Rise of the iBots: 0wning a telco network

Collin Mulliner and Jean-Pierre Seifert
Security in Telecommunications
Technische Universität Berlin and Deutsche Telekom Laboratories
D-10587, Berlin, Germany
collin,jpseifert@sec.t-labs.tu-berlin.de

Abstract

The undoubted success of very powerful and pervasively IP enabled cellular phones raises the obvious question whether the cellular world will also enter a severe security crisis like the PC itself. Moreover, this serious question is amplified through the use of new Open and even Web-OS oriented phone platforms. Considering the most dangerous security threat which might be given in the form of cellular botnets, a very recent paper measured already the potential impact of such a hypothetical botnet. While this theoretical work of Traynor et al pointed out some intrinsic challenges of a cellular botnet, they emphasized the significant threats of such botnets for the core network.

Unfortunately, this paper shows that this new attack vector is quite real. Indeed, we describe a cellular botnet and our solutions to the cellular challenges. In addition to that we also sketch and evaluate our real implementation on the world's most popular smart phone - the iPhone. Our devastating results, clearly ring an alarm for urgent cellular phone protection mechanisms.

Keywords: *Smart Phones, Malware, Botnet, Worms, P2P, SMS.*

1 Introduction

It is clear that mobile and smart phones are the future of personal computing — just as the personal computer was many years ago, and additionally also their IP connectivity follows this trend, i.e., more and more phones have a pervasive IP connectivity, i.e., WiFi, GPRS, EDGE, 3G, etc.

Sadly enough, this success of an (almost) single platform architecture (mainly MS Windows-based) installed on a nearly infinite number of IP-connected PC's has led to an unforeseen security crisis for the PC platform which culminated in one of the largest security threats for the IP world: large scale criminal botnets, cf. [1].

Given this similarity between the PC platform and the emerging (and dominating) smart phone platforms like iPhone, Android (Google) phone, and Windows Mobile, it is a legitimate question whether the cellular world could also enter a severe security crisis like the PC itself? Especially, we are interested in answering this question with regard to the existence of practical and functional cellular botnets.

The practical existence question is especially important, as the theoretical threat of cellular botnets was just recently investigated and emphasized by Traynor et al [1] — simply assuming the theoretical existence such botnets. Moreover, their focus was on the security impact for the fragile and complicated cellular architecture on which we are all depending on — day by day. Their main research result showed that a relatively small number of cellular bots can already force the collapse of a targeted victim core network. Interestingly, they smoothly concluded that the challenges for a functional and large-scale cellular botnet are noteworthy and that such botnets might not be too quickly seen in the wild.

Thus, the present paper perfectly complements their research from the cellular platform side, as we solve their cellular challenges and describe the architecture and even the implementation of a practical and simply realizable cellular botnet for the iPhone.

Especially, we show how we designed, implemented and evaluated an iPhone-based mobile botnet. We did this to understand what it takes to build a botnet that resides on mobile phones and on a mobile phone network. We think this is an important first step in order to start thinking about urgently needed counter measures for mobile phone botnets.

We started by following the current developments in botnet research and built a peer-to-peer (P2P) based mobile phone bot. The P2P bot was quite simple to design and implement, and, therefore, presents an easy path for an unskilled botmaster.

Diving deeper in to the specifics of mobile phone botnet we further created a Short Message Service (SMS) [2] based bot. A bot that can be controlled entirely via SMS. We further improved our SMS bot by turning it into a hybrid of SMS and HTTP in order to reduce the number of SMS messages that need to be sent for controlling the bots.

In the end we showed how powerful a mobile phone botnet could be if one combines the P2P with the SMS-HTTP hybrid approach. A bit frightened by the success of our cellular bots we recognized that this hybrid bot would be very hard to be detected and stopped if controlled by a skilled botmaster. Thus, we also stopped our further research at this point as our main questions and motivations were completely solved.

Our research produced the following main contributions:

- We showed a cellular botnet architecture and even evaluated it with several practical implementations.
- We solved the environmental challenges of such cellular botnets.
- We implemented and evaluated a P2P-based command and control mechanism for mobile phone botnets. Our bot implements the Kademlia P2P protocol and joins the Overnet network.
- We designed, implemented, and evaluated multiple SMS-based C&C mechanisms. The SMS approach raises the bar for the anti-botnet community.
- We created communication strategies for mobile phone-based botnets. The strategies are designed to increase the stealthiness of mobile phone botnets.

The rest of this paper is structured in the following way. In Section 2 we show how easy it can be to hijack many thousand iPhones using the Internet. Section 3 discusses the intrinsic challenges that cellular networks pose for botnets. In Section 4 we present our command and control mechanisms for mobile botnets, while Section 5 continues elaborating on our communication strategies for mobile phone botnets. Eventually, Section 6 details our proof-of-concept implementation of our mobile bots including a self-critical evaluation. Finally, Section 7 draws some important conclusions.

2 Howto hijack many thousand iPhones

In November 2009 somebody exploited the facts that jailbroken[1] iPhones get a default root password assigned, often have the secure shell daemon (sshd) installed, and get an public IP address assigned to create a mobile phone worm. The worm was named Ikee.A [3] and infected around 21.000 iPhones within two weeks by simply copying itself via secure copy (part of ssh) from iPhone to iPhone. Later somebody added a very simple command and control mechanism to Ikee to turn it into a botnet, this botnet was called Ikee.B. The command and control mechanism was simply polling a webserver to download and run a shell script.

This example shows how easy it is to hijack many thousand mobile phones through the Internet without any special knowledge about mobile phones or mobile phone security. Therefore we believe that this was just a first taste of what will happen in the future. Also if you look at vulnerabilities like [4] through witch an iPhone could have been hijacked through SMS it becomes clear that mobile botnets are sure to come to existence.

3 Cellular Challenges

Mobile and smart phones present a number of challenges that need to be meet in order to design a botnet that is able to exist and thrive in the mobile phone environment. The problems range from: 1) limited run time due to the use of batteries as the power source, to 2) connectivity problems due to the absence of public IP addresses, 3) constant change of connectivity, 4) the problem of diversity of mobile phone platforms, and 5) the costs of mobile communication. In the following we will discuss these problems in further detail.

3.1 Absence of public IP addresses

Public IP addresses are needed for direct communication of bots. Without public IP addresses an intermediate communication hub is required, unfortunately most mobile phone service operators put their customers behind a NAT[2] gateway and thus the devices are not directly reachable. Although the attack vector presented in Section 2 shows the picture of a

[1]http://en.wikipedia.org/wiki/Jailbreak_(iPhone_OS)

[2]Network Address Translation

mobile operator providing public IP addresses to customer phones, this is not the common case. Even if a mobile operator chooses to provide public IPs to his customers, mobile phones will still sit behind a NAT gateway for many hours during the day. This is the time the user spends at home where his phone is connected to the local wireless lan in order to benefit from higher Internet speeds and to lower the services charges by using his DSL or cable line.

3.2 Platform diversity

The size of a mobile phone botnet will be relatively small compared with botnets based on hijacked desktop computers. The main reason for the size limitation is related to the diversity of mobile phone platforms, therefore we think each mobile phone botnet will be targeted towards a specific device, platform, or platform version. Due to the small number of bots in a mobile phone botnet it will be hard and maybe impossible to build an independent communication infrastructure such as P2P network that exclusively consists of hijacked mobile phones.

3.3 Constant change of connectivity

Constant change of connectivity is something that is normal for a mobile phone compared with a desktop computer that is connected to the Internet via a DSL line. The connectivity of a mobile phone changes for many reasons. First, mobile phones move around the physical world. Their wireless connection comes and goes depending on the position of the device and the available type of mobile network capabilities. GPRS vs. EDGE vs. 3G. Individual phones might be disconnected for a relatively long time even though the phone itself is powered up. Second, is the earlier mentioned use of local wireless networks, this again would change the connectivity properties of a mobile bot. Therefore, a mobile phone botnet is likely to be very unstable in terms of the size and the kind of network connectivity of an individual node. Table 1 shows the connectivity times of the mobile phones of the authors and some of their colleagues.

Connectivity	Hours
WiFi	Early morning (still at home)
GSM/3G	Morning (travel to work/school)
GSM/3G	Day time (while at work/school)
WiFi	Early evening (back at home)
GSM/3G	Early Night (going out)
WiFi	Night (bed time)

Table 1. Connectivity times.

3.4 Communication Costs

In the world of mobile telecommunication most types of communication result in costs for the ones who communicate. These costs have to been taken into account when designing a botnet Command and Control mechanism since a significant rise of the phone bill will lead to investigation of the cause and thus may lead to detecting the bot infection. Especially interesting is SMS, since here each message sent costs money. Also deepening on the type of mobile phone contract SMS messages can be completely free when sent to subscribers on the same network. Further things such as roaming has to be considered since the charges for communication are significant higher during roaming. Mobile-data usage might be disabled during roaming but services like SMS still work. A mobile phone bot therefore might need to query the roaming status in order to fit in with the other applications running on the device.

4 C&C for Mobile Botnets

Command and Control (C&C) is the most important part of a botnet. For the botmaster it is the path to deliver commands to his botnet and for the defender it is the major attack vector in order to dismantle and destroy a botnet. The C&C channel therefore has to be carefully designed to be reliable for command delivery as well as resilient against many kinds of attacks.

In [5] the authors use Bluetooth as the transport channel for Command and Control of their mobile phone botnet. We believe that a botnet based on local wireless communication will be large enough to be of any use for a botmaster, therefore, we focus our research on Internet and mobile phone network based C&C.

Over the past years there has been some major development in botnet C&C, earlier botnets used IRC (Internet Relay Chat) for C&C but today most botnets use some kind of P2P scheme for C&C [6].

In our work we have followed two major paths for C&C. First, we evaluated a P2P-based approach since this seems to be the current overall trend in botnet research. Also we did not create our own P2P network as suggested in [7]. The second path we followed is a SMS-based approach. *We chose SMS because we think that SMS communication is much harder to observe, analyze and disrupt by security researchers and* *the anti-botnet community, and, therefore, it is likely to be chosen as the C&C channel by knowledgeable botnet creators.* We actually designed two SMS-based C&C mechanisms to get a broader overview of the possibilities and problems of SMS-based bot communication.

In the following we will first discuss the P2P-based approach since it is a bit simpler than the approach based on SMS. Before we discus the actual C&C channel we briefly talk about the additional required features in order to secure the commands sent over the C&C channel.

4.1 Securing the C&C Communication

In order to protect the commands sent over the Command and Control channel from tempering all commands carry a digital signature using public-key cryptography. Further to prevent replay attacks, commands carry a sequence number. Only commands that carry a sequence number that is higher then the one from the last accepted command will be accepted as a valid command.

4.2 Peer-to-peer C&C

For our P2P-based approach we choose Kademlia [8] as the protocol and Overnet[3] as the P2P-network to join. We chose to rather join an existing network instead of creating our own because of the mobile phone related problems and issues that we discussed in Section 3. The main reason being the unavailability of a stable set of public IP addresses.

The basic design idea for our P2P-based Command and Control channel is to use the P2P network as a kind of rendezvous point. The P2P network is only used as a basic communication channel using the publish and search functionality of the distributed hash table (DHT). The botmaster publishes a `command` to the P2P network and the bots search for a specific key in order to retrieve the `command`. The publish and search functionality is solely based on functionally offered through the DHT, and, therefore, no actual file sharing functionally needs to be present on either the botmaster nor the bot side. Figure 1 shows a high level view of a P2P-based mobile phone botnet.

Battery consumption plays a very important role in the mobile phone world, therefore it is very important for a mobile phone bot to not drain the battery significantly. A significant battery drainage will otherwise lead to detection of the bot rather easily. Battery drain is mostly related to two operations, high CPU load and

[3]http://en.wikipedia.org/wiki/Overnet

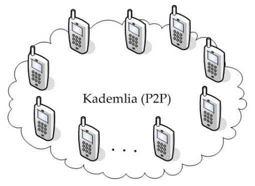

Figure 1. Kademlia P2P C&C.

heavy radio usage. Our main concern is the radio usage. In order to reduce the network activity of our bot it connects only briefly to the P2P network to search for the key that results in the command from the botmaster. After the search, the bot quickly disconnects from the P2P network. Initially we designed the bot in a way that it connects to the peer-to-peer network about every 15 minutes. Upon connection it waits until the connection has stabilized and is ready to fire search queries (this seems to take between 30-60 seconds). In a 20 second interval in searches three times for the rendezvous key and then disconnects. This communication pattern is very similar to a background email poll, and, therefore, should not cut to deep into battery consumption. For times where the botmaster needs faster responds times for his botnet he can issue a command that changes the time interval of connecting to the P2P network. The interval can of course also be increased for less battery consumption and lower responds times (for times where the botnet is not heavily used).

4.3 SMS C&C

In this section we present our two SMS-based C&C mechanisms. Both schemas are based on the fact that the botmaster has a complete list of bots or actually a list of phone numbers that correspond to the bots, at all times.

Below we will first provide a brief introduction to the Short Message Service, then we will discuss each SMS C&C schema and finally we will briefly talk about obfuscation of the C&C SMS messages.

4.3.1 The Short Message Service

is one of the basic services of the mobile phone network. SMS is used for text messaging by users and for background services that are not directly visible to the user. SMS supports transport of binary data, and, therefore, can be used to send arbitrary data such as Command and Control information for a botnet. Although SMS messages are limited to 140 octets each, we show that this is enough for a highly flexible and secure botnet communication. In this paper we will not discuss the details of the Short Message Service itself and will stick to the parts that are important for the design of the C&C channel.

The basics of SMS communication are. The sender only needs to know the phone number of the receiver in order to send him a message. To send a message, the sender encodes the phone number together with some flags and the payload in to the SMS PDU format and hands it over to the mobile phone modem using the GSM AT command set. The modem takes care about delivering the message to the mobile phone network. In the network SMS messages are handled by the Short Message Service Center (SMSC). The SMSC forwards the message to the receiving mobile phone. If the receiving mobile phone is switched off the SMSC buffers the message until the receiver is switched back on. The receiver, upon reception of the SMS, extracts the payload from the PDU. The payload is just a number of octets at the end of the PDU.

4.3.2 SMS-only C&C

In this scenario all communication from the botmaster to the botnet is carried out over SMS. There are a few exceptions such as an update of the bot software or data transfer back to the botmaster which are still carried out over IP. Sending SMS messages costs money in most cases (see our discussion on SMS in Section 3.4), therefore it does not make sense for the botmaster to send messages to each bot directly. In this section we describe our SMS-only communication schema. We separated the schema in four parts: infection, communication, repair, and management.

- *Infection* takes part in three steps. In the first step the bot-software is installed on the hijacked phone using a software or configuration vulnerability. In the second phase the newly installed bot sends an SMS message to its infector, the infector provides his own phone number during bot installation. The SMS is sent in order to determine the phone number of the new bot. Sending a SMS message is the only reliable way to determine the phone number of a mobile phone, since it is not necessarily stored on the SIM card or on the phone itself. In the third and last step the infector dumps his list of phone numbers of devices he infected to a drop-site to be collected by the botmaster.

- *Communication* takes place in a tree model as shown in Figure 2. Meaning the botmaster sends a command message to the root node of his botnet. Each individual bot forwards the message to all bots known to them.

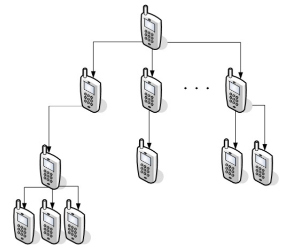

Figure 2. SMS only C&C.

- *Repair* takes place after the botmaster determined that the communication tree has broken at some point. In order to determine if the tree is intact once in a while the botmaster sends a *broadcast ping* that every node needs to answer. Nodes that fail to answer the ping message are removed from the tree. If a none leave node is removed all its sub-nodes are reassigned to other nodes. This is done by sending a message containing a list phone number(s) to an active node. Since the botmaster has at any time a complete overview of his botnet he can carry out a more intelligent repair phase by checking smaller sub-trees instead of the whole tree (the whole botnet) at once.

- *Management* of the botnet is required since it must be taken care of that a single node does not have too many direct sub-nodes. Each direct sub-node will require one SMS message to be send to in case of a message being forwarded. Further if a node with many direct sub-nodes disappears all the direct sub-nodes need to be moved to another node, leading in more SMS messages being sent.

In the ideal case the botmaster only needs to send out one SMS message to reach every node in the botnet, also he might not even need to send the message himself but rather have a hijacked phone send the initial message.

The SMS only design has a weak point, that is the existence of node lists (phone numbers) in most of the bot hosts. Therefore, making it easy for an attacker or anti-botnet researchers to warn the individual owner of an infected phone by simply sending him an SMS message. In order to partially prevent this from happening and to improve and ease the management and repair steps we designed a SMS-HTTP hybrid communication schema. This schema is discussed in the next Section.

4.3.3 SMS-HTTP hybrid

After realizing that a SMS-only-based Command and Control channel bears certain problems and the fact that IP communication is still required to accomplish any meaningful data transfer we designed a SMS-HTTP hybrid C&C channel for our mobile phone botnet.

The SMS-HTTP hybrid design additionally improves the SMS-only design in following ways. First, it removes the necessity to keep information about the botnet at each of the nodes. Therefore, making it a bit more resilient against attacks. Second, it eases the botnet management and repair by moving these task from the botnet to the botmaster. Third, it splits up the botnet in to multiple subnets and thus makes it harder to be detected.

The hybrid schema shares many properties of the SMS-only schema. The infection part of the hybrid schema works in exactly the same way. The only difference is that both, the newly infected bot and the infector, do not store the phone number of each other. The newly infected bot deletes the phone number of its infector after sending him the SMS message to determine his own phone number. The infector deletes the phone number of the newly infected phone right after dumping it at the drop-site in order to be collected by the botmaster.

The basic idea of the hybrid schema is to split the communication in to a HTTP and an SMS part. Command SMS messages are pre-crafted by the botmaster and are uploaded as encrypted files to some website. The URL to these files are then sent to random bots of the botnet via SMS. The bots download and decrypt the files and sent out the pre-crafted SMS messages. The encryption key is part of SMS message that contains the URL to the file. Figure 3 shows the three steps of the SMS-HTTP hybrid communication. In a decent sized botnet the first round of pre-crafted SMS messages could again contain URLs to another batch of pre-crafted command messages. Due to the fact that in the SMS-HTTP hybrid there is no fixed structure through that all communication is happening it is quite hard to determine if a botnet is active on a mobile phone network by just looking at the SMS traffic.

The repair part of the hybrid schema works in the same way as the repair part of the SMS-only schema. The botmaster has to regularly probe each bot to determine if it is still part of the botnet. The difference is that the botmaster does not need to move individual bots around the botnet in order to keep them in the working communication tree. Hosts that are no longer part of the botnet are just removed from the global bot list and thus are not considered the next time a command is sent out to the botnet. The management part as such does not exist in the hybrid design.

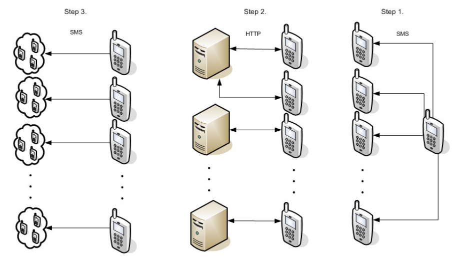

Figure 3. SMS-HTTP hybrid C&C.

4.3.4 Obfuscation

one idea for obfuscation is to encrypt all C&C SMS messages with a symmetric key. The key would be globally known by all bots, and, therefore, it will not prevent reverse engineering or off-line command analysis. But it will make life much harder for the mobile operators to filter out the messages, since they can not really tell what kind of message they see. To prevent hard coding of the key in to any IDS and anti-virus system there will be regular key updates. The key updates must be frequent enough so that C&C message parsing is required in the IDS. This will make it more costly to keep track of the botnet.

5 Communication Strategies

Communication is the most important part of a botnet, especially on a mobile phone since mobile phones have limited resources. A battery that drains faster than it used too, a high phone bill, slow or clogged 3G data can easily lead to detection and removal of the bot. In this Section we discuss our ideas for how a mobile phone botnet should carry out it's communication in order to stay hidden and still have maximum functionality.

5.1 IP-based communication

Even with a SMS-based C&C channel a bot still requires IP-based communication in order to transport larger chunks of data to and from the hijacked device. The data can be anything ranging from harvested information to a software update of the bot itself.

The problem with bulk data transfer on mobile phones is that the mobile connection can be slow, and, therefore, the transfer will take time and thus becomes detectable by the user. Especially if the user is also trying to use the network connection at the same time. If done regularly the costs might show up on the phone bill.

Internet peer-to-peer based botnets will more or less constantly communicate using IP packets, therefore, in order to decrease the possibility for detection the IP-based communication should be kept as hidden as possible.

We developed some strategies for IP-based communication for mobile botnets in order to keep the bots as hidden as possible. The main idea is to communicate mainly using the mobile phone network since this is somewhat harder to monitor. Bulk data should only be transfered using WiFi, if possible.

First, a bot should only initiate a bulk data transfer when connected through WiFi or a high speed 3G network. The 3G network should only be used if the bot some how determined that the phone it is running on has not used any WiFi network for some amount of time. This might be necessary since some mobile phone users will not pay for mobile-data usage, and, therefore, will not bother to use WiFi at all.

Second, all background communication (such as P2P chatter) should be carried out over the 3G network. The P2P chatter does not produce any significant amount of traffic. Also file transfer is not happening at all. This is in order to avoid detection of P2P traffic on the WiFi and DSL link. Also we anticipate that mobile phone network operators do not really monitor the traffic on their network. At least not traffic that does not require a lot of bandwidth, such as the P2P chatter.

Third, all bulk data transfer should be carried out using HTTP. This is to avoid blocked ports and any kind network monitoring.

5.2 SMS-based communication

Each SMS sent might produce costs on the sender and on the receiver side, depending on the contract. Therefore, we created a set of rules in order to reduce the number of SMS messages to send. Also not only the number of messages sent need to be considered but also the destination of the message. Destinations such as foreign countries are likely to be more expensive, but also phone numbers that are routed on another operators network could introduce more costs.

In order to design a useful strategy for sending SMS messages one has to analyze the most common mobile phone contracts that are attached to the targeted mobile platform. For example a big German mobile operator offers four different contracts for the iPhone. Only the contract with lowest monthly rate charges for individual SMS messages. The other contracts include free SMS messages sent within the same network. The contract with the highest monthly charges include 3000 SMS messages sent to any destination.

Taking these facts into account we came up with a simple rule for SMS communication. *Grouping of bots by country and by operator. Limit sending SMS messages between these groups to a minimum.* Messages sent within an operator can be considered free.

Determination of country is easy because of the country code. Determination of the operator is a bit more complex. In certain countrys this can be done by looking at the mobile number area code. If it is not possible to determine the operator from the area code each bot can still query the SIM card for the operator name.

This information needs to be communicated back to the botmaster. This could be done during infection time since here the infector has to deliver the phone number of the new bot to the botmaster anyways.

5.3 Data delivery

Mobile phone resident bots have access to interesting data. In order to transport data from the device back to the botmaster the bot encrypts the data to avoid detection during either transport or through a raid on the drop-site. The encryption is done using public key cryptography to prevent data decryption by extracting the key from a bot infected phone. Therefore, each bot carries the botmasters public key and uses it to encrypt a random symmetric key that is used for data encryption.

6 Bot Implementation

We created a proof-of-concept implementation of our bot design. The implementation includes both the SMS and the P2P-based Command and Control schemas.

We begin with a general description of how command-packets are structured in our botnet. The packet-format is designed in a way that fits both the P2P and SMS-based approaches. We chose to do this since our overall goal was to build a super-hybrid bot that features both C&C schemas. This way the botnet becomes more flexible and very hard to disrupt. The actual implementations are still separated but with spending a little more time on the implementation the super-hybrid bot can be easily build.

The two bots basically are composed of a single executable, a ECC public key to be used for command authentication using ECDSA, and a RSA public key for

Content	Bytes
Signature length	1
ECDSA Signature	variable
Sequence Number	4
Command Type	1
Command	variable

Figure 4. Basic command structure.

encrypting data sent from the bots to the botmaster. The P2P version additionally carries a initial peer-list for the Overnet P2P network. The SMS-client carries an additional dynamic link library for library injection.

6.1 Commands

Commands are composed out of four elements, shown in Figure 4. The command type, the command itself, a sequence number, and a signature. The command type simply specifies what kind of command the packet contains, this can be a shell sequence such as `ping -c 3 www.wired.com`. We will discuss some of the important command types later. The sequence number is a 32-bit counter to prevent replay attacks of commands. The bot will only execute commands with a sequence number higher then the one he has stored. The command packeted is signed and the signature is stored in the packet. The signature guarantees that only the botmaster can send commands to the botnet. In order to keep the command packets as small as possible ECDSA [9] is used for signing. ECDSA signatures are between 70 and 72 bytes, and, therefore, fit in to SMS messages while still leaving space for the actual command. In order to be able to use ECDSA on the iPhone we had to build our own versions of libssl and libcrypto, these were statically linked to our bot executable.

6.2 Kademlia P2P Client

We based our peer-to-peer bot on the KadC[4] Kademlia implementation. We chose KadC because it has no dependencies other than a minimal POSIX API which makes it highly portable. In theory one should be able to compile it for all current smart phone operating systems (including Android, Windows Mobile and Symbian). Further KadC only implements the DHT and not the file transfer part of the P2P network, thus making it the perfect candidate for our purpose. Below we discuss how we use Kademlia and Overnet as our C&C channel.

Once our bot joins the network it starts searching for a specific hash every 15 minutes. In order to send a command to the botnet the botmaster publish a entry in the DHT using the hash the clients search for. The command is transported using the meta information that can be published together with the hash. If the hash is found the client extracts the command from the meta information. The command-data is stored inside the meta information returned with the search result. Since Kademlia only seems to support ASCII data in meta information the command-data is based64 encoded. Although we had to change the actual alphabet used for encoding since Kademlia does not support uppercase characters.

6.3 SMS Client

We implemented the SMS bot-client to piggy back on the iPhone's telephony stack (`com.apple.CommCenter`). This was done using library pre-loading to sit between the iPhone's modem and the telephony stack in order to intercept SMS messages before the SMS application sees them. We use the technique that is described in [4]. The technique also works on other smart phone platforms such as Android and Windows Mobile, therefore we think this is a reasonable approach. The pre-loaded library monitors `open(2)` calls and replaces the file descriptors for the modem lines with file descriptors connected to the actual bot application. Thus all AT commands and results to and from the modem first pass through the bot. If the bot recognizes an incoming SMS message it tries to parse it, if the parsing is successful the SMS is so to say swallowed and not passed on to the telephony stack. All other SMS messages are passed on. Therefore, the control SMS messages never touch the SMS application or SMS database and stays hidden.

Sending SMS messages is done by issuing AT commands to the modem device (`/dev/tty.debug`). This again is not seen by the user in anyway since the SMS message is not handled by the telephony stack.

6.4 Evaluation

We evaluated our bot design and implementation by installing the bot on a number of iPhones in our lab. The bot did not have any kind of spreading functionality implemented in order to make sure it does not escape our test environment. Also the evaluation was

[4]http://kadc.sourceforge.net/

focused on the Command and Control (C&C) functionality, rather than the infection routines.

The evaluation was conducted by running the bot and sending it commands, either via the P2P network or directly via SMS. We constantly monitored the bot activity in order to determine if the commands were successfully received and executed. We did not only sent correct commands to the bots but also commands with broken signatures and invalid sequence numbers.

For evaluation we implemented a number of commands, these are:

- *Add phone number(s).* This command adds a list phone numbers to the forwarding list of a bot.
- *Set sleep interval.* This command is used to set the sleep time between connecting to the P2P network for searching for commands.
- *Execute shell sequence.* This command is used to execute a shell sequence.
- *Download URL.* This command is used for the SMS-HTTP hybrid to download a command file. Besides the URL the command also includes a 128-bit key which is used to decrypt the downloaded file.

Below we will first discuss the P2P-based approach followed by the SMS and SMS-HTTP hybrid design.

6.4.1 Kademlia P2P

We evaluated two scenarios, one where the devices are connected to the Internet via WiFi and one where the devices are connected using a mobile-data connection. The botnet was controlled by a special version of the bot that can issue commands, this version was running on a laptop connected to the university network.

We ran several tests where we executed shell commands and changed the sleep interval for connecting to the P2P network. A basic test was to `ping` one of our servers on the Internet, here we could easily monitor that all our bots actually executed the command.

All in all we where more then satisfied with the performance of our P2P bot-client, especially since it was rather easy to implement.

6.4.2 SMS and SMS-HTTP hybrid

In order to evaluate and test the SMS-based C&C design we implemented a small tool that crafts a command SMS message for our botnet. The tool takes the phone number, the type of command, and the command parameters as input and generates a ready to send SMS PDU. The PDU can then either be sent via the GSM AT command-set or be stored in a file to be used for the SMS-HTTP hybrid C&C mechanism.

We ran a number of tests in order to verify that our SMS C&C mechanism actually works. We again ran the ping-based test to verify that commands with a correct signature and sequence number are accepted. We further verified the basic functionality of file downloads by submitting a URL download command. Forwarding of command messages also worked as expected.

7 Conclusions

Through our work we confirmed that the threat of mobile botnets as pointed out by Traynor at al [1] is real and concrete. We determined that it is easily possible to create a fully functional mobile phone botnet on the most popular smart phone — Apple's iPhone. A mobile phone botnet has many similarities with a desktop computer based botnet but also has certain properties that need to be considered in order to keep the botnet running and hidden. We investigated those cellular specific challenges and properties. We determined that the hybrid approach of SMS and HTTP is the most promising and most dangerous botnet Command and Control structure. Our successful mobile bot implementation stresses that this mobile specific hybrid approach would require very specific and difficult counter measures from a telco. The reason is that two totally different cellular subsystems, i.e., SMS and IP, would be needed to monitor and synchronized for specific, but yet unknown events and messages. This would cause cumbersome burdens for a telco to detect and prevent such mobile botnets. Given our preliminary but devastating results from our research journey we feel that there is an urgent need for novel and appropriate cellular phone and network protection mechanisms.

References

[1] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, T. La Porta and P. McDaniel, "On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core," in *ACM Conference on Computer and Communications Security (CCS)*, November 2009.

[2] 3rd Generation Partnership Project. (2004, September) 3GPP TS 23.040 - Technical realization of the Short Message Service (SMS). http://www.3gpp.org/ftp/Specs/html-info/23040.htm.

[3] P.A. Porras, H. Saidi, V. Yegneswaran, “An Analysis of the iKee.B iPhone Botnet,” in *Proceedings of the 2nd International ICST Conference on Security and Privacy on Mobile Information and Communications Systems (Mobisec)*, May 2010.

[4] C. Miller, C. Mulliner. (2009, August) Fuzzing the Phone in your Phone. http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-SLIDES.pdf.

[5] 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)*, July 2010.

[6] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. H. Kang, and D. Dagon, “Peer-to-peer botnets: Overview and case study,” in *Proceedings of the Workshop on Hot Topics in Understanding Botnets*, April 2007.

[7] R. Hund, M. Hamann, T. Holz, “Towards Next-Generation Botnets,” in *4th European Conference on Computer Network Defense (EC2ND 08)*, 2008.

[8] P. Maymounkov and D. Mazi‘eres, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric,” in *Proceedings of the 1st International Workshop on Peer-to Peer Systems (IPTPS02)*, 2002.

[9] D. Johnson, A. Menezes, and S. A. Vanstone, “The elliptic curve digital signature algorithm (ecdsa).” *Int. J. Inf. Sec.*, vol. 1, no. 1, pp. 36–63, 2001. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijisec/ijisec1.html#JohnsonMV01