Android WebView Vulnerabilities

pentestlab.blog/category/mobile-pentesting/page/3

February 12, 2017

WebViews are used in android applications to load content and HTML pages within the application. Due to this functionality the implementation of WebView it must be secure in order not to introduce the application to great risk. The purpose of this article is to examine the things that will make WebView less secure and therefore the application itself. Both developers and penetration testers can use this article as a reference when conducting a mobile penetration test.

Loading Clear-Text Content

If WebView is allowing to load clear-text content from the Internet then it would be open to various forms of attack such as MiTM.

```
myWebView.loadUrl("http://www.droidsec.org/tests/addjsif/");
```

SSL Error Handling

The code below instructs the WebView client to proceed when an SSL error occur. This means that the application is vulnerable to MiTM attacks as it could allow an attacker to read or modify content that is displayed to the user since any certificate would be accepted by the application.

```
@Override
public void onReceivedSslError(WebView view, SslErrorHandler handler,
SslError error)
{
handler.proceed();
}
```

JavaScript Enabled

Allowing JavaScript content to be executed within the application via WebView might give the opportunity to an attacker to execute arbitrary JavaScript code in order to perform malicious actions. This setting allow WebView to execute JavaScript code.

```
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

The *JavascriptInterface* function allows bridging between JavaScript code and native Java code. This means that JavaScript code can access and inject Java objects and Java code to be called by JavaScript. Depending on the permissions of the application this could allow an attacker to interact with the functionality of the device (read SMS, microphone recording etc.) exposing the security of the application and the device itself into great risk.

```
WebView webView = new WebView(this);
setContentView(webView);
...
class JsObject {
private String sensitiveInformation;

...
public String toString() { return sensitiveInformation; }

}
webView.addJavascriptInterface(new JsObject(), "injectedObject");
webView.loadData("", "text/html", null);
webView.loadUrl("http://www.example.com");
```

Accessing Local Resources

If the WebView is allowing to access content from other applications that exist on the same device then it could be possible for an attacker to create a malicious html file that could be injected inside the target application through the use **file:scheme**. In order for this malicious file to be loaded needs to have world readable permissions.

```
public class MyBrowser extends Activity {
@override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

WebView webView = (WebView) findViewById(R.id.webview);

// turn on javascript
WebSettings settings = webView.getSettings();
settings.setJavaScriptEnabled(true);

String turl = getIntent().getStringExtra("URL");
webView.loadUrl(turl);
}
}
```

Summary

WebView due to it its functionality can expose the application and the device it great security risk. Mobile penetration testers should examine the following methods and their status when conducting and an android assessment to identify associated risks.

- setAllowContent Access
- setAllowFileAccess
- setAllowFileAccessFromFileURLs
- setAllowUniversalAccessFromFileURLs
- setJavaScriptEnabled

Credits

Code samples have been taken from the following websites:

References