Transparent Botnet Control for Smartphones over SMS

Georgia Weidman
Why Smartphone Botnets?

Nearly 62 million smartphones sold in Q2 2010

Development is similar to standard platforms
    Android = Linux
    iPhone = OSX
    Windows Mobile = Windows

Technical specs not as good as top of the line desktops. They are capable and improving rapidly.
Why SMS C&C?

Battery Management: IP runs down battery quickly

Fault Tolerant: If SMS fails it will queue and retry

Difficult for security researchers to monitor
How an SMS is sent and received

- Telephony Stack (Userspace)
- Serial Line/Modem Driver
- Modem
How an SMS is sent and received

- Telephony Stack (Userspace)
- Serial Line/ Modem Driver
- Modem
How an SMS is sent and received

- Telephony Stack (Userspace)
- Serial Line/ Modem Driver
- Modem
How an SMS is sent and received
How an SMS is sent and received
How an SMS is sent and received
How an SMS is sent and received
How an SMS is sent and received
How an SMS is sent and received
How an SMS is sent and received
Previous Work: SMS Fuzzing

At Blackhat 2009, Charlie Miller & Collin Mulliner proxied the application layer and modem to crash smartphones with SMS.

http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-PAPER.pdf
Previous Work: SMS Fuzzing

- Telephony Stack (Userspace)
- Serial Line/ Modem Driver
- Modem
Previous Work: SMS Fuzzing

Telephony Stack (Userspace)

Injector

Serial Line/ Modem Driver

Modem
Previous Work: SMS Fuzzing

- Telephony Stack (Userspace)
- Injector
- Serial Line/Modem Driver
- Modem
My Work: SMS Botnet C&C

- Telephony Stack (Userspace)
- Injector
- Serial Line/Modem Driver
- Modem
My Work: SMS Botnet C&C

- Telephony Stack (Userspace)
- BOT
- Serial Line/ Modem Driver
- Modem
## SMS-Deliver PDU

07914140540510F1040B916117345476F100000121037140044A0AE8329BFD4697D9EC37

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of SMSC</td>
<td>07</td>
</tr>
<tr>
<td>Type of Address (SMSC)</td>
<td>91</td>
</tr>
<tr>
<td>Service Center Address (SMSC)</td>
<td>41 40 54 05 10 F1</td>
</tr>
<tr>
<td>SMS Deliver Info</td>
<td>04</td>
</tr>
<tr>
<td>Length of Sender Number</td>
<td>0B</td>
</tr>
<tr>
<td>Type of Sender Number</td>
<td>91</td>
</tr>
<tr>
<td>Sender Number</td>
<td>51 17 34 45 88 F1</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00</td>
</tr>
<tr>
<td>Data Coding Scheme</td>
<td>00</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>01 21 03 71 40 04 4A</td>
</tr>
<tr>
<td>User Data Length</td>
<td>0A</td>
</tr>
<tr>
<td>User Data</td>
<td>E8 32 9B FD 46 97 D9 EC 37</td>
</tr>
</tbody>
</table>

## SMS-Deliver PDU

07914140540510F1040B916117345476F100000121037140044A0AE8329BFD4697D9EC37

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of SMSC</td>
<td>07</td>
</tr>
<tr>
<td>Type of Address (SMSC)</td>
<td>91</td>
</tr>
<tr>
<td>Service Center Address (SMSC)</td>
<td>41 40 54 05 10 F1</td>
</tr>
<tr>
<td>SMS Deliver Info</td>
<td>04</td>
</tr>
<tr>
<td>Length of Sender Number</td>
<td>0B</td>
</tr>
<tr>
<td>Type of Sender Number</td>
<td>91</td>
</tr>
<tr>
<td>Sender Number</td>
<td>61 17 34 54 76 F1</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00</td>
</tr>
<tr>
<td>Data Coding Scheme</td>
<td>00</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>01 21 03 71 40 04 4A</td>
</tr>
<tr>
<td>User Data Length</td>
<td>0A</td>
</tr>
<tr>
<td>User Data</td>
<td>E8 32 9B FD 46 97 D9 EC 37</td>
</tr>
</tbody>
</table>
How the Botnet Works

1. Bot Receives Message
2. Bot Decodes User Data
3. Bot Checks for Bot Key
4. Bot Performs Payload Functionality
How It Works

1. Bot Receives Message
   Bot receives all communication from modem
   If SMS (code CMT) continue analysis
   If not SMS pass up to user space

2. Bot Decodes User Data

3. Bot Checks for Bot Key

4. Bot Performs Payload Functionality
How It Works

1. Bot Receives Message

2. Bot Decodes User Data
   Moves through PDU to User Data
   Decode 7 bit GSM to plaintext

3. Bot Checks for Bot Key

4. Bot Performs Payload Functionality
How It Works

1. Bot Receives Message

2. Bot Decodes User Data

3. Bot Checks for Bot Key
   Bot checks for secret key in message
   If bot message continue analysis and swallows message (user never sees it)
   If not bot message passed to user space

4. Bot Performs Payload Functionality
How It Works

1. Bot Receives Message
2. Bot Decodes User Data
3. Bot Checks for Bot Key
4. Bot Performs Payload Functionality
   Bot reads functionality request in message
   If found perform functionality
   If not found fail silently
Botnet Structure
Master Bot
Master Bot

Handled by botherders

Switched out regularly to avoid detection
Prepay SIM Cards + Kleptomania

In charge of bot structure

Sends instructions to Sentinel Bots
Sentinel Bots
Sentinel Bots

Several “trustworthy” long infected bots receive instructions from master bot and pass on instructions to a set of slave bots.
Slave Bots

Receive instructions from sentinel bots

No direct contact with master bots

Carry out botnet payload functionality (DDOS, SPAM, etc.)
Robustness

Master Bot:
- May change device, platform, SIM at will
- Prepayed phones are difficult to track
- Has knowledge of all active bots

Sentinel Bots:
- Reserved for long time bots
- The only bots that interact directly with the master
- Master may promote any slave when needed

Slave Bots:
- A compromise results in at most finding the identity of a single sentinel
Security Concerns

Impersonation:
Use cryptographic keys to authenticate master bot and sentinel bots

Replay:
SMS timestamps
Sequence numbers/ one time keys

Elliptic Curve Algorithm
Limitations

Possibility of detection from phone bills

User Data is limited to 160 characters
(instructions and keys must fit in this space)

On some platforms only the modem knows the phone number
Getting The Bot Installed

Regular Users:
App + Local Root Exploit (Sendpage etc.)
Example: John Oberheide's Twilight
Android Botnet Defcon Skytalks 2010

Root-level/Jailbroken Users:
Root level app using proxy function for AWESOME + Bot
Example: flashlight + tether for iPhone

Remote:
Remote root exploit (rooted and nonrooted)
Example: iKee-B “Duh” Worm for iPhone
Example Payloads

Spam
  Creating SMS-Send PDUs and passing them to the modem
  Example: SMS ads

DDOS
  Millions of smartphones vs. a server

Loading New Functionality
  Send URL in payload
  Download the module into known payloads

Degrading GSM service
  Overloading the network with bogus requests
What This Really Means

If attackers can get the bot installed they can remotely control a user's phone without giving any sign of compromise to the user.
Mitigation

Integrity checks of base smartphone operating systems

Liability for smartphone applications including root level

User awareness
Parallel Research:

*Rise of the iBots: Owning a Telco Network*
Collin Mulliner and Jean-Pierre Seifert

SMS/P2P hybrid smartphone botnet research

iPhone based

DEMO : )

Android Bot with SMS Spam Payload

Released code has the bot without payloads (have fun)
Thanks

To Mom for helping me master stuff like this:

```c
char* encodedmessage;
encodedmessage = malloc(13);
encodedmessage = hellogeorgia;
```
Contact

Georgia Weidman
Email: Georgia@grmn00bs.com
Website: http://www.grmn00bs.com
Twitter: vincentkadmon

Slides and Code are on the website
Selected Bibliography

SMS fuzzing: http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhonePAPER.pdf
Cell bots attack GSM core:
http://www.patrickmcdaniel.org/pubs/ccs09b.pdf
Twilight botnet:
SMS/P2P iPhone bots:
False Starts: User Header Data

User Header Data (UHD) is just ahead of User Data in a PDU

Tells the phone how to handle the SMS (ex. Concatenated message)

Previous security research found faws in how these are handled resulting in compromises
False Starts: User Header Data

Not all UHD codes are used

Planned to use unused codes for bot instruction indicators

This worked fine with fuzzers and emulators

SMSCs drop PDUs with unused codes. UHD based bots are not usable in the wild

Some used codes are also dropped