pentest all the things...

Android Pentesting Command Cheatsheet

Useful tools

Android SDK

Android Studio

GenyMotion emulator

BusyBox

apktool

dex2jar

enjarify

jd-gui

MWR's Drozer testing framework

Fino

`android/adt-bundle-linux-ver-number/sdk/platform-tools` : adb tool

`android/adt-bundle-linux-ver-number/sdk/tools` : emulator, android tools

`~/.android/avd/<emulator-device-name>.avd/` : emulator config files

Useful Windows file paths

`C:\Users\username\AppData\Local\Android\sdk`

Useful Android paths

`/data/app` : location of app on android device

`/data/data/[packagename]/*` : app data files

`/data/Dalvik-cache` : Classes.dex for all installed apps

Useful configuration settings

Enable hardware keyboard in emulator:

Add: `hw.keyboard=yes` to the config file:

`~/.android/avd/<emulator-device-name>.avd/config.ini`

Install test environment

Install Android SDR only

https://developer.android.com/studio/releases/platforms.html

OR

Install Android Studio

OR

Install GenyMotion

https://www.genymotion.com

Install APIs

```
$ android
```

and choose API version to install with gui

List available Android targets

```
$ android list targets
```

Create an Android image for a target

```
$ android create avd -n test -t 3
```

```
$ android create avd -n test -t 3 -abi default/x86
```

Start emulator

Start the emulator for the created image @test:

```
emulator @test
```

```
$ adb shell
```

Install BusyBox

```
$ adb push busybox /data/local
$ su
# mount -o remount,rw -t yaffs2 /dev/block/mtdblock3 /system
# mkdir /system/xbin
# cat /data/local/busybox > /system/xbin/busybox
# chmod 755 /system/xbin/busybox
# busybox --install /system/xbin
# mount -o ro,remount -t yaffs2 /dev/block/mtdblock3 /system
# sync
# reboot
```

Install Burp cert on emulator

Create an sdcard for the emulator:

```
$ mksdcard -l pisd 1G /tmp/sdcard
```

Launch emulator with -sdcard option:

```
$ emulator-x86 -sdcard /tmp/sdcard -avd test -qemu -m 1024 -enable-kvm -http-proxy
"http://192.168.1.2:8080"
```

Get Portswigger cert from visiting a page in browser, export and save as Portswigger.crt

Note: must have .crt extension for android to recognise it on sdcard

```
$ adb push Portswigger.crt /mnt/sdcard
```

Next, go to “Settings” and install from sdcard

Installing Drozer

Install drozer app on test device and run it.

From your laptop, connect to the app:

```
$ adb forward tcp:31415 tcp:31415
```

```
$ drozer console connect
```

```
$ drozer console --server 10.0.2.15:31415 connect
```

Copy an APK off device

Installed APKs are located in /data/app on the device

```
$ adb pull /data/app/AppName-1.apk .
```

Extracting Java code from an APK

Extract the APK using APKTool. Run: `apktool d AppName.apk`

Extract the classes.dex file found in the APK file. Run: `jar xvf classes.dex`

Extract the classes from classes.dex file. Run: `dex2jar classes.dex`

Extract the classes.dex.dex2jar.jar. Run: `jar xvf classes.dex.dex2jar.jar`

Browse Java code: Open the extracted jar file in `jd-gui`

First, unpack the apk using unzip

```
$ unzip AppName.apk
```

```
$ axmlprinter AndroidManifest.xml
```

Check package information

Using Drozer:

```
dz> run app.package.list
```

```
dz> run app.package.attacksurface com.targetpackage
```

```
dz> run app.provider.info -a com.targetpackage
```

Check for the debug flag

Using Drozer:

```
dz> run app.package.debuggable
```

OR

Check manifest file.

Check for SQL injection

Using Drozer:

```
dz> run scanner.provider.injection -a com.targetpackage
```

Check for path traversal

Using Drozer:

Check the Content Providers

Content providers provide access to structured data.

Can be affected by: SQLi, directory traversal

Useful Drozer commands for working with content providers:

```
dz> run app.provider.info -a com.targetpackage
dz> run app.provider.finduri com.targetpackage
dz> run scanner.provider.finduris -a com.targetpackage
dz> run scanner.provider.traversal -a com.targetpackage
dz> run scanner.provider.injection -a com.targetpackage
dz> run app.provider.query content://com.targetpackage...
dz> run app.provider.query content://com.targetpackage... --vertical --selection "'"
dz> run app.provider.query content://com.targetpackage... --projection "* FROM
SQLITE_MASTER WHERE type='table';--"
dz> run app.provider.query content://com.targetpackage... --projection "* FROM Key;--
"
dz> run app.provider.read content://com.targetpackage...
dz> run app.provider.read content://com.targetpackage../etc/hosts
dz> run app.provider.download
content://com.targetpackage../data/data/com.targetpackage]/databases/database.db
```

Activities provide user facing components.
Can be affected by UI redressing attacks e.g. tap jacking etc

Useful Drozer commands for working with activities:

```
dz> run app.activity.info -a com.targetpackage
dz> run app.activity.start --component com.targetpackage
com.targetpackage.ActivityName
```

Check the Services

Useful Drozer commands for working with services:

```
dz> run app.service.info -a com.targetpackage
dz> run app.service.send com.targetpackage com.targetpackage.ServiceName
```

Check the Intents

Intents can be implicit or explicit.
Check Manifest file for public intents, e.g.

```
<receiver android:name="my.special.receiver">
<intent-filter>
<action android:name="my.intent.action" />
</intent-filter>
</receiver>
```

```
<receiver android:name="my.special.receiver"
android:exported=false>
...
</receiver>
```

OR

```
<receiver android:name="my.special.receiver"
android:exported=false>
android:permission="my.own.permission"
...
</receiver>
```

Check the Broadcast Receivers

Broadcast receivers handle implicit intent messages or system wide events

Useful Drozer commands for working with broadcast receivers:

```
dz> run app.broadcast.info -a com.PackageName.AppName
dz> run app.broadcast.send --action [name of action from manifest file] --component
com.PackageName.AppName com.PackageName.AppName.push.GCMPushReceiver
dz> run app.broadcast.send --action [name of action from manifest file] --component
com.PackageName.AppName.push.GCMPushReceiver --extra string paramName paramValue --
extra sting paramName2 paramValue2
```

```
<uses-permission android:name="android.permission.BROADCAST_STICKY"/>
```

Check the files stored on the device

Grep for:

```
http, https, ://, user, pass, hmac, login
```

Check for insecure data storage on the device

Perform a search for files relating to package name:

```
root@android:/ # find / -name com.PackageName -print
```

Check the SD Card:

```
/mnt/sdcard
```

```
/mnt/sdcard/Android/data/com.PackageName
```

Check the data directory:

```
/data/data/com.PackageName
```

Check the database files on the device:

```
sqlite> select * from StoredProperties;
```

```
429482317|OPEN_SESAME|pTpkKAqfa9ly2oLqmivPIMKZDhTVlPOMLC9Ogi3c8Z0fkXL+H8u66ytJ0aFh+QY4
N4rX9Iq5qVuKnCon0a+lirekLJD3/6uoh/e5vaNptxI=
```

`/data/data/packagename`

`cp name.db /mnt/sdcard`

`$ adb pull /mnt/sdcard/name.db .` (otherwise won’t have perms to copy)

Check the log files:

`logcat -b events`

`/data/anr`

`/data/dontpanic`

`/data/tombstones`

`dmesg`

Check the device memory

Check memory stats:

`$ adb shell dumpsys meminfo > mem.txt`

`$ adb shell dumpsys meminfo 'com.PackageName'`

Dump the memory:

`$ adb shell dumpsys > mem.txt`

`$ adb shell dumpstate > mem.txt` (can show params passed to intents etc.)

Checking memory and logcat together:

`$ adb shell bugreport > bugreport.txt`

`http://domain.com/api/save.php?t=" + paramString1 + "&u=" + paramString2);`

reflection

etc.

Check for WebViews

Search the decompiled folder for:

addJavascriptInterface

```
grep -r -n -i --include=*.java addJavascriptInterface *
```

```
grep -r -i --include=*.java \@JavascriptInterface *
```

shouldOverrideUrlLoading

```
grep -r -n -i --include=*.java shouldOverrideUrlLoading *
```

Use Drozer module:

```
run ex.scanner.jsifenum -a com.targetpackage
```

Check the transport security

Use the emulator to dump traffic to a pcap flle with the option:

```
-tcpdump
```

Use the emulator to proxy traffic with the option:

0xsh / July 1, 2016 / android, cheatsheet, pentesting

Leave a Reply

Your email address will not be published. Required fields are marked *

COMMENT

NAME *

WEBSITE

POST COMMENT

PREVIOUS

Wifi Pentesting Command Cheatsheet

NEXT

Installing multimon-ng Pager Decoder Tool on Ubuntu (16.04 LTS)

RECENT POSTS

- Data exfiltration via PSK31 without GNU Radio
- Data exfiltration via PSK31 with GNU Radio
- Android Testing Environment Cheatsheet (Part 2)
- Android Testing Environment Cheatsheet (Part 1)
- Listening to Iridium satellite traffic on Ubuntu (16.04 LTS)

RECENT COMMENTS

CATEGORIES

- android
- cheatsheet
- infrastructure
- pentesting
- sdr

ARCHIVES

- March 2017
- October 2016
- September 2016
- July 2016

Search …

META

- Log in

- Comments RSS
- WordPress.org