including bot candidate selection and network construction, and focused on index poisoning and Sybil attacks. Overbot [8] is a botnet protocol based on Kademlia. The strength of this protocol lies in its stealth in the communication between the bots and the botmaster, which leverages a public-key model. Davis *et al.* [9] compared the performance of Overnet with that of Gnutella and other complex network models under three disinfection strategies. Singh *et al.* [10] evaluated the viability of email communication for botnet C&C. Nappa *et al.* [11] proposed a botnet model exploiting Skype's overlay network to make botnet traffic undistinguishable with legitimate Skype traffic. All of these dealt with botnets in the PC world, while our work targets mobile botnets, in which C&C channel and network structure requirements are different, in view of unique services and resource constraints on smartphones. Dagon *et al.* [12] proposed key metrics to measure botnets' utility for conducting malicious activities and considered the ability of different response techniques to disrupt botnets.

There are numerous efforts on mobile malware focusing on vulnerability analysis and attack measurements. The former investigates ways of exploiting vulnerable mobile services, such as Bluetooth and MMS [13, 14], while the latter characterizes the feasibility and impact of large-scale attacks targeting mobile networks, mostly Denial of Service (DoS) attacks [15]. There are a few recent papers treating the idea of mobile botnets. In [16], the focus is on the attack aspect—whether compromised mobile phones can generate sufficient traffic to launch a DoS attack. Singh *et al.* [17] investigated using Bluetooth as a C&C to construct mobile botnets without any analysis on their network structure. Hua *et al.* [18] proposed a SMS-based mobile botnet using a flooding algorithm to propagate commands with the help of an internet server. The use of the central server, however, may lead to single-point-of-failure. Mulliner *et al.* [19] demonstrated the ways to command and control botnets via SMS or IP-based P2P networks using a tree topology. Under such topology, when a node fails, all of its subnodes will be isolated from the botnet, difficult to get commands. Weidman [20] also considered utilizing SMS messages for botnet C&C and presented a method to conceal malicious SMS messages from users on smartphones. It is worth noting that, different from all these works, our SMS-based botnet is built upon a decentralized P2P topology, without assistance from any central servers. The integration of SMS and P2P makes our botnet stealthy and resilient to disruption.

3. MOBILE BOTNET DESIGN

We now present the detailed design of a proof-of-concept mobile botnet. The botnet design requires three main components: (1) vectors to spread the bot code to smartphones; (2) a channel to issue commands; (3) a topology to organize the botnet. We will briefly overview approaches that can be used to propagate malicious code and then focus on C&C and topology construction.

3.1 Propagation

The main approaches used to propagate malicious code to smartphones are user-involved propagation and vulnerability exploits.

In the first approach, the most popular vector is social engineering. Like their PC counterparts, current smartphones have frequent access to the Internet, becoming targets of malicious attacks. Thus, spam emails and MMS messages with malicious content attachments, or spam emails and SMS messages with embedded links pointing to websites hosting the malicious code, can easily find their way into a mobile phone's inbox. Without enough caution or warning, a mobile phone user is likely to execute the attachments or click those links to download malicious programs. The advantage of such schemes is that they can reach a large number of phones. Nevertheless, as smartphones run on a variety of operating systems, we expect multiple versions of bot code prepared to guarantee its execution. Another user-involved propagation vector can be Bluetooth, which utilizes mobility. Mobile phone users move around so that the compromised phones can use Bluetooth to search for devices nearby and after pairing with them successfully, try to send them malicious files.

Exploiting vulnerabilities to spread malicious code is common in the PC world. However, since there are various mobile platforms and most of them are closed-source, it is relatively difficult to find vulnerabilities in mobile devices. To date, some vulnerabilities have been discovered. For example, the HTC's Bluetooth vulnerability, which allows an attacker to gain access to all files on a phone by connecting to it via Bluetooth, was disclosed by a Spanish security researcher [21]. Mulliner *et al.* [22] discovered a way of directly manipulating SMS messages on different mobile platforms, without necessarily going through the mobile provider's network. In both cases, OS vendors immediately released patches to the public after the vulnerabilities were publicized, leaving few opportunities for a real exploit in the wild. Once launched in their targets, vulnerability exploits always have a higher success rate than that of user-involved approaches. As mobile platforms open up and mobile applications and services become abundant, vulnerability exploits will play a major role in mobile malware propagation.

3.2 Command and Control

In our mobile botnet, SMS is utilized as the C&C channel, i.e., compromised mobile bots communicate with botmasters and among themselves via SMS messages. Botnets in the PC world mostly rely on IP-based C&C delivery. For example, traditional botnets use centralized IRC or HTTP protocol, whereas newly-emerged botnets take advantage of P2P communication. Unlike their PC counterparts, smartphones can hardly establish and maintain steady IP-based connections with one another. One reason is that they move around frequently. Another reason is that private IPs are normally used when smartphones access networks such as EDGE, 3G and 4G networks, meaning that accepting incoming connections directly from other smartphones is a difficult task. Given this limitation, if a mobile botnet considers an IP-based channel as C&C, it needs to resort to centralized approaches in which bots connect to central servers to obtain commands. Such approaches, however, are vulnerable to disruption because the servers are easy to be identified by defenders. Thus, to construct a mobile botnet in a more resilient manner, a non-IP-based C&C is needed.

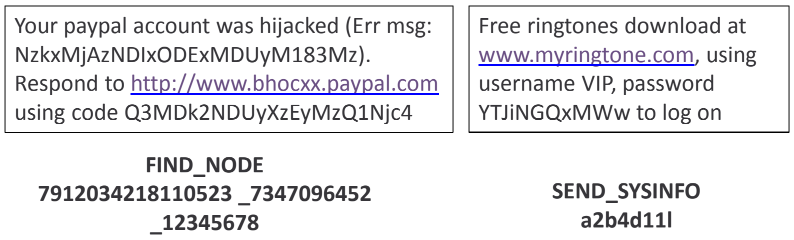
There are a few advantages for choosing SMS as a C&C channel. First, SMS is ubiquitous. It is reported that SMS text messaging is the most widely used data application on the planet, with 2.4 billion active users, or 74% of all mobile phone subscribers sending and receiving text messages on their phones [23]. When a mobile phone is turned on, this application always remains active. Second, SMS can accommodate offline bots easily. For example, if a phone is turned off or has poor signal reception in certain areas, its SMS communication messages will be stored in a service center and delivered once the phone is turned back on or the signal becomes available. Third, malicious content in the C&C communication can be hidden in SMS messages. According to a survey in China [24], 88% of the phone users polled reported they had been plagued by SMS spamming. As SMS spamming becomes prevalent, bots can encode commands into spam-looking messages so that users will not suspect. Last but not least, currently there are multiple ways to send and receive free SMS messages directly on smartphones [25, 26] or through some web interfaces. We will describe such methods in Section 3.2.2. Even when the free texting is unavailable, as many phone users use SMS plans to avoid per-message charge and incoming messages are free of charge in some countries, with the design goal of minimizing the number of SMS messages we expect that using SMS as C&C will not incur considerable costs.

3.2.1 Protocol Design

Our goal is to let a phone that has installed our bot code perform activities according to the commands in SMS messages without being noticed by the user. In our design, every compromised phone has an 8-byte passcode. Only by including this passcode into the SMS messages, can other phones successfully deliver C&C information to this particular phone. Upon receipt of a SMS message, this phone searches for its passcode and pre-defined commands embedded in the message to tell if it is a C&C message. If found, the commands are immediately executed by the phone. Two issues need to be addressed here. First, how are passcodes allocated among compromised phones? Second, how to make C&C SMS messages appear harmless so that users may not notice the malicious content?

In our botnet, passcodes are allocated by botmasters to segment a botnet into sub-botnets, each with a different function. For example, one sub-botnet is responsible for sending out spam messages, while another is in charge of stealing personal data and transferring them to a malicious server. Each sub-botnet will be identified by its unique passcode that is hard-coded into the bot's binary. In other words, all bots within the same sub-botnet share the same passcode so that they can communicate with one another and also with the botmaster. Using a unique passcode for each bot will be more secure than using one passcode for an entire sub-botnet because in the latter case, the passcode will be discovered more easily. However, there is a trade-off: using a unique passcode will add more overhead due to the pairwise passcode exchange before each communication. The additional cost is undesirable since our goal is to minimize the number of SMS messages to be sent.

Not only do we require a passcode included in each SMS communication message, but also we encode commands to make it difficult for a user to figure them out. In fact, on the Android platform, it is possible for an application to send out SMS messages stealthily, to get immediate notification of every incoming SMS message by registering itself as a background service and to read and execute commands or even delete the message before the user sees it. We still want to hide the C&C messages because other mobile platforms are more restricted than Android; they may not allow our bots to both send and receive SMS messages without notifying the user. If malicious messages show contents directly, they will be easily captured and manipulated by defenders. To evade such detection, we want to make a command-embedded SMS message look like a common message such as a spam message. There are benefits of using spam-like messages to transmit C&C. As pointed out in [27], cellular carriers cannot simply block offending SMS messages because the senders have paid for the messages and the carriers fear permanent deletion of legitimate messages when there are no spam folders available. We will present a real-world experiment in Section 4.3. Even if in the future the carriers filter out spam messages and dump them into spam folders, similar to the email filtering, spam messages can still reach


Your paypal account was hijacked (Err msg: NzkxMjAzNDIxODExMDUyM183Mz). Respond to http://www.bhocxx.paypal.com using code Q3MDk2NDUyXzEyMzQ1Njc4

Free ringtones download at www.myringtone.com, using username VIP, password YTJiNGQxMWw to log on

FIND_NODE
7912034218110523 _7347096452
_12345678

SEND_SYSINFO
a2b4d11l

Figure 1: Disguised SMS messages

the target phones by going to the spam directory, which actually helps hide the C&C because users tend to ignore spam.

Considering the fact that each SMS message only contains up to 160 characters, commands in our botnet are concise. For example, "FIND_NODE" instructs a bot to return the phone numbers of certain nodes; "SEND_SYSINFO" asks a bot to reply with system information. To disguise messages, each command is mapped to one spam template. Additional information such as the phone number and the aforementioned passcode are variables in the templates, and they are Base64-encoded. Figure 1 shows two disguised SMS messages. The first one is a "FIND_NODE" message (146 characters) with passcode 12345678 requiring the recipient to locate a bot whose ID is 7912034218110523, and the result should be returned to the bot whose phone number is (734)7096452. NzkxMjAzNDIxODExMDUyM183Mz and Q3MDk2NDUyXzEyMzQ1Njc4—two random strings together —are the Base64-encrypted 7912034218110523_7347096452 _12345678. The entire encoded string is split into two—disguising one as an error message and the other as a code—making it resemble a spam message. The second example is a "SEND_SYSINFO" message (98 characters) with a passcode a2b4d11l. This template is different from that of the "FIND_NODE" message. The passcode is also Base64-encoded and appears as a password in the disguised message. To decode messages, each bot keeps a command-template mapping list. Only tens of commands are needed in our botnet, so this list is not long. To make detection harder, one command message can correspond to different spam templates and the templates can be updated periodically. As just shown, a command along with additional information can be easily embedded into one SMS message which appears to be a spam, familiar to today's phone users, so users are likely to ignore such messages even if they open and read them. If users choose to delete these messages, it will not cause any problem to the botnet because the commands have already been executed upon their receipt.

3.2.2 *Sending SMS Through the Internet*

Although sending SMS messages through the cellular networks is always possible, the botmasters want to hide their identity and lower costs as much as possible. To achieve this goal, botmasters can use the Internet to disseminate C&C messages to the mobile botnet. There are several ways to do this. Many advertisement-based websites provide free SMS services. Botmasters can type in messages via these websites and have them sent to mobile bots, feasible for low-volume messaging. Using such services does not require the sender's mobile number, an email address is sufficient if a reply is expected. If the botnet is large, botmasters need to create an account with mobile operators or SMS service providers to make high-volume messaging possible at the lowest price. Usually, this can be done by sending and/or receiving SMS messages via email through a SMS gateway connecting directly to a Mobile Operator's SMSC (Short Message Service Center). Currently, smartphone applications such as [25, 26] offer free domestic and international text messaging when the phone is connected to a WiFi and support both one-on-one and group texting. The user only needs to provide a screen name to send and receive messages without revealing its identity. Both the botmasters and bots can take advantage of such a service whenever possible to avoid messaging costs.

To sum up, using SMS messages as the C&C is a viable solution for a mobile botnet. Not only is SMS ubiquitous to every mobile phone, but botmasters and bots are also able to disguise SMS messages, send bulk messages from the Internet at very low cost while hiding their identities. Thus, using SMS is both economical and efficient for the botnet.

3.3 Mobile Botnet Topology

In the previous section we have described the way SMS messages form the C&C communication in our mobile botnet. In what follows, we introduce P2P topologies that may be utilized to organize the botmaster and bots for publishing and retrieving commands, and describe how to leverage existing P2P architectures to meet the need for mobile botnet construction.

3.3.1 *Possible Topologies*

Similar to botnets in the PC world, a mobile botnet can be structured in a traditional centralized way or in a newly-emerged decentralized P2P fashion. In the first approach, botmasters hard-code into each bot's executable a set of phone numbers that are under their direct control. When a mobile phone is converted to a bot, it contacts those hard-coded phones to request commands or wait for commands to be pushed to them. Such a centralized topology is easy to implement but not resilient to disruption. Obviously, once defenders obtain these phone numbers, they can track down the botmasters and then disable the botnet. To make our botnet robust to defenses, we adopt a P2P structure instead.

Currently, there are several structures for P2P networks; they can be divided into three categories: centralized, decentralized but structured, and unstructured. Centralized P2P networks have a constantly-updated directory hosted at central locations. Peers query the central directory to get the addresses of peers having the desired content. This structure is similar to the traditional centralized botnet architecture and hence vulnerable to the central-point-of failure. Decentralized but structured P2P networks have no central directory and contents are not placed at random nodes but at specific locations. The most common systems in this category are Distributed-Hash-Table (DHT)-based P2P networks, ensuring that any peer can efficiently route a search to some peer with the desired content. One notable implementation is Kademlia [28], used by several current P2P applications, such as eMule and BitTorrent. Decentralized and unstructured P2P networks have neither central directories nor control over content placement. If a peer wants to find certain content in the network in old protocols such as Gnutella, it has to flood its query to the entire network to find peers sharing the data. To address the scalability issues, current unstructured networks adopt different query strategies to avoid flooding. There have also been extensive studies on how to make Gnutella-like systems scalable. One such design is Gia [29].

3.3.2 Design

Both structured and unstructured P2P architectures can be modified to suit our need for the mobile botnet because their decentralized nature hides the botmaster's identity. Since the mobile botnet design should consider not only robustness but also feasibility and efficiency on smartphones, we need to compare these two architectures to see which is more suitable. Specifically, we base our structured and unstructured botnet topology on Kademlia and Gia, respectively, for comparison. Note that in our botnet, bots obtain commands mainly in a *pull* style, i.e., the botmaster publishes commands and bots are designed to actively search for these commands. The other possible mechanism for command transfer is *push*, meaning that bots passively wait for commands. We prefer *pull* to *push* because *push* will get malicious activities exposed easily. That is, under *push* many SMS messages are sent out from one or a few central nodes, whereas *pull* can be implemented in a more distributed fashion. In what follows, we overview each protocol and describe our design.

Kademlia is DHT-based and has a structured overlay topology, in which nodes are identified by node IDs generated randomly and data items are identified by keys generated from a hash function. Node IDs and keys are of the same length (128-bit). Data items are stored in nodes whose IDs are close to data items' keys. The distance between two identifiers, x and y, is calculated by bitwise exclusive or (XOR) operation: $d(x, y) = x \oplus y$. For each $0 \leq i < 128$, each node keeps a list for nodes of distance between 2^i and 2^{i+1} from itself. This list is called a k-bucket, and can store up to k elements. There are four types of RPC messages in Kademlia: PING, STORE, FIND_NODE and FIND_VALUE. PING checks whether a node is online. STORE asks a node to store data. FIND_NODE provides an ID as an argument and requests the recipient to return k nodes closest to the ID. FIND_VALUE behaves similarly to FIND_NODE. The only exception is that when a node has the data item associated with the key, it returns the data item. Since there is no central sever, each node has a hard-coded peer list in order to bootstrap into the network.

Considering the differences between smartphones and personal computers as well as the SMS C&C channel we adopt, we modify Kademlia's design to be suitable for our mobile botnet's structured overlay construction. First, we do not use PING messages to query whether a node is alive and should be removed from its k-bucket. One reason for this is that SMS messages transmitting C&C can always reach their recipients even if these phones are not online (messages are stored in the SMSC for later delivery). The other reason is that our design tries to minimize the number of messages sent and received. Removing PING messages effectively reduces C&C traffic and thus, the possibility of being noticed by phone users and defenders. Second, instead of being randomly generated, a node ID is constructed by hashing its phone number, similar to the notion in Chord [30] that a node ID is the hash of its IP address. Doing so can undermine the effectiveness of Sybil attacks in which defenders add nodes to join the botnet to disrupt C&C transmission. Evidently, if node IDs are allowed to be randomly chosen, defenders will take advantage of this by selecting IDs close to command-related keys to ensure a high probability that these sybil nodes are on the route of command search and publish queries. In addition, the absence of an authentication mechanism in Kademlia, meaning that anyone can insert values under specific keys, presents an opportunity for defenders to launch index poisoning attacks by publishing fake values under command-related keys once they know these keys, in order to disrupt C&C. We thus use a public key algorithm to secure the command content. While publishing a command, the publisher (the botmaster) needs to attach a digital signature to that command. The signature is the hash value of the command signed by the botmaster's private key. Its corresponding public key is hard-coded in each bot's binary. In this way, bots that will store the command are able to verify that the command is indeed from the botmaster not anyone else.

Gia improves Gnutella protocol and has an unstructured overlay topology. Since Gnutella has a scaling problem due to the flooding search algorithm, Gia modifies Gnutella's design and improves its scalability significantly. There are four key components in Gia's design: (1) a topology adaptation protocol to put most nodes within short reach of high-capacity (able to handle more queries) nodes by searching and adding high-capacity and high-degree nodes as neighbors; (2) an active flow control scheme to avoid overloaded nodes by assigning flow-control tokens to nodes based on capacity; (3) one-hop replication to maintain pointers to the content offered by immediate neighbors; (4) a search algorithm based on biased random walks directing queries to nodes that are likely to answer the queries.

Our design of unstructured overlay topology is based on Gia as mentioned before. Our design removes the one-hop replication scheme because it requires each node to index the content of its neighbors and to exchange this information periodically. This scheme may help reduce the number of hops for locating a command, but will incur additional storage and computation overheads. Moreover, each SMS message has a limited length so that the exchange of index information cannot be done with a single message but requires multiple messages, increasing the number of messages generated. In our mobile botnet, the drawbacks of using such a scheme will outweigh its benefits, and we thus opt out of this scheme. Three other components are important to our botnet because their combination ensures queries to be directed to high-capacity nodes that can provide useful responses without getting overloaded. This is desirable, especially in a mobile phone network, since smartphones also have different capacities under different situations. For example, in a poor-signal area or when the phone is on a voice call (SMS messages use the same control channel as voice calls for delivery), the phone's capability of handling SMS messages is lowered, so it can only answer fewer queries. Overloading mobile bots is also a concern. If one bot receives/sends a large number of SMS messages during a short period of time, its battery can be drained quickly, and draw the user's attention. Overloading can be prevented using the flow-control scheme in Gia. Another design choice worth mentioning is that similar to the modified Kademila, a digital signature is attached to every command to be published.

4. EVALUATION

4.1 Comparing Two P2P Structures

We now describe our simulation study of structured and unstructured P2P architectures for mobile botnets and compare their performances. In the simulation, all nodes are assumed to have already been infected by the vectors described in Section 3.1. Our evaluation focus is not on how the malicious bot code propagates, but on how the botnet performs under two different P2P structures.

We modified OverSim [31], an open-source overlay network simulation framework, to simulate mobile botnets with the two P2P structures. While comparing P2P structures' performances, logical connections (SMS activities) among mobile nodes matter most, i.e., what we care is the overlay network not the underlying physical network. For example, the fact that mobile bots move around is not important in our simulation because the change of geographic location hardly affects bots' SMS message sending/receiving.

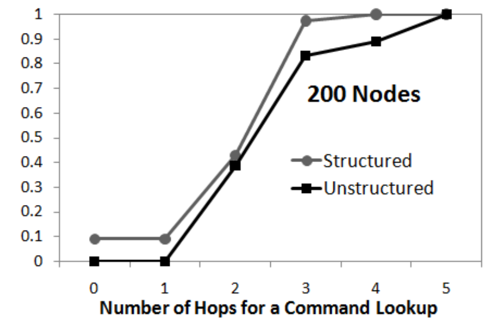
The metrics we use to measure performance are: number of overlay hops needed for a command lookup; total number of SMS messages sent (number of those sent = number of those received) when a botmaster-issued command is acquired by every node; percentage of total number of SMS messages sent by each node during this entire command-lookup; and message delay (from the start of the query until a command is received). These metrics reflect how well each architecture meets the requirement of our mobile botnet, namely, minimizing the number of SMS messages sent and received, load-balancing and locating commands in a timely manner.

The churn (participant turnover) model we adopted in the simulation is the lifetime churn. In this model, on creation of a node, its lifetime will be drawn randomly from a Weibull distribution which is widely used to characterize a node's lifetime. When the lifetime is reached, the node is removed from the network. A new node will be created after a dead time drawn from the same probability distribution function. We set the mean lifetime to 8*3600=28800s, assuming that each phone will stay connected to the botnet for an average of 8 hours. Considering the unavailability of real field data on mobile phones' online behavior, we made this rough estimate. We will later evaluate the effect of different mean lifetimes on the botnet performance.

Besides the aforementioned performance metrics, another important metric is scalability for which we simulated two botnets with 200 and 2000 nodes, respectively. In each botnet, a command from the botmaster is published, and every node is designed to locate this command by issuing lookup queries. The simulation ends when all nodes successfully retrieve the command. In the structured botnet case, we ran the modified Kademlia protocol, with k-bucket size $k = 8$ and the number of nodes to ask in parallel $\alpha = 3$. In the unstructured botnet case, we ran the modified Gia protocol, with minimum number of neighbors $min\_nbrs = 3$, maximum number of neighbors $max\_nbrs = 10$ and maximum number of responses $max\_responses = 1$.

Now, we present and discuss the comparison results. For each metric, we first look at the 200-node botnet and then the 2000-node botnet. Figure 2 plots the CDFs of the number of hops needed to retrieve a targeted command. In the 200-node botnet, for the structured architecture, 97% of lookups can be completed within 3 hops. The corresponding number for the unstructured botnet is 5 hops. In the 2000-node botnet, despite the increased network size, 99% of lookups under the structured architecture are fulfilled within 4 hops, but under the unstructured 8 hops are required. Figure 3 shows the CDFs of the total number of SMS messages sent from each node when the command spreads to the entire botnet, which is the total communication overhead. In the 200-node botnet, under the structured architecture, about 80% of nodes generate fewer than 15 messages during the entire period, while under the unstructured architecture 69% of nodes can do so. The average number of messages sent is 11 for the structured and 15 for the unstructured, respectively. In the 2000-node botnet, with more nodes and more lookups, the message overhead unsurprisingly increases. 80% of nodes send fewer than 20 messages (51% of nodes send fewer than 10 messages) for the structured architecture with an average of 22 messages sent by each node. Only 40% of nodes send fewer than 20 messages for the unstructured architecture with an average of 44 messages.

From the above observations, we can see that the structured botnet, in general, requires fewer number of hops to locate a command and incurs a lower message overhead (15 to 20 messages) on each node than the unstructured one does in both 200- and 2000-node cases. Compared to the unstructured botnet, the structured architecture also scales better, considering its slight increases in the number of hops and messages when the botnet becomes large. This is expected because in a structured network, data items are placed at deterministic locations so that fewer hops and query messages are required to locate the targeted data and the network can accommodate a large number of nodes. As mentioned before, on smartphones such as Android-based phones, bots are able to send and receive C&C SMS messages stealthily without notifying users. Users may figure that out while seeing the monthly bills, but by then bots have already performed malicious tasks. Even if users are able to see them on the phone, since the C&C messages are disguised as spam, they cause little suspicion. Even so, one may still wonder: would SMSC observe a surge of messages among infected phones and raise alerts? SMS market statistics show that: "In 2009, U.S. cell phone subscribers sent and received on average 390 text messages per month according to the Mobile Business Statistics [32]." We believe that tens of messages overhead per phone may not draw much attention from the SMSC considering a phone's normal messaging volume. Also, since most attacks such as information steal-



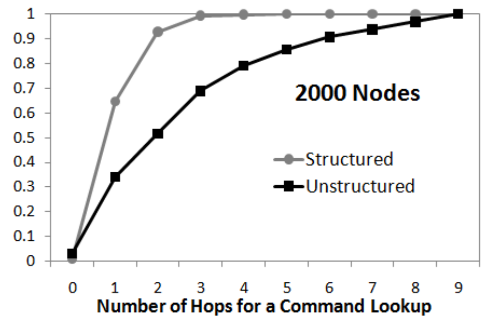
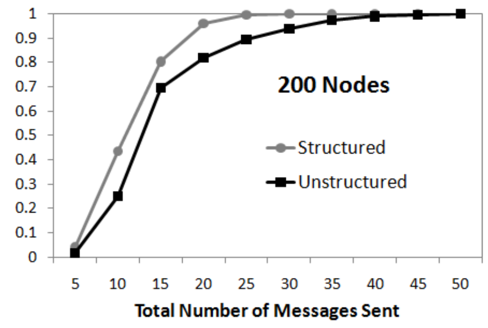


Figure 2: CDFs of the number of hops needed for a command-lookup



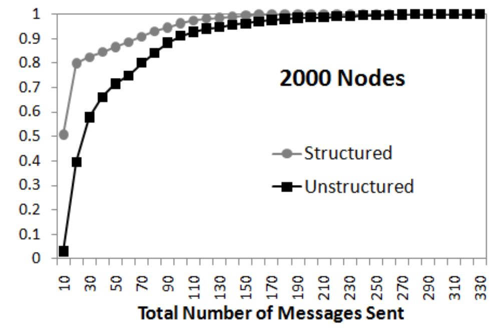
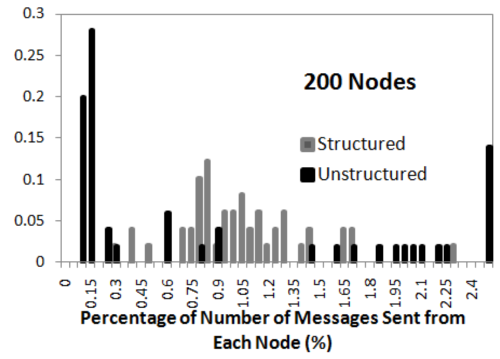


Figure 3: CDFs of the total number of messages sent to perform all lookups



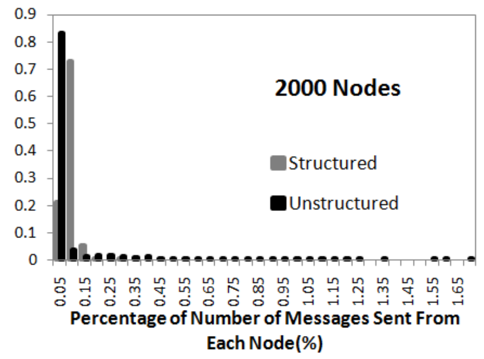


Figure 4: Histograms of the percentage of total messages sent from each node

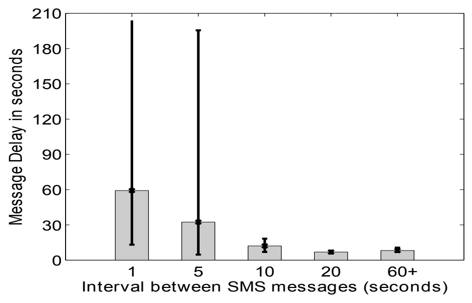


Figure 5: SMS message delays

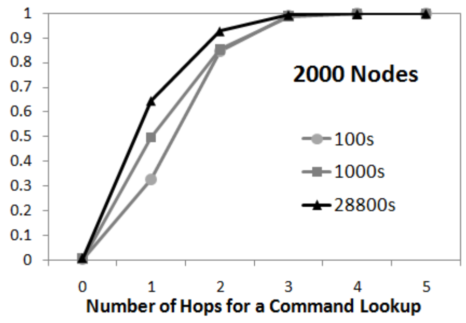


Figure 6: CDFs of the number of hops for a command lookup under different mean lifetimes (in sec)

ing and spamming are not time-critical, bots do not have to pull commands all at the same time. To further minimize the number of messages sent/received, each bot can be restricted by a threshold. If the number of messages reaches the threshold, the bot will stop sending/receiving messages. The threshold can be customized depending on the usage pattern of SMS on that particular phone. If a bot has frequent normal SMS messaging behavior, its threshold of allowing bot communication could be high since this phone is very likely to use a SMS plan and a few blended malicious messages are less noticeable.

Figure 4 shows the histograms of load distribution on each node, which is the percentage of total messages each node accounts for during the entire simulation. In the 200-node botnet, 76% of nodes in the structured botnet each accounts for 0.75% – 1.25% of total messages sent, whereas in the unstructured one, the percentage values are spread out among different nodes ranging from 0.10% to 6%. The average percentage for the structured one is 1.02% and for the unstructured is 1.01%. To gauge the load-balancing more accurately, we calculated a metric defined as: $\sum_{i=1}^{n} |p_i - \overline{p}|$ $(*)$, where n is the total number of nodes, p_i is the load percentage at node i, and $\overline{p}$ is the average percentage across all nodes. The $(*)$ values for the structured and the unstructured are 13.40% and 55.89%, respectively. In the 2000-node case, all nodes' percentages in the structured botnet range from 0.05% to 0.25% while those in the unstructured botnet are distributed within 0.05% – 1.65%, although the average percentages for both the structured and the unstructured are 0.07%. The metric $(*)$ values for the structured and the unstructured are 23.73% and 145.48%. The unstructured case varies more in load distribution leading to poor load-balancing, probably because Gia uses schemes to direct most queries to a few nodes—forming hub nodes.

To estimate the actual delay of locating a command in our mobile botnet, we measured one-hop latency by sending SMS messages between two smartphones. We implemented a SMS send/receive utility on the Android platform and installed it on two G1 phones: one connected to T-mobile and the other to AT&T. The software continually sent out and received SMS messages between two phones and recorded the exact timestamps. The intervals between two consecutive SMS messages were chosen from 1 second to tens of minutes and the message contents were also randomly generated with various lengths to simulate the realistic SMS usage. During the entire experiment, we sent out a total of 138 SMS messages and collected the corresponding message delays, i.e., the difference between the time sending a message from one phone and the time of receiving that message from the other phone. Figure 5 depicts min/max/average message delays based on different sending intervals (sending rates). We can see that when SMS messages are sent frequently, the message delays vary a lot and have high average values. Take 1 second as an example. Under this interval, delays range from 15 to 205 seconds with an average of 60 seconds. Similar delay patterns occur when the interval is 5 seconds. The general trend is that as intervals become larger, both delay average and variance drop, and that when the interval is greater than 60 seconds, the delays become stable.

Since mobile attacks such as confidential information stealing (especially related to credit card, account number, etc.) are not time-sensitive, bots can send messages at relatively long intervals to shorten the delay and avoid detection. Using a greater than 1 minute sending interval's delay, we now estimate the total delay for a command-lookup. Under structured Kademlia which uses iterative search, the estimated delay is given by $AverageTotalDelay = 2 \times AverageHops \times AverageOneHopDelay$. When it comes to unstructured Gia which employs recursive search, the equation should be the same. By plugging in the data we obtained, the estimated command-lookup delay is 17 seconds for the structured and 36 seconds for the unstructured in the 2000-node botnet.

The delays seem to be large compared to that of IP-based connections. As briefly mentioned before, our current design does not opt for IP-based C&C or existing IP-based P2P networks for the following reasons. First, some smartphones may not have data plans, not always accessible to the Internet. Second, for smartphones with the Internet access, they can initiate connections to retrieve commands from designated servers but are likely to suffer from a single-point-of-failure. To work in a decentralized P2P fashion, mobile bots should be able to accept incoming connections without any difficulty, which presents a challenge due to private IPs used in most scenarios. A possible solution is to obtain assistance from a third-party such as a mediator server or a rendezvous server, adding complexity to the C&C. Since SMS is ubiquitous across all mobile phones, using SMS as the C&C channel to construct a P2P structure is a feasible and reliable solution for mobile botnets. As future work, we can incorporate IP-based command-transfer into our botnet. For mobile bots without network access, they transmit C&C exclusively via SMS messages. For bots with network access, they can pull commands from an IP-based P2P network. Such a network consisting of PCs can be either constructed by the botmaster or part of an existing P2P network. Doing so may help reduce the message overhead and the delay.

In summary, our simulation results show that the structured architecture outperforms the unstructured one in terms of total number of messages sent, hops needed and delays for a lookup as well as load-balancing, although both the original protocols—Kademlia and Gia—have already been tailored to our mobile botnet's needs through several modifications. Thus, the structured architecture is indeed better suited for our mobile botnet.

4.2 Effect of Churn Rates

Now that we have chosen the structured architecture, we would like to see the effect of different mean lifetimes or churn rates on the number of hops for a command lookup, which directly affects the delay of locating a command. To see the trend, in a 2000-node botnet, we varied the mean lifetimes—100s, 1000s and 28800s. The higher the mean lifetime, the lower the churn rate. Presumably, a large mean lifetime indicates a relatively stable network in which fewer steps are needed to locate a command. This assumption is verified in our simulation. We can see that in Figure 6, differences, though minimal, exist among the three CDFs. With the mean equal to 100s, the average number of hops is 1.8; with the mean equal to 1000s, the average reduces to 1.7; with the mean equal to 28800s, the average decreases further to 1.4. It turns out that a higher churn rate does not degrade much of the lookup performance.

4.3 Can Disguised C&C Messages Go Through?

One concern with our spam-like C&C messages is what if they are filtered and deleted by the service providers without reaching the recipients, which might be the only effective way to mitigate SMS spam (spam-filtering at the end device is not useful as the recipient needs to pay for the messages already). According to some sources [27, 33], mobile carriers do not automatically block SMS spam because there is no spam folder with SMS so that accidental deletion of legitimate messages from the carrier's side cannot be recovered by the users. Also, senders are presumably charged for these messages unlike emails. To confirm this, we ran experiments to see whether carriers will let our spam-like C&C messages pass through. Table 1 shows the spam templates for C&C, which are typical spam messages. The random strings highlighted in grey are variables such as passcodes and node IDs. We tried two methods to send them: web-based and smartphone-based. For the first method, we sent all messages twice to an AT&T phone via free texting service at Text4Free.net and txt2day.com respectively. 100% of them reached the designated phone. For the second method, we wrote an application and installed it on an AT&T Samsung Captivate phone running Android OS 2.2. This application automatically sent the spam messages 5 times at different times of a day to another AT&T phone. The application also kept track of whether a message was sent successfully. Out of the 50 messages, 48 messages were sent and delivered to the target phone and 2 messages failed to be sent due to some generic failure at sender's phone that had nothing to do with the carrier. Although we were not able to thoroughly test every possible spam message on different networks, our experimental results were in line with the aforementioned reports and we believe that as few spam-fighting mechanisms are in place, our disguised C&C messages can safely go through the network.

4.4 How Do SMS C&C and P2P Structure Become One?

Having an impression of how SMS transmits C&C messages and how a structured P2P topology fits our mobile botnet, one may want to know in detail the way we integrate both into the mobile botnet. We now use a simplified example (Figure 7) to illustrate the command publish and search process. For illustration purpose, node IDs and data items' keys are 4-bit long, and SMS messages transmitted are not disguised as spam. In this figure, node 1111 wants to publish certain data—a command—under the key 0111. Note that in Kademlia, data items are stored in nodes whose IDs are close to data items' keys. To locate such nodes, node 1111 first sends SMS messages to nodes in its hard-coded node list; these nodes help to obtain nodes closer to the target from their node lists. The process continues till no closer nodes could be found (this process is omitted in the figure). Finally, node 1111 finds the closest node 0110 ($0110 \oplus 0111 = 0001$), so a publish message containing the command's key (0111), the encrypted command (XXXX) along with a passcode (8888) is sent to node 0110. After verifying the pre-defined passcode and command, node 0110 stores this information so that later any node requests the command associated with key 0111 it is able to return this command. As for the search process, it is similar to the publish process described. Node 0000 looks up a command associated with key 0111 and it has to find the node whose ID is closer to this key. Node 0000 first asks node 0010; node 0010 points it to node 0100; node 0100 provides the closest one, node 0110. Node 0000 contacts node 0110 to request the command.

5. DISCUSSION ON COUNTERMEASURES

Although we have focused on the design of a stealthy and resilient mobile botnet, we would like to discuss potential defensive strategies and challenges in using these techniques.

Similar to the patching mechanism in the PC world, to prevent malicious code from infecting mobile devices by vulnerability exploits, OS vendors and software providers need to push patches to end devices in a timely manner. Certification (only approved applications can be installed) is also an important security measure, but it is far from being perfect as some malware has been able to get around [34] as a disguised harmless application. To nip the mobile malware in the bud, additional protection features are necessary. For example, Kirin [35] is designed for the Android-platform whose certification process is not stringent; it provides application certification at install time using a set of predefined security rules that determine whether the security configuration bundled with an application is safe. With the aid of Kirin, users may be more cautious while installing applications.

Host-based approaches that detect malware at runtime could also serve as a solution. Signature-based detection is effective but cannot handle unknown or polymorphic malware. We prefer use of behavior-based detection. Since our bots send SMS messages stealthily without the user's involvement or awareness, the detector could first characterize the normal process of sending SMS messages by a system-call state-diagram and then keep monitoring the system calls that generate outgoing messages to see if there is any deviation from the normal behavior. To detect incoming C&C messages, the detector needs to know the encoding scheme probably through binary analysis so that it can intercept and delete malicious messages before any application's access. However, the botmaster can apply advanced packing and obfuscation techniques to make the binary analysis harder, and periodically update the spam templates as well as the mapping between them and corresponding commands. In addition, host-level detection is susceptible to compromise by the malware, and consumes much resource.

Deploying detection schemes at the SMSC is another possible solution. Compared to the host-level detection, this centralized approach can acquire a global view of all phones' SMS activities, although the information of each phone might be limited. As mentioned before, simply filtering out spam will not effectively cut off the botnet's C&C. The reason is that even if carriers dump spam-like SMS messages into a spam folder like email service providers do, spam messages will still reach target phones, stay at a less noticeable place—the spam folder and get commands executed. Black-listing and SMS sending/receiving rate-limiting may be difficult because our design attempts to minimize the total number of messages sent/received and to balance the load on each bot. As always, matching signatures extracted from known bots' messages can be bypassed by malicious messages with completely new formats or contents. To differentiate between mobile bots and normal phones, the detector at the SMSC needs to extract more distinctive features from SMS traffic patterns. For example, normal phones may have regularities in whom they send messages to and the sending frequency

Table 1: Spam templates with variable fields in grey

1	Your paypal account was hijacked (Err msg: NzkxMjAzNDlxODExMDUyM183Mz). Respond to http://www.bhocxx.paypal.com using code Q3MDk2NDUyXzEyMzQ1Njc4
2	Free ringtone download at www.myringtone.com, using username VIP, password YTJiNGQxMWw to log on
3	Dear Customer, your order ID dWFuaWRpb3Q is accepted. Please visit: www.xajq.apple.com for more info
4	Your business is greatly appreciated and we would like to award you a free gift. http://www.protending.com/ebay/anVzdDRmdW4
5	To confirm your online bank records, follow the link https://login.personal.wamu.com/logon.asp?id=YWhhaGFoYWg
6	Hey, come on - Purchase G.e.n.e.r.i.c V I A G R A! http://www.WQ9.wesiwhchned.com/default.asp?ID=MTA5MzIxMnc
7	Citi Users: This is an important step in stopping online fraud. Please verify your account at https://www.citi.com.Y2Nzc3Vja3M/verify/
8	Hey alice, I forgot to tell you yesterday that the password to that account(MDkyMzkxMDM0OTgxMjAzN) should be 183MzQyNjIwOTM5XzUxOTQwMTI5
9	Don't miss the chance to win an iPhone 4. Go to www.apple.hak/index.asp?id=OTAxMjc1MjM4OTExMTIzOD, password: QyXzQxNDMyMTg3MzlfNjQ4MTkyMDQ
10	Guess who is tracking your location info? Log on to www.whoistrackingme.com/index.asp?num=YWxqc2hmdy0

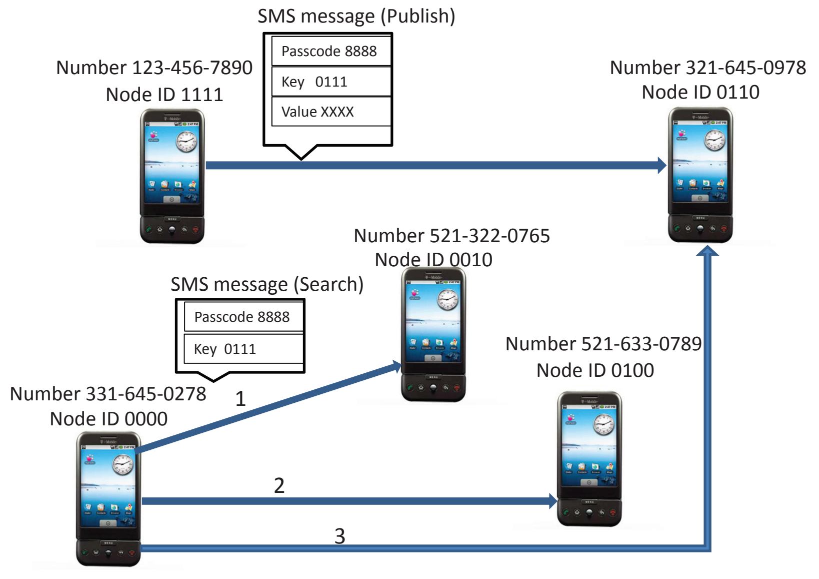


Figure 7: Publish and Search

[36]. The detector can therefore build normal profiles and identify anomalies accordingly. The detector may also adopt a high-level view for detection. As our botnet utilizes a P2P architecture, the resultant network topology stemmed from SMS activities may be different from that formed by benign phones, given the fact that P2P applications are rare in today's mobile phone networks.

6. CONCLUSION

As smartphones are getting more powerful, they become potential targets of profit-driven attacks, especially botnets. In this paper, we presented the design of a mobile botnet that utilizes SMS to transmit C&C messages and a P2P structure to construct its topology. Using simulation, we compared two types of P2P architectures—the structured and the unstructured—based on several metrics critical to the mobile botnet performance. We found that the modified Kademlia—a structured architecture—is more suitable for our botnet in terms of message overhead, delay, and load-balancing. We also investigated possible ways to counter the mobile botnet threat. As future work, we plan to combine SMS-based C&C and IP-based C&C utilizing existing DHT or P2P networks. Since our current work focuses on the aspects of feasibility and efficiency in botnet design, we would also like to measure robustness, i.e., how our botnet performs under different detection and mitigation strategies.

Acknowledgments

The work reported in this paper was supported in part by the US National Science Foundation under Grant No. CNS 1114837.

7. REFERENCES

[1] "Google yanks over 50 infected apps from android market," http://www.computerworld.com/s/article/9212598/Google_yanks_over_50_infected_apps_from_Android_Market.

[2] SymbOS.Exy.A, http://www.symantec.com/security_response/writeup.jsp?docid=2009-022010-4100-99.

[3] Ikee.B, http://www.symantec.com/security_response/writeup.jsp?docid=2009-112217-4458-99.

[4] "More droiddream details emerge: It was building a mobile botnet," http://www.readwriteweb.com/archives/droiddream_malware_was_going_to_install_more_apps_on_your_phone.php.

[5] RootSmart, http://www.csc.ncsu.edu/faculty/jiang/RootSmart/.

[6] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," in *HotBots'07*.

[7] P. Wang, L. Wu, B. Aslam, and C. C. Zou, "A systematic study on peer-to-peer botnets," in *ICCCN 2009*.

[8] G. Starnberger, C. Kruegel, and E. Kirda, "Overbot - a botnet protocol based on kademlia," in *SECURECOMM 2008*.

[9] C. R. Davis, S. Neville, J. M. Fernandez, J.-M. Robert, and J. McHugh, "Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures?" in *Proceedings of 13th European Symposium on Research in Computer Security (ESORICS'08)*.

[10] K. Singh, A. Srivastava, J. Giffin, and W. Lee, "Evaluating email's feasibility for botnet command and control," in *Proceedings of 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'08)*.

[11] A. Nappa, A. Fattori, M. Balduzzi, M. Dell'Amico, and L. Cavallaro, "Take a deep breath: A stealthy, resilient and cost-effective botnet using skype," in *Proceedings of 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'10)*.

[12] D. Dagon, G. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*.

[13] A. Bose and K. G.Shin, "On mobile viruses exploiting messaging and bluetooth services," in *Proceedings of the 2nd International Conference on Security and Privacy in Communication Networks (SecureComm'06)*.

[14] R. Racic, D. Ma, and H. Chen, "Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery," in *Proceedings of the 2nd International Conference on Security and Privacy in Communication Networks (SecureComm'06)*.

[15] W. Enck, P. Traynor, P. McDaniel, and T. L. Porta, "Exploiting open functionality in sms-capable cellular networks," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*.

[16] P.Traynor, M.Lin, M.Ongtang, V.Rao, T.Jaeger, P.McDaniel, and T.L.Porta, "On cellular botnets: Measuring the impact of malicious devices on a cellular network core," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'09)*.

[17] K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee, "Evaluating bluetooth as a medium for botnet command and control," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2010)*.

[18] J. Hua and K. Sakurai, "A sms-based mobile botnet using flooding algorithm," in *The 5th Workshop in Information Security and Privacy (WISTP'11)*.

[19] C. Mulliner and J.-P. Seifert, "Rise of the ibots: 0wning a telco network," in *The 5th IEEE International Conference on Malicious and Unwanted Software (Malware), 2010*.

[20] G. Weidman, "Transparent botnet command and control for smartphones over sms," in *Shmoocon 2011*.

[21] "Htc bluetooth vulnerability," http://www.cio.com/article/497146/HTC_Smartphones_Left_Vulnerable_to_Bluetooth_Attack.

[22] C.Mulliner and C.Miller, "Fuzzing the phone in your phone," in *BlackHat Security Conference, 2009*.

[23] SMS, http://en.wikipedia.org/wiki/SMS.

[24] "China cracks down on sms spam," http://www.redherring.com/Home/19081.

[25] textPlus, http://www.textplus.com/.

[26] Textfree, http://itunes.apple.com/us/app/textfree-unlimited-send-text/id305925151?mt=8.

[27] "Gsma launches sms spam reporting service," http://www.pcworld.com/businesscenter/article/192469/gsma_launches_sms_spam_reporting_service.html.

[28] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS, 2002*.

[29] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutellalike p2p systems scalable," in *ACM SIGCOMM, 2003*.

[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM 2001*.

[31] OverSim, http://www.oversim.org/.

[32] "Sms market statistics 2009," http://www.massmailsoftware.com/blog/2010/04/sms-market-statistics-2009-know-your-customer/.

[33] "Mobile_phone_spam," http://en.wikipedia.org/wiki/Mobile_phone_spam.

[34] "Researcher says app store open to malware," http://www.iphonealley.com/current/researcher-says-app-store-open-to-malware.

[35] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security (CCS '09)*.

[36] G. Yan, S. Eidenbenz, and E. Galli, "Sms-watchdog: Profiling social behaviors of sms users for anomaly detection," in *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID'09)*.