[Android WebView的Js对象注入漏洞解决方案](http://blog.csdn.net/leehong2005/article/details/11808557)

## 1，使用场景

我们很多时候要使用WebView来展示一个网页，现在很多应用为了做到服务端可控，很多结果页都是网页的，而不是本地实现，这样做有很多好处，比如界面的改变不需要重新发布新版本，直接在Server端修改就行了。用网页来展示界面，通常情况下都或多或少都与Java代码有交互，比如点击网页上面的一个按钮，我们需要知道这个按钮点击事件，或者我们要调用某个方法，让页面执行某种动作，为了实现这些交互，我们通常都是使用JS来实现，而WebView已经提供了这样的方法，具体用法如下：

1. mWebView.getSettings().setJavaScriptEnabled(**true**);
2. mWebView.addJavascriptInterface(**new** JSInterface(), "jsInterface");

我们向WebView注册一个名叫“jsInterface”的对象，然后在JS中可以访问到jsInterface这个对象，就可以调用这个对象的一些方法，最终可以调用到Java代码中，从而实现了JS与Java代码的交互。

我们一起来看看关于addJavascriptInterface方法在Android官网的描述：

* This method can be used to allow JavaScript to control the host application. This is a powerful feature, but also presents a security risk for applications targeted to API level [JELLY\_BEAN](http://developer.android.com/reference/android/os/Build.VERSION_CODES.html" \l "JELLY_BEAN" \t "http://blog.csdn.net/leehong2005/article/details/_blank) or below, because JavaScript could use reflection to access an injected object's public fields. Use of this method in a WebView containing untrusted content could allow an attacker to manipulate the host application in unintended ways, executing Java code with the permissions of the host application. Use extreme care when using this method in a WebView which could contain untrusted content.
* JavaScript interacts with Java object on a private, background thread of this WebView. Care is therefore required to maintain thread safety.
* The Java object's fields are not accessible.

简单地说，就是用addJavascriptInterface可能导致不安全，因为JS可能包含恶意代码。今天我们要说的这个漏洞就是这个，当JS包含恶意代码时，它可以干任何事情。

## 2，漏洞描述

通过JavaScript，可以访问当前设备的SD卡上面的任何东西，甚至是联系人信息，短信等。这很恶心吧，嘎嘎。好，我们一起来看看是怎么出现这样的错误的。

1，WebView添加了JavaScript对象，并且当前应用具有读写SDCard的权限，也就是：android.permission.WRITE\_EXTERNAL\_STORAGE

2，JS中可以遍历window对象，找到存在“getClass”方法的对象的对象，然后再通过反射的机制，得到Runtime对象，然后调用静态方法来执行一些命令，比如访问文件的命令.

3，再从执行命令后返回的输入流中得到字符串，就可以得到文件名的信息了。然后想干什么就干什么，好危险。核心JS代码如下：

1. **function** execute(cmdArgs)
2. {
3. **for** (**var** obj **in** window) {
4. **if** ("getClass" **in** window[obj]) {
5. alert(obj);
6. **return**  window[obj].getClass().forName("java.lang.Runtime")
7. .getMethod("getRuntime",**null**).invoke(**null**,**null**).exec(cmdArgs);
8. }
9. }
10. }

## 3，漏洞证明

**举例一：**为了证明这个漏洞，写了一个demo来说明。我就只是加载一个包含恶意JS代码的本地网页，HTML其代码如下：

1. **<html>**
2. **<head>**
3. **<meta** http-equiv="Content-Type" content="text/html; charset=UTF-8"**>**
4. **<script>**
5. var i=0;
6. function getContents(inputStream)
7. {
8. var contents = ""+i;
9. var b = inputStream.read();
10. var i = 1;
11. while(b != -1) {
12. var bString = String.fromCharCode(b);
13. contents += bString;
14. contents += "\n"
15. b = inputStream.read();
16. }
17. i=i+1;
18. return contents;
19. }
21. function execute(cmdArgs)
22. {
23. for (var obj in window) {
24. console.log(obj);
25. if ("getClass" in window[obj]) {
26. alert(obj);
27. return window[obj].getClass().forName("java.lang.Runtime").
28. getMethod("getRuntime",null).invoke(null,null).exec(cmdArgs);
29. }
30. }
31. }
33. var p = execute(["ls","/mnt/sdcard/"]);
34. document.write(getContents(p.getInputStream()));
35. **</script>**
37. **<script** language="javascript"**>**
38. function onButtonClick()
39. {
40. // Call the method of injected object from Android source.
41. var text = jsInterface.onButtonClick("从JS中传递过来的文本！！！");
42. alert(text);
43. }
45. function onImageClick()
46. {
47. //Call the method of injected object from Android source.
48. var src = document.getElementById("image").src;
49. var width = document.getElementById("image").width;
50. var height = document.getElementById("image").height;
52. // Call the method of injected object from Android source.
53. jsInterface.onImageClick(src, width, height);
54. }
55. **</script>**
56. **</head>**
58. **<body>**
59. **<p>**点击图片把URL传到Java代码**</p>**
60. **<img** class="curved\_box" id="image"
61. onclick="onImageClick()"
62. width="328"
63. height="185"
64. src="http://t1.baidu.com/it/u=824022904,2596326488&fm=21&gp=0.jpg"
65. onerror="this.src='background\_chl.jpg'"**/>**
66. **</p>**
67. **<button** type="button" onclick="onButtonClick()"**>**与Java代码交互**</button>**
68. **</body>**
69. **</html>**

这段HTML的运行效果如下：



图一：期望运行结果图

上图中，点击按钮后，JS中传递 一段文本到Java代码，显示一下个toast，点击图片后，把图片的URL，width，height传到Java层，也用toast显示出来。

要实现这样的功能，就需要注Java对象。

**简单说明一下**

1，请看**execute()**这个方法，它遍历所有window的对象，然后找到包含getClass方法的对象，利用这个对象的类，找到java.lang.Runtime对象，然后调用“getRuntime”静态方法方法得到Runtime的实例，再调用exec()方法来执行某段命令。

2，getContents()方法，从流中读取内容，显示在界面上。

3，关键的代码就是以下两句

1. **return** window[obj].getClass().forName("java.lang.Runtime").
2. getMethod("getRuntime",**null**).invoke(**null**,**null**).exec(cmdArgs);

Java代码实现如下：

1. mWebView = (WebView) findViewById(R.id.webview);
2. mWebView.getSettings().setJavaScriptEnabled(**true**);
3. mWebView.addJavascriptInterface(**new** JSInterface(), "jsInterface");
4. mWebView.loadUrl("file:///android\_asset/html/test.html");

需要添加的权限：

1. **<uses-permission** android:name="android.permission.INTERNET"**/>**
2. **<uses-permission** android:name="android.permission.ACCESS\_NETWORK\_STATE" **/>**
3. **<uses-permission** android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" **/>**

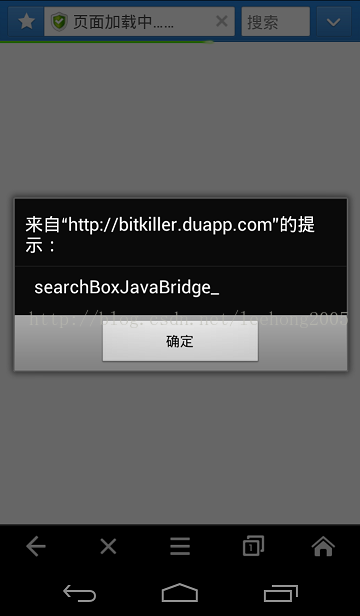
当点击LOAD菜单后，运行截图如下：（理论上应该出现图一界面）



图二：实际运行结果，列出了SDCard中的文件

**举例二：**360浏览器也存在这个问题，我测试的系统是android 4.0.2，360浏览器版本是：4.8.7

在浏览器输入框中输入：http://bitkiller.duapp.com/jsobj.html，然后前往，它会出现如下的界面



图三：360浏览器运行结果

说明：其中searchBoxJavaBridge\_不是360注入的对象，而是WebView内部注入的，这是在3.0以后的Android系统上添加的。

在关闭这个对话框之后，它会列出当前SDCard上面的所有文件列表，如下图所示



图四：错误结果

## 4，解决方案

1，Android 4.2以上的系统

在Android 4.2以上的，google作了修正，通过在Java的远程方法上面声明一个@JavascriptInterface，如下面代码：

1. **class** JsObject {
2. @JavascriptInterface
3. **public** String toString() { **return** "injectedObject"; }
4. }
5. webView.addJavascriptInterface(**new** JsObject(), "injectedObject");
6. webView.loadData("", "text/html", **null**);
7. webView.loadUrl("javascript:alert(injectedObject.toString())");

2，Android 4.2以下的系统

这个问题比较难解决，但也不是不能解决。

首先，我们肯定不能再调用addJavascriptInterface方法了。关于这个问题，最核心的就是要知道JS事件这一个动作，JS与Java进行交互我们知道，有以下几种，比prompt, alert等，这样的动作都会对应到WebChromeClient类中相应的方法，对于prompt，它对应的方法是onJsPrompt方法，这个方法的声明如下：

**[java]** [view plain](http://blog.csdn.net/leehong2005/article/details/11808557" \o "view plain) [copy](http://blog.csdn.net/leehong2005/article/details/11808557" \o "copy)

1. **public** **boolean** onJsPrompt(WebView view, String url, String message,
2. String defaultValue, JsPromptResult result)

通过这个方法，JS能把信息（文本）传递到Java，而Java也能把信息（文本）传递到JS中，通知这个思路我们能不能找到解决方案呢？

经过一番尝试与分析，找到一种比较可行的方案，请看下面几个小点：

【1】让JS调用一个Javascript方法，这个方法中是调用prompt方法，通过prompt把JS中的信息传递过来，这些信息应该是我们组合成的一段有意义的文本，可能包含：**特定标识，方法名称，参数**等。在onJsPrompt方法中，我们去解析传递过来的文本，得到方法名，参数等，再通过反射机制，调用指定的方法，从而调用到Java对象的方法。

【2】关于返回值，可以通过prompt返回回去，这样就可以把Java中方法的处理结果返回到Js中。

【3】我们需要动态生成一段声明Javascript方法的JS脚本，通过loadUrl来加载它，从而注册到html页面中，具体的代码如下：

**[javascript]** [view plain](http://blog.csdn.net/leehong2005/article/details/11808557" \o "view plain) [copy](http://blog.csdn.net/leehong2005/article/details/11808557" \o "copy)

1. javascript:(**function** JsAddJavascriptInterface\_(){
2. **if** (**typeof**(window.jsInterface)!='undefined') {
3. console.log('window.jsInterface\_js\_interface\_name is exist!!');}
4. **else** {
5. window.jsInterface = {
6. onButtonClick:**function**(arg0) {
7. **return** prompt('MyApp:'+JSON.stringify({obj:'jsInterface',func:'onButtonClick',args:[arg0]}));
8. },
10. onImageClick:**function**(arg0,arg1,arg2) {
11. prompt('MyApp:'+JSON.stringify({obj:'jsInterface',func:'onImageClick',args:[arg0,arg1,arg2]}));
12. },
13. };
14. }
15. }
16. )()

**说明：**

1，上面代码中的jsInterface就是要注册的对象名，它注册了两个方法，onButtonClick(arg0)和onImageClick(arg0, arg1, arg2)，如果有返回值，就添加上return。

2，prompt中是我们约定的字符串，它包含特定的标识符MyApp:，后面包含了一串JSON字符串，它包含了方法名，参数，对象名等。

3，当JS调用onButtonClick或onImageClick时，就会回调到Java层中的onJsPrompt方法，我们再解析出方法名，参数，对象名，再反射调用方法。  
4，window.jsInterface这表示在window上声明了一个Js对象，声明方法的形式是：**方法名:function(参数1，参数2)**

## 5，一些思考

以下是在实现这个解决方案过程中遇到的一些问题和思考：

**【1】生成Js方法后，加载这段Js的时机是什么？**

刚开始时在当WebView正常加载URL后去加载Js，但发现会存在问题，如果当WebView跳转到下一个页面时，之前加载的Js就可能无效了，所以需要再次加载。这个问题经过尝试，需要在以下几个方法中加载Js，它们是WebChromeClient和WebViewClient的方法：

* onLoadResource
* doUpdateVisitedHistory
* onPageStarted
* onPageFinished
* onReceivedTitle
* onProgressChanged

**目前测试了这几个地方，没什么问题，这里我也不能完全确保没有问题。**

**【2】需要过滤掉Object类的方法**

由于通过反射的形式来得到指定对象的方法，他会把基类的方法也会得到，最顶层的基类就是Object，所以我们为了不把getClass方法注入到Js中，所以我们需要把Object的公有方法过滤掉。这里严格说来，应该有一个需要过滤方法的列表。目前我的实现中，需要过滤的方法有：

        "getClass",  
        "hashCode",  
        "notify",  
        "notifyAll",  
        "equals",  
        "toString",  
        "wait",

**【3】通过手动loadUrl来加载一段js，这种方式难道js中的对象就不在window中吗？也就是说，通过遍历window的对象，不能找到我们通过loadUrl注入的js对象吗？**

关于这个问题，我们的方法是通过Js声明的，通过loadUrl的形式来注入到页面中，其实本质相当于把我们这动态生成的这一段Js直接写在Html页面中，所以，这些Js中的window中虽然包含了我们声明的对象，但是他们并不是Java对象，他们是通过Js语法声明的，所以不存在getClass之类的方法。本质上他们是Js对象。

**【4】在Android 3.0以下，系统自己添加了一个叫searchBoxJavaBridge\_的Js接口，要解决这个安全问题，我们也需要把这个接口删除，调用removeJavascriptInterface方法。这个searchBoxJavaBridge\_好像是跟google的搜索框相关的。**

**【5】在实现过程中，我们需要判断系统版本是否在4.2以下，因为在4.2以上，Android修复了这个安全问题。我们只是需要针对4.2以下的系统作修复。**