

# SECURITY CAFÉ

SECURITY RESEARCH AND SERVICES



## Mobile penetration testing on Android using Drozer

📅 July 8, 2015   👤 Daniel Tomescu

Mobile phones have become an indispensable part of our daily life. We use mobile phones to communicate with our loved ones, for quick access to information through the Internet, to make transactions through mobile banking apps or to relax reading a good book.

In a way, a big part of our private life has moved into the digital environment. Mobile phones seem to be a pocket-sized treasure of secrets and information, hiding our most valuable photos, mails, contacts and even banking information. There's no wonder why we need mobile phones to have bullet-proof security.



Android is the most common operating system for mobile devices and is particularly interesting from the security point of view. It is very permissive, allowing its users to customize about anything, administrative privileges (a.k.a. rooting) can be unlocked on most phones, it has a very fuzzy system for the permissions required by applications and it features different ways for one application to interact with other applications.

In this blog post, we are going to focus on how Android apps can interact with each other and how the security of those interactions can be tested.

## How apps can interact with each other

The main methods for inter-app communications are:

1. One application can send an ***intent*** in order to start an ***activity exported*** by another application;
2. One application can access ***content provided*** by another application, using ***content://*** URIs;
3. One application can ***broadcast an event*** across applications in order to interact with a ***broadcast receiver*** implemented in another application;
4. One application can access a ***service exported*** by another application.

As you can see, there are a many interactions. Testing in a old-fashioned way (by creating an app for every test you have in mind, installing it on your device and running the test-app) is very time consuming, so it's not really a solution. Here is where Drozer comes into play.

## Introducing Drozer

Drozer is an ambitious project developed by MWR InfoSecurity. Although it is a *Swiss army knife* kind of tool, I will focus on how it can be used in order to start with an exported activity.

First of all, you need to install **Drozer**. You can do so by going to the official [page](#) and follow the instructions, or you can download **Appie** – a set of tools created for mobile penetration testing, it contains a portable version of Drozer.

Second of all, you need to install **Drozer Agent** on your mobile device. The mobile device does not need to be rooted. Drozer Agent is an Android application that requires minimum privileges (internet access only) and acts as a bridge between the Drozer install from your computer and your mobile device. You can download it from the same sources as mentioned above, keeping in mind that the *Drozer* version and *Drozer Agent* version should match.

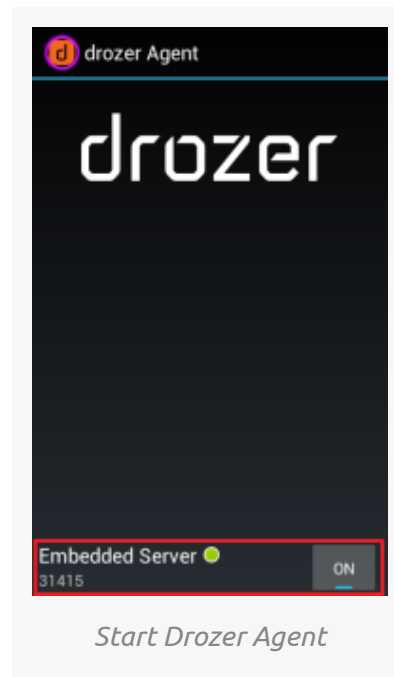
The third step is to install on your mobile device the application that you want to test. For testing and training, you can use one of the following applications:

- *sieve* – a damn-vulnerable application developed by MWR InfoSecurity;
- *FourGoats* – a vulnerable application included in OWASP's **GoatDroid** project;
- *Herd Financial* – another vulnerable application included in OWASP's **GoatDroid** project;
- a real-world application, I'll use for this demo **Firefox for Android** – installed from Google Play.

## Connecting Drozer to the mobile device

In order to connect Drozer to the mobile device, we need to follow the next steps:

- Connect your mobile device to your computer using a USB cable;
- Open Drozer Agent application on your mobile device and click the **ON** button from the bottom-right;



- Use ***adb.exe*** to open a TCP socket between your computer and the server embedded in Drozer Agent: ***adb.exe forward tcp:31415 tcp:31415***
- Go to the folder where you installed Drozer and connect to the mobile device: *drozer.bat console connect*

```

C:\drozer>drozer.bat console connect
Selecting 3cb47f1b16b7939f

      ..          ..:
    ..0..        .r..
    ..a.. . . . . . .nd
      ro..idsnemesisand..pr
      .otectorandroidsneme.
      ..sisandprotectorandroids+.
      ..nemesisandprotectorandroidsn:.
      .emesisandprotectorandroidsnemes..
      ..isandp,...rotectorandro,...idsnem.
      .isisandp..rotectorandroid..snemisis.
      ,andprotectorandroidsnemisisandprotec.
      .torandroidsnemesisandprotectorandroid.
      .snemisisandprotectorandroidsnemesisan:
      .dprotectorandroidsnemesisandprotector.

drozer Console (v2.3.4)
dz>

```

*Drozer list commands*

## Starting an activity from another package

Ok, now we have an interactive Drozer console. What can we do? Let's start an activity, command by command:

- ***list***, will display a list of commands available in Drozer

```

dz> list
app.activity.forintent      Find activities that can handle the given intent
app.activity.info           Gets information about exported activities.
app.activity.start          Start an Activity
app.broadcast.info          Get information about broadcast receivers
app.broadcast.send          Send broadcast using an intent
app.broadcast.sniff         Register a broadcast receiver that can sniff particular
app.package.attacksurface   Get attack surface of package
app.package.backup          Lists packages that use the backup API (returns true on
app.package.debuggable      Find debuggable packages
app.package.info            Get information about installed packages
app.package.launchintent    Get launch intent of package
app.package.list            List Packages
app.package.manifest        Get AndroidManifest.xml of package
app.package.native          Find Native libraries embedded in the application.
app.package.shareduid       Look for packages with shared UIDs
app.provider.columns        List columns in content provider

```

*Drozer list commands*

- **run app.package.list -f firefox** to find a list of packages that contain the string “firefox”; we found *org.mozilla.firefox*.

```

dz> run app.package.list -f firefox
org.mozilla.firefox (Firefox)
dz>

```

*Find package name with Drozer*

- **run app.package.attacksurface org.mozilla.firefox** to identify the attack surface for our application; we found 113 exported activities, 12 exported broadcast receivers, 8 exported content providers and 1 exported service; this is a good example of a big attack surface.

```
dz> run app.package.attacksurface org.mozilla.firefox
Attack Surface:
  113 activities exported
  12 broadcast receivers exported
  8 content providers exported
  1 services exported
  Shared UID (org.mozilla.firefox.sharedID)
dz>
```

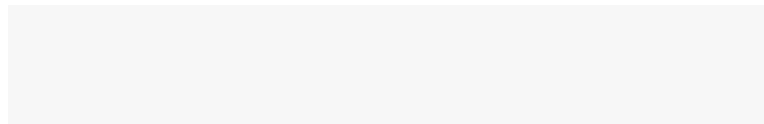
*Find attack surface with Drozer*

- ***run app.package.manifest org.mozilla.firefox*** to extract the AndroidManifest.xml file for our package (to find details about the exported activities).

```
dz> run app.package.manifest org.mozilla.firefox
<manifest sharedUserId="org.mozilla.firefox.sharedID"
  versionCode="2015062534"
  versionName="39.0"
  installLocation="0"
  package="org.mozilla.firefox"
  platformBuildVersionCode="21"
  platformBuildVersionName="5.0-1521886">
  <uses-sdk minSdkVersion="11"
    targetSdkVersion="21">
  </uses-sdk>
  <uses-permission name="android.permission.GET_ACCOUNTS">
  </uses-permission>
  <uses-permission name="android.permission.ACCESS_NETWORK_STATE">
  </uses-permission>
  <uses-permission name="android.permission.MANAGE_ACCOUNTS">
  </uses-permission>
  <uses-permission name="android.permission.USE_CREDENTIALS">
  </uses-permission>
  <uses-permission name="android.permission.AUTHENTICATE_ACCOUNTS">
  </uses-permission>
```

*Extract AndroidManifest.xml with Drozer*

- ***run app.activity.info -a org.mozilla.firefox*** to list the exported activities; we can see that there is an exported activity named *org.mozilla.firefox.App* that does not require any permission to be started.



```

dz> run app.activity.info -a org.mozilla.firefox
Package: org.mozilla.firefox
org.mozilla.gecko.BrowserApp
Permission: null
org.mozilla.firefox.App
Permission: null
Target Activity: org.mozilla.gecko.BrowserApp
org.mozilla.gecko.webapp.Dispatcher
Permission: null
org.mozilla.gecko.Webapp
Permission: null
org.mozilla.firefox.Webapp
Permission: null
Target Activity: org.mozilla.gecko.Webapp
org.mozilla.gecko.webapp.Webapps$Webapp0
Permission: null
org.mozilla.gecko.webapp.Webapps$Webapp1
Permission: null
org.mozilla.gecko.webapp.Webapps$Webapp2
Permission: null
org.mozilla.gecko.webapp.Webapps$Webapp3
Permission: null
org.mozilla.gecko.webapp.Webapps$Webapp4
Permission: null
org.mozilla.gecko.webapp.Webapps$Webapp5
Permission: null

```

*List exported activities with Drozer*

By analyzing the *AndroidManifest.xml* file that we previously retrieved, we can see that *org.mozilla.firefox.App* is defined as an alias for *web.mozilla.gecko.BrowserApp* activity and that it has multiple intent filters (an intent must comply with at least one filter in order to start the activity).

```

<intent-filter>
  <action name="android.intent.action.VIEW">
  </action>
  <category name="android.intent.category.BROWSABLE">
  </category>
  <category name="android.intent.category.DEFAULT">
  </category>
  <data scheme="file">
  </data>
  <data scheme="http">
  </data>
  <data scheme="https">
  </data>
  <data mimeType="text/html">
  </data>
  <data mimeType="text/plain">
  </data>
  <data mimeType="application/xhtml+xml">
  </data>
</intent-filter>

```



- **`run app.activity.start --component org.mozilla.firefox org.mozilla.firefox.App --action android.intent.action.VIEW --category android.intent.category.DEFAULT --data-uri http://securitycafe.ro`** to start an activity.

```
dz> run app.activity.start --component org.mozilla.firefox org.mozilla.firefox.App
                             --action android.intent.action.VIEW
                             --category android.intent.category.DEFAULT
                             --data-uri http://securitycafe.ro
dz>
```

*Starting an exported activity with Drozer*

As a result, Firefox opened a new tab on the mobile device and navigated to a really cool website.



*Opened a new tab in Firefox  
with an intent initiated*

So, what happened? The short story is that Drozer Agent, an application without special permissions installed on our mobile device, can start activities that were exported by other apps. And this is just the tip of the iceberg.

It is important to understand that any other application out there can initiate an Intent like this, not only Drozer. If you want to test by yourself and create a simple Android application that starts an intent, here is the equivalent code for the Drozer syntax used above:

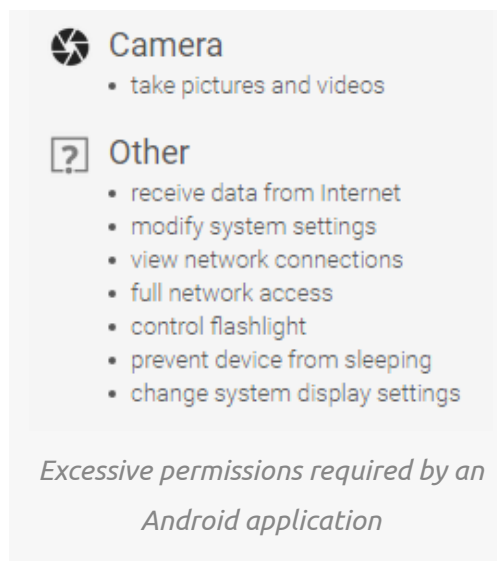
```
1  import android.content.Intent;
2  ...
3
4  public void sendIntent() {
5      Intent startFirefox = new Intent(Intent.ACTION_VIEW);
6      startFirefox.setComponent(
7          new ComponentName("org.mozilla.firefox",
8                          "org.mozilla.firefox.App")
9          );
10     startFirefox.setData(Uri.parse("http://securitycafe.ro"));
11     startService(startFirefox);
12 }
```

*Java code for starting an activity from another package*

## What if the activity requires special permissions to be started?

If the activity is protected from one of the permissions listed [here](#), then we can build our own Drozer Agent with extended permissions. It is not impossible for malware apps to get installed with extended permissions, most users will accept any permission request without reading them.

As a short detour, I consider Android's permission system to be broken. Here is an example (permissions required by one of the most used *Flashlight* apps from Play Store):



Should a lantern be able to take pictures and videos with my camera? No. Should a lantern be able to download data from the Internet? No. Should I allow a lantern to have full network access? No. Should I allow a lantern to modify my system settings? Of course not.

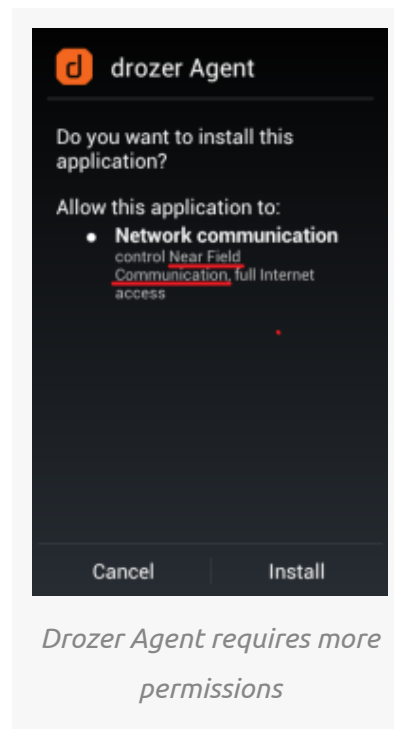
That is just an example, but it illustrates really well how you can't tell if an app is harmless or not, judging only by the permissions it requires. So, if a Flashlight app gets installed on more than 100.000.000 devices with extended permissions, then it's safe to assume that malware will get installed on victim's device with some permissions; it would look suspicious if the malware app wouldn't require any permission at all.

Back to our goal, we can build a Drozer Agent with specific permissions using the Drozer instance from our computer: ***drozer.bat agent build -permission "android.permission.SOMETHING"***

```
C:\drozer>drozer.bat agent build --permission "android.permission.NFC"
Version Mismatch: Consider updating your build(s)
Agent Version: 2.3.3
drozer Version: 2.3.4
Done: c:\users\daniel\appdata\local\temp\tmptt7u9z\agent.apk
C:\drozer>
```

*Building a drozer agent with extended permissions*

If we install the newly built Drozer Agent on our mobile device, we will notice that it requires the added permissions.

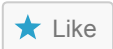


After installing the new Drozer Agent, we can follow the same steps, described in this article, to start an activity that requires specific permissions.

## Conclusions

In the end, I'd like to point out that Android features a complex security model and, sometimes, complexity is the worst enemy of security. Having in mind all the interactions that are possible between apps, we should test in depth the security of the apps that we build. It's better to be safe than sorry!

Share this:



Be the first to like this.

📁 [Mobile security, Pentest techniques](#) 🔖 [android security, drozer, exported activities, mobile penetration testing](#)

◀ PREVIOUS ARTICLE

[Hacknet 2015](#)

NEXT ARTICLE ▶

[NetRipper – Smart traffic sniffing for penetration testers](#)

## 10 THOUGHTS ON “MOBILE PENETRATION TESTING ON ANDROID USING DROZER”

**RAVI**

January 11, 2016 / 10:24 am

Hey Daniel,

Nice and to the point write up. I have one question that , i want to test android application that is installed on my android phone. I have drozer agent in my device and drozer on windows 8.1 (attacking machine ) also and both works fine.

I am facing problem in the “adb.exe forward tcp:31415 tcp:31415” step. From where i can download adb.exe , i tried but it was showing no device found error.

What should I do so that I can connect drozer to drozer agent on port 31415.



↩ Reply

---

**DANIEL TOMESCU**

January 19, 2016 / 10:48 am

Hi Ravi,

You can download ADB from here: <http://adbshell.com/downloads>, but I recommend you to install Android Studio with all USB drivers, it will also contain adb.

Make sure you have your phone connected to your computer via USB. Also, make sure that the DROZER agent is running on your phone and you don't have apps installed that could block the communication (antiviruses, firewalls etc).

On your phone, you should also have USB debugging enabled. Depending on your phone and version, you can find it around here: Settings->Developer Options->USB debugging.

If still nothing works, try switching your USB connection mode from your android device. More details here: <http://www.howtogeek.com/192732/android-usb-connections-explained-mtp-ftp-and-usb-mass-storage/>

Hope it helps,  
Daniel

★ Like

↩ Reply

---

**ANUDEEP**

January 27, 2016 / 1:13 pm

Hi Daniel,

Nice Post!!!

I blocked on the step drozer.bat console connect. Its throwing error as java not found, but java is on my path file and in system too 😞

And also tried updating the drozer\_config file as per the error thrown in the prompt but no use 😞

Could you please help me on how to get in to the drozer prompt ?

★ Like

↩ Reply

---

**ANUDEEP**

January 27, 2016 / 1:37 pm

i am unable to enter in to drozer console after running this command :drozer.bat console connect

Its throwing java error, actually java is found on my machine

Please help me out !!!

★ Like

↩ Reply

---

**ANUDEEP**

January 30, 2016 / 11:14 am

Hi Daniel,

Java is not recognised while running drozer. Could you please help me out on this ?

★ Like

↩ Reply

---

**DANIEL TOMESCU**

February 3, 2016 / 11:56 pm

Hi, what version of Java are you using? Maybe uninstalling Java and installing an older version will do the trick. Also, try manually adding the Java Path in ".drozer\_config" file. More on this here: <https://github.com/mwrlabs/drozer/issues/92>

★ Like

↩ Reply

---

Pingback: [Drozer and ADB | Patrick Lavery](#)

---

**DINA**

August 2, 2016 / 12:17 pm

Thanks for your effort :

I want to make dynamic analysis on android applications to extract features from it and get report.

For example if app send SMS in the background I want to know the number of recipient and the content of the SMS .

I mean I want actions like : action sendnet is the action that sends data over the network,

action phonecalls is the action that makes a phone call,

and action sendsms is the action that sends SMS messages.

Can you help me please???

★ Like

↩ Reply

---

**RAJA**



September 20, 2016 / 12:46 pm

Use a java version 1.6 and manually add ".drozer\_config" under folder system/user

★ Like

↩ Reply

---

Pingback: [Android pentesting resources | nebulam87](#)

---

Leave a Reply

Enter your comment here...

#### OTHER SECURITY BLOGS

---

🔗 [Finnish Blog](#)

🔗 [Netherlands Blog](#)

🔗 [Spanish Blog](#)











## RECENT POSTS

---

-  [Tricking blind Java deserialization for a treat](#)
-  [Robot hacking research](#)
-  [Phishy Basic Authentication prompts](#)
-  [Going further with Responder's Basic Authentication](#)
-  [Exploiting Timed Based RCE](#)

## CATEGORIES

---

-  [Announcements](#) (1)
-  [Conferences](#) (3)
-  [Embedded systems security](#) (1)
-  [Ethical Hacking](#) (6)
-  [General security](#) (7)
-  [IT Security Assurance](#) (1)
-  [IT Security Audit](#) (1)
-  [Metasploit](#) (1)
-  [Misc](#) (10)
-  [Mobile security](#) (2)

- 📁 [Network security](#) (2)
- 📁 [Operating systems](#) (1)
- 📁 [Penetration Testing](#) (5)
- 📁 [Pentest techniques](#) (15)
- 📁 [Web security](#) (7)

## BLOG STATS

---

- 195,518 hits

## FOLLOW US VIA EMAIL

---

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 82 other followers

**FOLLOW**

- [RSS - Posts](#)
- [RSS - Comments](#)