ANTI-PLUGIN:
DON’T LET YOUR APP PLAY AS AN ANDROID PLUGIN
Bio

• Black Hat Veteran.
• Principle Security Researcher @ PANW.

Mobile Security
- Discover Malware
- Android Security

Web Security
- Exploit Kit Detection.
- Browser Security.

Explore & Exploit
- Fuzzing & CVEs.
- Attacks.
Agenda

• Plugin Technology Background
• Demystify Plugin Technology
• Abuse by Malwares
• Solution
Background of Plugin Technology
Why Plugin Technology is Popular?

Want to log in Multiple Accounts Simultaneously?
What is Android Plugin Technology?

• Launch an APK file within an Android app.
• In the unrooted device.
• “Host App” = Android app
• “Plugin” = APK file.
• No need to install the plugin.
What is Android Plugin Technology?

Host App

Plugin

Launch APK w/o installation
vs Dynamic Code Loading (DCL)

• Load + Execute code at runtime.
• Not part of its initial static code base
• Use API like Java Class loader, Runtime.exec.
• Plugin technology is more advanced.
Parallel Space

The Most popular Plugin App

ADDITIOINAL INFORMATION

Updated
January 17, 2017

Installs
10,000,000 - 50,000,000

Current Version
3.1.6672

Requires Android
4.0.3 and up

Content Rating
Everyone

Learn more

Permissions
View details

Offered By
LBE Tech
DroidPlugin

- The most popular SDK implemented Plugin technology.
- Open-Sourced.
- developed by Qihoo 360. 🌍
Demystify Plugin Technology
Droidplugin Overview

Host app

proxy

Plugin app 1

Plugin app 2

Plugin app 3

Proxy Hook

Android Framework
How to create a virtual environment?

• Hooking.
• How to hook API?
  • Java Dynamic Proxy API.
  • Java Reflection.
• What API to hook?
What API to hook?

- Load and launch plugin (APK) without installation.
- Manage the lifecycle of app components (activity, service, content-provide, broadcast-receiver).
- Inter-plugin communication.
- Plugin management (download, update)
• Launch APK file without installation.

loadClass("plugin.class")

findClass("plugin.class")

BaseDexClassLoader

DexPathList

Element []

Dex File

Dex File

Load

"/data/app/" Host App

Plugin

APK
Manage the lifecycle of App components

• App Components
  • Activity
  • Service
  • Broadcast Receiver
  • Content Provider

• System maintain the lifecycle
Start New Activity in Android

Start Activity

Current Activity

Notify Activity to Pause Status

ActivityPaused

Activity Manager Service (AMS)

Create New Activity

Create New Activity

Launch ActivityThread

attachApplication

bindApplication

Activity Thread

Handle Activity

onCreate

Plugin Activity
Current Activity

ActivityManager Service (AMS)

Activity Thread

Plugin Activity

startActivity (intent)

Intent

Intent to AMS

Intent to AMS

PluginActivity.class

Intent

Handle LaunchActivity (intent)

onCreate(...)

API Calls
`startActivity(intent)`

Current Activity

ActivityManager Service (AMS)

Pre-defined Stub Components

<activity android:name="$$StubActivity01"/>

...
```
startActivity

ActivityManagerNative.getDefault() {
  gDefault.get();
}
Singleton<IActivityManager> gDefault;
// ServiceManager.getService("activity");
// IActivityManager am = asInterface();

ActivityThread.mH.mCallback

handleLaunchActivity

onCreate(...)
```
Hook to Start New Service.
Abusing Plugin Technology by Malware
Abusing of DroidPlugin

Android App Powered by DroidPlugin

Benign: 5268
Malicious: 114630

Graph showing the trend of Malicious DroidPlugin apps with numbers from 2015/07 to 2017/02.
Benefit of Abusing DroidPlugin

- Update/Install New Malware Without Rooting the Phone
- Evade Static Detection
- Phish on Authenticated App Without Repackaging
PluginPhantom: New Android Trojan Abuses “DroidPlugin” Framework

By Cong Zheng and Tongbo Luo
November 30, 2016 at 1:00 PM
Category: Unit 42 Tags: Android, DroidPlugin, Google, PluginPhantom, threat research

PluginPhantom trojan exploits Android plugins to snoop

"PluginPhantom" Android Trojan Uses Plugins to Evade Detection

A new class of Trojan as it is the first to abuse Android Plugin technology
Modularized Malware Functionality
Malware
DualTwitter

APK

Malicious Host App
Our Solution: Plugin-Killer
Potential Solutions

• Block Plugin Technology.
• Support plugin by Android system.
• Improving Detection Technique.
• Opt-out options for APK file => PluginKiller.
Plugin Killer

• Protect legitimate app from running in malicious host app.
• App fails to be aware of being launched as a plugin.
• Our Solution: PluginKiller.
  • Lightweight Library.
  • Compatible to all Android versions.
• Mechanisms to detect the virtual environment.
Use PluginKiller

```java
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        if (isLoadedAsPlugin()) {
            TerminatesApp();
        }
        ...
    }
}
```

Similar to FrameBuster JavaScript code used in browser.
Detect Virtual Environment

• Mismatch in the Manifest Info
  • Service/Activity Name.
  • Permissions.

• Detect from Runtime Info
  • Process with same UID.
  • Working Directory.
  • Process Name.

• Runtime Change component Features.
  • Enable a broadcast Receiver declared as Disabled in manifest.

• Broadcast Receiver
  • unregister all dynamic receivers and try to trigger static receivers.
Mismatch in the Manifest Info

Plugin’s Manifest File

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.panw.lab.blackhatdemo">
  <service android:name="com.panw.lab.BlackHatDemo" />
</manifest>
```

DroidPlugin

com.morgoo.droidplugin.stub.ServiceStub $StubP08$P00

Undeclared
But granted 125
Permissions
### Detect from Runtime Info

<table>
<thead>
<tr>
<th>Process Name With Same UID</th>
<th>DataDir: Directory assigned to the package for its persistent data</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.panw.lab.blackhatdemo</td>
<td>/data/data/com.panw.lab.blackhatdemo</td>
</tr>
<tr>
<td>com.droidplugin.demo:Plugin P02</td>
<td>/data/data/com.droidplugin.demo/Plugin/com.panw.lab.blackhatdemo</td>
</tr>
<tr>
<td>DroidPlugin com.droidplugin.demo</td>
<td>/data/data/com.droidplugin.demo/data/com.panw.lab.blackhatdemo</td>
</tr>
</tbody>
</table>
Detect from App Component Behavior

• Number of launched Activity and Service.
  • DroidPlugin defined 10 stub activities and 10 stub services.
  • Launch more than 10 services.

• Static Broadcast Receiver.
  • DroidPlugin converts Static Receiver to Dynamic.
  • Define a Static Receiver, Unregister all Dynamic.
  • In DroidPlugin, no receiver is alive.
Enable Broadcast Receiver with static intent-filter.

```xml
<receiver android:name="AntiReceiver"
    android:enabled="false">
    <intent-filter>
        <action android:name="ANTI_STATIC" />
    </intent-filter>
</receiver>
```

Whether or not the broadcast receiver can be instantiated by the system

```java
ctx.getPackageManager().setComponentEnabledSetting(
    ComponentName, COMPONENT_ENABLED_STATE_ENABLED, ...)
```
Test Environments

Parallel Space by LBE Tech
Go-Multiple By GO Dev Team X
Parallel Accounts By ImaTech Innovations
Parallel Box By ParallelBoxTeam
Gemini Multi Accounts

DroidPlugin
VirtualApp
## Anti Plugin SDK Evaluation

<table>
<thead>
<tr>
<th>Service Name Check</th>
<th>Droid Plugin</th>
<th>Go Multiple</th>
<th>Multiple Accounts</th>
<th>Parallel Space</th>
<th>Parallel Accounts</th>
<th>Parallel Box</th>
<th>Gemini</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DETECTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DETECTED</td>
</tr>
<tr>
<td>Undeclared Permission</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td></td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
</tr>
<tr>
<td>SharedUID Process Check</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
</tr>
<tr>
<td>AppRuntimeDir Check</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td></td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
<td>DETECTED</td>
</tr>
<tr>
<td>ReceiverFilter Check</td>
<td>DETECTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DETECTED</td>
</tr>
<tr>
<td>EnabledComp Check</td>
<td>DETECTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DETECTED</td>
</tr>
</tbody>
</table>
Three Takeaways

• Android Plugin Technology.
• Abusing of Plugin Technology by malware.
• Lightweight Solution to protect your app.
Q & A

• Looking for collaboration on New detection mechanism.