Transparent Botnet Command and Control for Smartphones over SMS

Shmoocon 2011

Georgia Weidman

Abstract

I present a system for command and control for smartphone based botnets using SMS. The bot application sits below the smartphone application layer in the base operating system and is thus transparent to smartphone users. Additionally I present a botnet structure for smartphones that maximizes robustness and security.

1) Smartphone Botnets

As smartphones become increasingly ubiquitous and powerful, they become appealing targets for botnet infections. Many of the top selling smartphone platforms are built on common PC operating systems. This makes the transition from developing PC based malware to smartphone based malware nearly trivial. Smartphone malware and specifically botnets have been seen both in security research and in the wild. I believe that in the coming years smartphone based malware incidents will be as common as PC based malware is today.

2) SMS Command and Control

Botnet command and control should be implemented in such a way as to prevent detection of the infection by the smartphone users. The traditional IP based methods of botnet command and control are applicable to smartphone platforms. However, IP based communications have the side effect of draining the smartphone battery. Significant drain of the battery runs a significant risk of detection. In a traditional peer to peer IP based botnet, an infected bot regularly initiates a connection to check for new instructions from its master. Since most cellphone providers place smartphones on their networks behind NAT devices, smartphones do not have public IP addresses. Without a public IP address a command and control server cannot initiate a connection with a bot when instructions are available. Thus any IP based command and control system for smartphones would require this sort of polling system where the smartphones regularly check in for instructions, thus causing significant battery drain.

In the GSM world a phone number functions much as a public address in the IP world. Any GSM based system may use this number to directly contact another. The GSM modem can be viewed as a public IP address without filtering or firewall capabilities. Valid packets from any phone number that are received by the modem will be delivered. Additionally, GSM communications are more difficult for security researchers to monitor than IP, and it would require assistance from cell service providers to detect and dismantle GSM based botnets. The natural conclusion to create a less detectable smartphone botnet is to move as much functionality as possible from IP to GSM.

GSM command and control has been used previously in smartphone malware research. For example in [6] a phone call from a master number triggered a command shell from a infected phone. For command and control I chose instead the GSM function of SMS messages. If a smartphone is not available on the GSM network due to being powered off or out of service range, when an SMS message arrives for delivery, the message is queued and delivered when the smartphone is next available. This adds fault tolerance to a SMS based command and control that would not be available via phone calls.

3) Bot Implementation

The telephony stack of a smartphone handles communications between the GSM modem and the application processor. Naming conventions and implementation differ from platform to platform but usually a multiplexing serial line modem driver that translates instructions between application API calls and GSM modem AT commands. The structure is shown in Figure 1.

Application Layer
Modem Driver Serial Line
GSM Modem

Figure 1

In [2], the authors created a SMS fuzzing application for multiple smartphone platforms that acted as a proxy between the modem driver and smartphone application layer. This allowed the authors to inject SMS messages into smartphones without alerting the carriers and incurring SMS based charges. Rather than being transparent to cell carriers, my goal was to have botnet communications be transparent to smartphone users. Having encountered difficulties intercepting SMS communications consistently before they are presented to the user from the application layer, I used a similar technique to intercept SMS communications before they reach the application layer. As the bot application is in the base operating system below the application layer telephony stack as shown in Figure 2, all GSM communications will be intercepted before being presented to the user. This allows botnet related SMS messages to be intercepted and never delivered to the application layer.

Application Layer
BOT
Modem Driver Serial Line
GSM Modem

Figure 2

To avoid detection, on top of hiding botnet related SMS messages from the user, non botnet related SMS messages and other GSM messages need to be passed up to the user in a timely fashion. To make service delay negligible the first functionality performed by the bot application is to check for the SMS-Deliver GSM code (+CMT). All non SMS GSM packets will then be immediately passed to the application layer. Next the bot application will check the SMS-Deliver packet for botnet keys. If valid keys authenticating the SMS message as valid botnet instructions are not found, the SMS message is passed up to the application layer and delivered to the user normally. Thus normal smartphone functionality is in no way interrupted by the presence of the bot application.

Botnet instructions are included in SMS-Deliver PDU packet in the user data field as shown in Figure 3. The user data field includes the message presented to the user when an SMS message is received encoded in 7bit GSM. The example in Figure 3 has user data E 8 3 2 9 B F D 4 6 9 7 D 9 E C 3 7. This translates to the message hellohello when it is decoded. A botnet instruction SMS would instead include instructions understood by the bot application.

Example SMS-Deliver PDU:
07914140540510F1040B916117345476F1000001210371400044A0A**E8329BFD4697D9EC37**

Field	**Value**
Length of SMSC	0 7
Type of Address (SMSC)	9 1
Service Center Address (SMSC)	4 1 4 0 5 4 0 5 1 0 F 1
SMS Deliver Info	0 4
Length of Sender Number	0 B
Type of Sender Number	9 1
Sender Number	6 1 1 7 3 4 5 4 7 6 F 1
Protocol Identifier	0 0
Data Coding Scheme	0 0
Time Stamp	0 1 2 1 0 3 7 1 4 0 0 4 4 A
User Data Length	0 A
User Data	**E 8 3 2 9 B F D 4 6 9 7 D 9 E C 3 7**

Figure 3

4) Botnet Structure

The proposed botnet structure is organized into three tiers as seen in Figure 4 to maximize security and robustness of the botnet. The first tier is made up of master bots that are directly handled by bot herders. These bots are not required to be bots at all; infection with the bot application is not necessary. There is no requirement that these phones be smartphones or even cellphones at all. PCs with GSM modems or Google Voice accounts would suffice. Master bots should have phone numbers changed regularly to minimize the risk of detection. The second tier of the botnet is the sentinel bots. The sentinel bots act as proxies between the master bots and the mass of the botnet. These bots should be trusted smartphones that have been in the botnet without detection for a significant time. Early on in the life cycle of a botnet bot herders may want to use phones under their own control to fill this role and later promote slave bots to sentinel bots. The third tier of the botnet is the slave bots. Most infected smartphones in the botnet will be in this tier. These bots receive instructions from the sentinel bots and perform the botnet payloads. They have no direct contact with master bots.

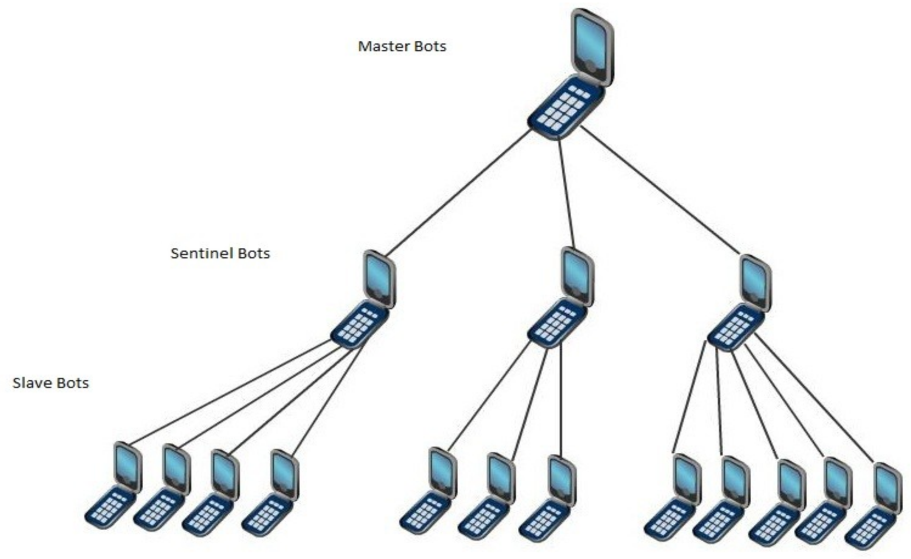

Figure 4

5) Robustness

In the suggested system the master bots are the only bots in the botnet that are aware of all bots. Managing botnet structure and communication is handled by the bot herders. It is also a viable command and control system that each sentinel bot handles the structure and communication of all slave bots under its direct command and thus stores phone numbers of these bots in a known location. This sort of self managed botnet would require less overhead work for bot herders, but has the side effect that in the event of bot detection a group of other bots may also be potentially compromised. While a completely bot herder managed system creates more work for bot herders, this may be desirable to herders as it decreases potential information disclosures. Additionally much of the overhead work of maintaining the botnet structure may be automated by bot herders.

When researching a botnet in the wild the goal of security researchers is to “behead” the botnet by stopping the command and control server. In this case the command and control server changes its phone number regularly to avoid detection. In the event that a phone number of a master bot is discovered, the number will soon be replaced by another, and the discovery process will need to begin again. Sentinel bots will also rotate so a compromise will not result in a prolonged link to the master bots.

Occasionally it is inevitable that a bot infection will be detected by the user. If the infection is detected by a skilled security researcher all communications sent and received by the bot may be intercepted. The three tier system and the structure rotations minimize the inherent risk of botnet take down. A compromised slave bot receives communications from a single sentinel bot. A compromised sentinel receives communications from a single master which again is not active for long. The sentinel sends communications to a set of slave bots. The only clues to botnet structure that may be found from a compromised smartphone are in the communications. No botnet related phone numbers are stored in either sentinel or slave bots.

6) Security Concerns

SMS messages botnet instructions need to be obfuscated in such a way that cell carriers or security researchers potentially reading the messages cannot glean the intent of the messages from the 7bit GSM encoded user data or the decoded plain text. It should be noted however that knowledge of messages combined with reverse engineering of a compromised bot application could lead to security researchers being able to construct valid botnet instruction messages.

Given that sentinel and slave bots do not contain a list of valid botnet phone numbers by which to authenticate botnet related messages, a cryptographic system must be in place to authenticate an SMS as a valid botnet instruction that should be acted upon. The use of symmetric key algorithms has the caveat that since the key is stored in all infected phones, anyone who compromises a single bot may gain access to the key. This opens the possibility that a usurper may piggyback on another bot herders botnet, though due to the secrecy of the structure, a usurper would have additional difficulties figuring out where to send the valid botnet SMS messages created. This problem can be further assuaged by regularly switching the symmetric keys used.

However, an asymmetric cryptographic algorithm is better suited to the goals of this botnet implementation. Thus a compromised bot would only have a public key and thus would not lead a security researcher to be able to masquerade as a valid bot herder. Due to the space limitations in SMS the Elliptic Curve Algorithm discussed in [6] is well suited for this application. Additionally a onetime element would need to be included in the cryptographic scheme of the botnet to prevent replay attacks.

7) Methods of Infection

One method of infection is remote root exploitation. An infected phone exploits a vulnerable phone and infects it with the bot application. This kind of attack has been seen in the wild with the iKee.B “Duh” iPhone botnet discussed in [4]. However, local exploitation is also a viable attack vector for botnet infection. In [5] the author created a seemingly innocuous application for Android smartphones and uploaded it to the Android marketplace. The application took advantage of the Android trust model to infect phones that installed the application with botnet code. Within hours of uploading the application, hundreds of users were infected. Since our bot application requires root privileges, the application would need to exploit a local privilege escalation vulnerability. Additionally, smartphone users who have jail broken their phones and have access to the operating system may be enticed to install a application with bot functionality with root privileges. Both local infection methods rely on lack of user awareness of these threats as phones are infected only as users install infected applications. As seen in [5] this is a viable attack vector for infecting smartphones.

Smartphone platforms have built in security mechanisms to help users make an informed decision when installing applications. For example, on the Android smartphone platform, before a application is installed, a list of functions the application requires access to is presented to the user for approval. However, since the bot application runs at root level an infected application may be installed that requires no access to SMS or other GSM functions. The bot will still be able to access SMS without alerting the user.

Upon infection a smartphone needs to register with the bot herder to be included in the botnet structure. The bot herder needs to know the phone number of the newly infected bot. In many smartphone implementations the application layer does not know its own phone number. This information is only stored in the modem. Thus simple check in over IP is not viable in this case. Newly infected bots need to check in over GSM using a message that includes the phone number. Bot herders will need separate check in points from master bot phones to further complicate the detection process.

8) Example Botnet Functionality

As smartphone capabilities become on par with average desktop PCs, smartphone botnets will be able to perform all the traditional botnet functionality such as participating in a distributed denial of service (DDOS) attack and sending spam. IP connections through the cell network are slower than the WiFi and DSL connections seen on a desktop PC which does have a limiting impact the capabilities of smartphones to participate in DDOS attack. However, many smartphones have the option to join an available WiFi network and carrier based internet connections become faster on a regular basis. Therefore it is not out of the realm of possibility to see DDOS attacks perpetrated by a large group of infected smartphones in the future.

Another common function of traditional botnets is sending spam messages through email. Smartphone botnets may do this as well, however smartphones have a unique attack vector for spam in the GSM functions. As noted before GSM is liken to an internet connection with a public IP address with no filtering or firewall at the phone level. Any valid SMS message that is received will be delivered to the user. Therefore bot herders may leverage smartphone botnets to send SMS based spam. Whereas even free email clients have spam filters that manage to catch a good deal of received spam, GSM has no such filtering and every message will be delivered to the user interface regardless of spam content.

In [1] the authors discuss the possibility of using cellphone botnets to attack the underlying GSM service. This is also a viable functionality for a smartphone botnet. A large scale infection could lead to enough generated bogus GSM traffic to degrade service for legitimate users as discussed further in [1]. This combined with the GPS functionality of smartphones could be used to cause a denial of service of GSM service in a certain geographic location during a crisis situation, serving to escalating the impact.

9) Mitigation

Cellphone carriers should be aware of this potential threat. Carriers should also insure that all security updates are pushed to all smartphones on their networks in a timely manner. Additionally smartphone users should take care what applications they install on their phones as smartphone trust model security functions do not apply in this case. Users should also insure that their smartphones are receiving the newest version of their software in a timely manner just as PCs are updated regularly. To mitigate the threat of root level malware security measures should be implemented in the base operating system of the smartphone such as integrity checks.

10) Conclusions

Smartphone botnets are a realizable threat as shown by this work. The use of GSM communications as opposed to traditional IP methods of botnet command and control is preferable for these botnets. I was able to build a SMS controlled smartphone botnet using techniques and code publicly available under GPL. Cellphones and security researchers should be aware of this threat and develop strategies to combat mass malware infection of smartphones.

References:

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

[2] Miller, C. Mulliner, C. "Fuzzing the Phone in Your Phone." Blackhat USA, July 2009.

[3] Porras P. Saidi, H. Yegneswaran, V. “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] Oberheide, J. “Android Hax,” SummerCon, July 2010.

[5] Papathanasiou, C. Percoco N. “This is not the droid you're looking for...” Defcon 18, August 2010.

[6] Koblitz, N. “Elliptic Curve Cryptosystems.” *Mathematics of Computation, 48(177),* 203-209.

[7] 3rd Generation Partnership Project. 3GPP TS 23.040 - Technical realization of the Short Message Service (SMS). http://www.3gpp.org/ftp/Specs/html-info/23040.htm, September 2004.