How to Construct a Mobile Botnet?

Yuanyuan Zeng, Xin Hu, Kang G. Shin
{gracez, huxin, kgshin}@eecs.umich.edu
The University of Michigan, Ann Arbor, MI 48109-2121, USA

I. Introduction

Botnets are one of the most serious security threats to the Internet and the personal computer (PC) world, but they have not yet caused major outbreaks in the mobile world. Nevertheless, attacks on mobile networks and devices have recently grown in number and sophistication. With the arrival of smartphones such as the iPhone and Android-based phones, there has been a drastic increase in downloading and sharing of third-party applications and user-generated content, making smartphones vulnerable to various types of malware. Smartphone-based banking and payment services have also become popular without protection features comparable to those on PCs, enticing cyber crimes. As smartphones handle more personal data and gain more computing power and capabilities but have little security protection, early naive attacks targeting mobile devices have become more sophisticated. Since the appearance of the first 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 functionalities. Recent mobile malware demonstrates more sophisticated behavior. SymbOS.Exy.A trojan was discovered in February 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 recent Ikee.B worm appearing late November 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 issue commands to and coordinate the infected devices to launch large-scale attacks. Considering this potential, botnets will likely soon become a serious threat to smartphones.

In this paper, we propose the design of a mobile botnet that makes the most of mobile services and is resilient to disruption. The goal of our 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 design makes it possible for people to investigate and develop new countermeasures before mobile botnets become a major threat. Within our mobile botnet, all C&C communications are done via SMS messages since SMS is available to almost every mobile phone and can delay message delivery for offline phones. To hide the identity of the botmaster, there are no central servers dedicated to command dissemination that is easy to be identified and 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.

II. Mobile Botnet Design

We now briefly describe the design of a proof-of-concept mobile-botnet, which 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.

A. Propagation

The main approaches for propagating malicious code to smartphones are user-involved propagation and vulnerability exploits, both of which can get our mobile bots installed onto smartphones. In the first category, the most popular vector is social engineering. Spam emails and MMS/SMS messages with malicious content attachments or embedded links pointing to malicious websites, can easily find their way into a mobile phone's inbox. The advantage of such schemes is that they can reach a large number of phones. Another user-involved propagation vector can be Bluetooth utilizing 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 difficult to find vulnerabilities. Once launched to 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 abound, vulnerability exploits will play a major role in mobile malware propagation.

B. Command and Control

In our mobile botnet, we utilize SMS as our Command & Control channel so that compromised mobile bots communicate with botmasters and among themselves via SMS messages. Our goal is to let a phone that has installed our bot's code conduct activities according to commands received in SMS messages without being noticed by the user if possible. In our design, every compromised phone has a key. Only by including this key into the SMS messages, can other phones transmit C&C information to this particular phone. Upon receipt of a SMS message, this phone searches for its key 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. One challenge here is

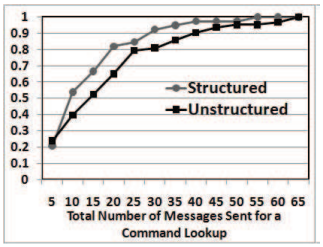

Fig. 1. CDFs of the total number of messages sent for a command-lookup

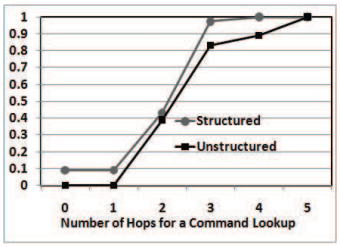

Fig. 2. CDFs of the number of hops needed for a command-lookup

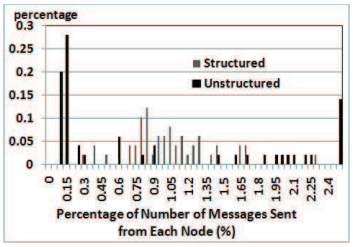

Fig. 3. Histogram of the percentage of total messages sent from each node

how to make C&C SMS messages appear to be harmless so that users cannot figure out the malicious content. Our solution is to make a command-embedded SMS message look like a common message such as a spam. To disguise commands, a simple word mapping technique can be utilized. For example, one disguised SMS message a bot receives may read: "Free message Tone: Free ringtones download at www.xyz.com". "ringtones" corresponds to the command "GetContactList". Our crafted messages look like advertisements or even spam, familiar to today's phone users, so they are likely to be ignored by users. Without monitoring phone behaviors or reverse engineering, defenders may have a hard time figuring out the mapping between regular words and commands.

C. Mobile Botnet Topology

Similar to botnets in the PC world, a mobile botnet can be either structured in a traditional centralized way or in a newly-emerged decentralized P2P fashion. A centralized topology is relatively easy to be implemented but not resilient to disruption. 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 are similar to the traditional centralized botnet architecture and hence vulnerable to the central-point-of failure. In decentralized but structured P2P networks, 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 which ensure that any peer can efficiently route a search to some peer that has the desired content. One notable implementation is Kademlia. 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. One design for this purpose is Gia. Both structured and unstructured P2P architectures can be modified to suit the need for our mobile botnet because their decentralized nature hides the botmaster's identity. However, 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 topologies on Kademlia and Gia, respectively, for comparison.

III. Preliminary Results

To compare the two modified P2P structures, we used OverSim, an open-source overlay network simulation framework. Our metrics to measure performances are: the total number of SMS messages sent from all nodes involved for a command lookup, the percentage of total number of SMS messages sent by each node during the entire a-command-lookup period, and the number of hops needed for a command lookup. These metrics can reflect how well each architecture meets the requirement of our mobile botnet, namely, minimizing the number of SMS messages sent and load-balancing. In the structured network simulation, we simulated 200 nodes running the modified Kademlia protocol generating about 100 lookup queries. We did the same in the unstructured network simulation. Figure 1 shows the CDFs of total number of SMS messages sent for a lookup. As we can see, under the structured architecture about 80% lookups generate fewer than 20 messages sent, while under the unstructured architecture 65% lookups can do so. The average number of messages sent is 15 for the structured and 20 for the unstructured, respectively, showing that unstructured architecture requires more SMS messages sent for a lookup. Figure 2 are the CDFs of the number of hops needed to reach a targeted command. For the structured architecture, 97% lookups can be done within 3 hops. The number for the unstructured one is 5 hops. The above two observations are understandable because in a structured network, data items are placed at deterministic locations requiring fewer number of messages and hops to reach a target. Figure 3 presents the histogram of load distribution on each node, which is the percentage of total message sent each node accounts for during the entire simulation. It turns out that 76% nodes in the structured network each account for 0.75% – 1.25% of total messages sent whereas in the unstructured one the percentages values are spread out among different nodes ranging from 0.10% to 6%, showing that the latter varies more in load distribution. Gia uses a few schemes to direct queries to high-capacity nodes, which may be the reason for its poor load-balancing. To sum up, the structured architecture outperforms the unstructured one in terms of total number of message sent, hops needed and load-balancing. Thus, the structured architecture is more suitable for our mobile botnet.