



Malware Analysis Report

Trojan: AndroidOS/DroidDeluxe
September 2011



Kindsight
Security
Labs

Kevin McNamee
Kevin@kindsight.net
www.kindsight.net

Analysis Summary

Name: AndroidOS/DroidDeluxe
MD5: 705738ec081bfac0810ad94045ab4892
Size: 677548 bytes
Source: Contagio
File Type: Android Application
Sample Collected: 2011-09-27 11:38:08
Sample Inserted: 2011-09-27 11:38:08
Application Name: Recovery Deluxe 0.1
Application: com.pocketluxus.recovery

Description

The purpose of the app is to “recover” passwords from various system files on the phone. To accomplish this it roots the phone and changes the file access controls to allow world-read/write access to certain key files. It was not successful in our tests so this may not actually be malware but just a badly written app.

When you press the “Recovery” button it claims to be “collecting data”, but in the test environment this never actually completes. In the background it attempts to get root access to the phone using the “Rage Against the Cage” exploit (CVE-2010-EASY). This exploit uses a vulnerability in the USB debug daemon (adb). If this debug daemon is not active it will prompt the user to activate it, presumably so the password recovery will succeed. If root access is achieved the malware changes the file access controls to allow anyone read/write access to system accounts, e-mail and contact information.

Once read/write access is provided, there appears to be code to extract user/password information from these files. However, in the test environment this did not appear to work and the app failed with an error message. It also uses Google Analytics to report usage and system information.

Infection

The infection occurs when the user runs the app on their phone. People looking for a “password recovery” tool may be duped into downloading and installing this app.

Threat

The malware roots the phone, changes file permissions to allow world-write access to some system files and sends information about the phone to Google Analytics. Although it leaves the phone in a vulnerable state, it does not appear to do any further damage or leverage the write access to the system files, other than the apparently failed attempt to extract user/password information.

Remediation

The threat can be removed by uninstalling the application. You may also want to remove world-read/write access permissions on the following files:

- /data/system/accounts.db
- /data/data/com.android.email/databases/EmailProvider.db
- /data/data/com.android.providers.contacts/databases/contacts2.db
- /data/data/com.android.providers.settings/databases/settings.db
- /data/data/com.android.providers.telephony/databases/mmssms.db

But to do this you will need root access to the phone!

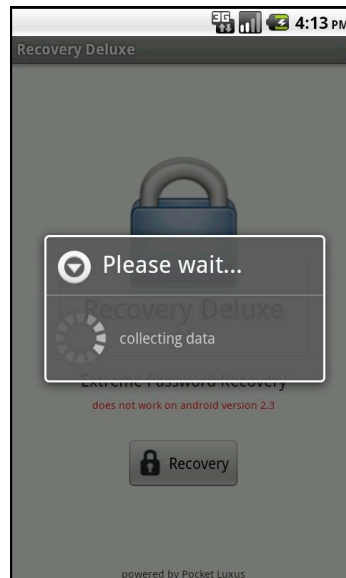
Detailed Analysis

Installation and Infection

The malware appears to be a password recovery tool called Recovery Deluxe.



When started the app admits it will not work on 2.3 and prompt the user to press the “Recovery” button.



When the user pressed the “Recovery” button the following is displayed. In the test environment it loops forever.

The app functions by getting root access to the phone, without telling the user. Then it changes the file permissions on some key system files so that it can get access to userid/password information. In the test environment this didn't work properly. When the app was first run it looped forever “collecting data”. When run for a second time it terminated with an error message.

Getting Root

When the “Recovery” button is pressed the malware attempts to get root access to the phone using the “Rage Against the Cage” exploit (CVE-2010-EASY). This exploit uses a vulnerability in the USB debug daemon (adb). If this debug daemon is not active, the malware will prompt the user to activate it, presumably so the password recovery will succeed. The malware copies the following files from the res/raw directory in the apk file to the local directory on the phone.

```

-rwxrwxrwx app_33 app_33 601454 2011-09-27 16:58 busybox
-rwxrwxrwx app_33 app_33 14931 2011-09-27 16:58 special
-rwxrwxrwx app_33 app_33 5392 2011-09-27 16:58 password
    
```

The “password” file contains the exploit, “special” contains the logic to provide world-write access to system files and “busybox” provides utility commands. The progress of the exploit is recorded in the system log.

```
D/NATIVE ( 234): Running command: chmod 777 /data/data/com.pocketluxus.recovery/password
D/NATIVE ( 234): Running command: chmod 777 /data/data/com.pocketluxus.recovery/special
D/NATIVE ( 234): Running command: chmod 777 /data/data/com.pocketluxus.recovery/busybox
E/NATIVE ( 234): Execute exploite
D/dalvikvm( 234): Trying to load lib /data/data/com.pocketluxus.recovery/lib/libandroidterm.so 0x44ede458
D/dalvikvm( 234): Added shared lib /data/data/com.pocketluxus.recovery/lib/libandroidterm.so 0x44ede458
I/Exec ( 234): JNI_OnLoad
I/NATIVE ( 234): Got processid: 247
I/NATIVE ( 234): Running exploit in order to obtain root access...
I/NATIVE ( 234): chmod 777 /data/data/com.pocketluxus.recovery/password
I/NATIVE ( 234): chmod 777 /data/data/com.pocketluxus.recovery/special
I/NATIVE ( 234): chmod 777 /data/data/com.pocketluxus.recovery/busybox
I/NATIVE ( 234): /data/data/com.pocketluxus.recovery/password
I/NATIVE ( 234): $
I/NATIVE ( 234): $
I/NATIVE ( 234): $
I/NATIVE ( 234): $
I/NATIVE ( 234): [*] CVE-2010-EASY Android local root exploit (C) 2010 by 743C
I/NATIVE ( 234):
I/NATIVE ( 234): [*] checking NPROC limit ...
I/NATIVE ( 234): [+] RLIMIT_NPROC={1024, 1024}
I/NATIVE ( 234): [*] Searching for adb ...
I/NATIVE ( 234): [+] Found adb as PID 39
I/NATIVE ( 234): [*] Spawning children. Dont type anything and wait for reset!
I/NATIVE ( 234): [*]
I/NATIVE ( 234): [*] If you like what we are doing you can send us PayPal money to
I/NATIVE ( 234): [*] 7-4-3-C@web.de so we can compensate time, effort and HW costs.
I/NATIVE ( 234): [*] If you are a company and feel like you profit from our work,
I/NATIVE ( 234): [*] we also accept donations > 1000 USD!
I/NATIVE ( 234): [*]
I/NATIVE ( 234): [*] adb connection will be reset. restart adb server on desktop and re-login.
I/NATIVE ( 234): $
D/dalvikvm( 58): GC_FOR_MALLOC freed 13100 objects / 506208 bytes in 209ms
I/NATIVE ( 234):
I/NATIVE ( 234): [+] Forked 1011 childs.
I/NATIVE ( 234): FORKED FOUND!
W/dalvikvm( 1266): cannot setuid(10033) errno: 11
I/ActivityManager( 58): Start proc com.pocketluxus.recovery:remote for broadcast
com.pocketluxus.recovery/.AlarmReceiver: pid=1266 uid=10033 gids={3003}
D/dalvikvm( 1266): Trying to load lib /data/data/com.pocketluxus.recovery/lib/libandroidterm.so 0x44ede370
D/dalvikvm( 1266): Added shared lib /data/data/com.pocketluxus.recovery/lib/libandroidterm.so 0x44ede370
I/Exec ( 1266): JNI_OnLoad
I/NATIVE ( 1266): Got processid: 1272
I/NATIVE ( 1266): /data/data/com.pocketluxus.recovery/special
I/NATIVE ( 1266): echo finished checked
I/NATIVE ( 1266): /data/data/com.pocketluxus.recovery/busybox killall password
I/NATIVE ( 1266): ENDE
D/NATIVE ( 1274): EUID=0 UID=0
D/NATIVE ( 1274): NEW EUID=0 UID=0
D/NATIVE ( 1274): special done
```

Accessing Information

As can be seen in the log, once root access is obtained the file "special" is run to provide read/write access to the following files:

- /data/system/accounts.db
- /data/data/com.android.email/databases/EmailProvider.db
- /data/data/com.android.providers.contacts/databases/contacts2.db
- /data/data/com.android.providers.settings/databases/settings.db
- /data/data/com.android.providers.telephony/databases/mmssms.db

It is not clear how these files are subsequently used by the malware. There appears to be some code in the DB class that extracts userid and password information from them, but it did not decompile well enough for detailed analysis and it failed in the test environment with the following error.



Google Analytics

The malware uses Google Analytics to collect information about the phone. The following code sends information as Google Analytics events.

```
String str5 = Build.FINGERPRINT;
localAnalytics.trackEvent((String)localObject4, "Fingerprint", str5);
String str6 = Build.DEVICE;
localAnalytics.trackEvent((String)localObject4, "Device", str6);
String str7 = Build.BRAND;
localAnalytics.trackEvent((String)localObject4, "Brand", str7);
String str8 = Build.DISPLAY;
localAnalytics.trackEvent((String)localObject4, "Display", str8);
String str9 = Build.HARDWARE;
localAnalytics.trackEvent((String)localObject4, "Hardware", str9);
String str10 = Build.HOST;
localAnalytics.trackEvent((String)localObject4, "Host", str10);
String str11 = Build.MANUFACTURER;
localAnalytics.trackEvent((String)localObject4, "Manufacturer", str11);
String str12 = Build.MODEL;
localAnalytics.trackEvent((String)localObject4, "Model", str12);
```

While none of this information is particularly sensitive, it does illustrate how services such as this can be leveraged by malware writers for information collection.

The sample analyzed did not successfully communicate with Google Analytics server as illustrated below. Perhaps the account ID used was no longer active.

```
GET /__utm.gif?utmwv=4.3&utmz=1551353841&utmt=event&utme=5(android-buildFRF91*Display*sdk-  
eng 2.2 FRF91 43546 test-keys)(1)&utmcs=UTF-8&utmsr=480x800&utmul=en-  
US&utmcc=__utma%3D999.407276812.1317139831.1317139831.1317139831.1  
HTTP/1.1  
Host: www.google-analytics.com  
User-Agent: GoogleAnalytics/1.0 (Linux; U; Android 2.2; en-us; sdk; Build/FRF91)  
  
HTTP/1.0 400 Bad Request  
Content-Type: text/html; charset=UTF-8  
Content-Length: 11782  
Date: Tue, 27 Sep 2011 16:46:30 GMT  
Server: GFE/2.0
```

Conclusion

This was probably not actually intended as malware. It is just a badly written app that uses dubious techniques to access information and leaves the phone in a compromised state. It attempts to root the phone without the user's knowledge, changes file permissions to allow world-write access to some system files and sends information about the phone to Google Analytics. Once the damage is done the user will require root access to undo the access permission changes that were made.

Appendix

XML Manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="@string/app_vers"
package="com.pocketluxus.recovery"
  xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.GET_ACCOUNTS" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-sdk android:minSdkVersion="4" />
  <supports-screens android:anyDensity="true" android:smallScreens="true"
android:normalScreens="true" android:largeScreens="true" android:resizeable="true" />
  <application android:theme="@android:style/Theme.Light" android:label="@string/app_name"
android:icon="@drawable/icon" android:name=".RApplication" android:debuggable="false">
  <receiver android:name=".AlarmReceiver" android:process=":remote" />
  <activity android:label="@string/app_name" android:name=".RecoveryDeluxe">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
  <activity android:label="@string/app_name" android:name=".Recovery">
    <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
  </activity>
</application>
</manifest>
```