



Auditing Your APKs Like a Black Hat Hacker

Evading Android Application Vulnerability Exploitation

AndroidTO | 11.06.18



\$whoami

KRISTINA BALAAM

- Security Intelligence Engineer @ Lookout
- Formerly Application Security Engineer @ Shopify
- MSc. Student in Information Security Engineering, SANS Tech



@CHMODXX_



@CHMODXX blog.chmodxx.net

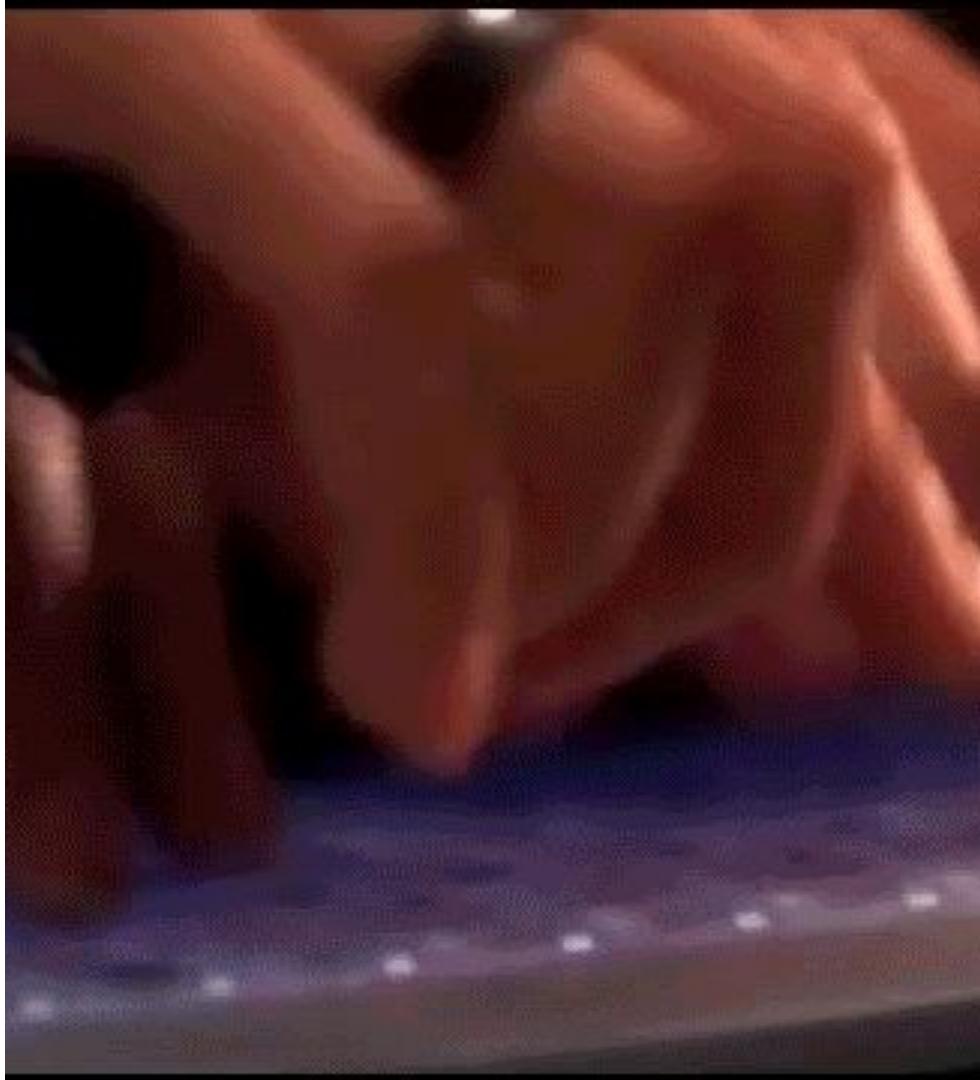
DISCLAIMER

- ✓ The information provided is intended to educate mobile developers & application security engineers to better protect their applications.
- ✓ Use these techniques to build solid apps and audit those within your company.
- ✓ If you want to hack for \$\$ and fame, *please* join a bug bounty program like **HackerOne** or **BugCrowd**.
- ✓ *Be responsible and disclose vulnerabilities* □



AGENDA

1. Intro to Reverse Engineering
2. Common Attack Surfaces
3. Finding Common Vulns
4. Fixing Common Vulns
5. Continued Learning



> INTRO TO REVERSE ENGINEERING



REVERSE ENGINEERING



The process of analyzing a subject system to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction

IEEE

WHY DO WE REVERSE ENGINEER OUR APPS?

<https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04c-Tampering-and-Reverse-Engineering.md>

- ✓ To enable black-box testing of mobile apps.
- ✓ To enhance static analysis in black-box security testing.
- ✓ To assess resilience against reverse engineering.



@CHMODXX_



@CHMODXX

blog.chmodxx.net

I ❤️ LINUX

These examples are going to use Linux but most - if not all of these tools - will run on other OSes as well.



KALI LINUX

"Hacking" distro; super popular, well-supported and documented



SANTOKU LINUX

Mobile-specific security distribution; comes loaded with all the tools you need



UBUNTU

User-friendly, great UI, well-maintained and documented



LINUX MINT

Another favourite amongst the Linux internet population; "Vanilla" Linux

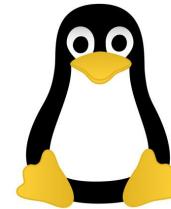


@CHMODXX_



@CHMODXX blog.chmodxx.net

SETTING UP THE ENVIRONMENT



Android SDK

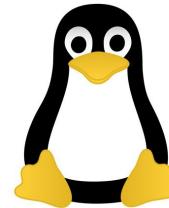
```
$ sudo apt-get install android-sdk  
$ sudo ln -s /usr/share/android-sdk/platform-tools/adb /bin/adb  
$ sudo chmod +x /usr/share/android-sdk/tools/android
```

- ★ **adb** interacts with devices, emulators to give a shell, read logs, etc.
- ★ **android** manages emulators

SETTING UP THE ENVIRONMENT

Creating an Android Emulator

```
$ android sdk # install SDK platforms and tools  
$ android avd # create an emulator  
$ emulator -avd [emulator name] # run your emulator
```



Getting an Interactive Shell

```
$ adb devices # list all devices on your computer  
$ adb -s DEVICE_ID shell # start an interactive shell for DEVICE_ID
```

- ★ Emulators get root by default (!!!) but don't always work properly for hardware tests

SETTING UP THE ENVIRONMENT



- ★ Download from your connected device via adb

```
$ adb pull [REMOTE] [LOCAL]
```

- ★ Third-party download sites like
<https://apkpure.com/> and <https://apkbucket.net>**
** Sketchy. Proceed with caution.

APKTool

- ✓ Converts resources back to pretty much their original form
- ✓ DEX (binary Dalvik bytecode) → **smali** (human readable)
- ✓ Allows you to decompile and recompile APKs



"A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc." - APKTool

<https://ibotpeaches.github.io/Apktool/>

```
$ wget  
https://bitbucket.org/iBotPeaches/apktool/downloads/apktool_2.3  
.3.jar  
$ wget  
https://raw.githubusercontent.com/iBotPeaches/Apktool/master/  
scripts/linux/apktool  
  
$ mv apktool_2.3.3.jar apktool.jar  
$ mv -t /usr/local/bin/ apktool.jar apktool  
$ chmod +x apktool; chmod +x apktool.jar  
$ apktool d [APPLICATION].apk# decompile the apk
```

dex2jar

- ✓ Converts .dex files to .class files and packages them into a jar.
- ✓ Necessary step for analysis in JD-GUI

pxb1988 / dex2jar

Watch 375 Star 4,922 Fork 1,072

Code Issues 162 Pull requests 7 Projects 0 Wiki Insights

Tools to work with android.dex and java.class files

556 commits 2 branches 32 releases 6 contributors Apache-2.0

Branch: 2.x New pull request Create new file Upload files Find file Clone or download

pxb1988 translate DEX_037 to V1_8 ... Latest commit b577140 on Jul 4

Commit	Message	Time Ago
d2j-base-cmd	remove maven pom.xml	4 months ago
d2j-j6	support git/hg revision in meta-info	3 years ago
d2j-jasmin	fix build error	4 months ago
d2j-small	fix build error	4 months ago
dex-ir	insert a Nop between two LabelStmt if both have phis	2 months ago
dex-reader-api	[dex038] write class version 1.7 if dex version > DEX_037	2 months ago
dex-reader	[dex038] write class version 1.7 if dex version > DEX_037	2 months ago
dex-tools	[dex038] write class version 1.7 if dex version > DEX_037	2 months ago
dex-translator	translate DEX_037 to V1_8	a month ago
dex-writer	remove maven pom.xml	4 months ago
gradle(wrapper)	update gradle to 4.0	4 months ago
.hgignore	ignore build folder for gradle	4 years ago
.htags	clean exec mode from file	5 years ago
translate script	run libD for translation	8 months ago

<https://github.com/pxb1988/dex2jar>

```
$ git clone https://github.com/pxb1988/dex2jar.git
```

JDGU

- ✓ Displays a somewhat accurate representation of the .class files converted by Dex2Jar



[jd-gui-0.3.5.linux.i686.tar.gz](#)

Size : 1.1 MB

MDS checksum : 3E82FFCB98508971D96150CF57837B13



[jd-gui-0.3.5.osx.i686.dmg](#)

Size : 1.5 MB

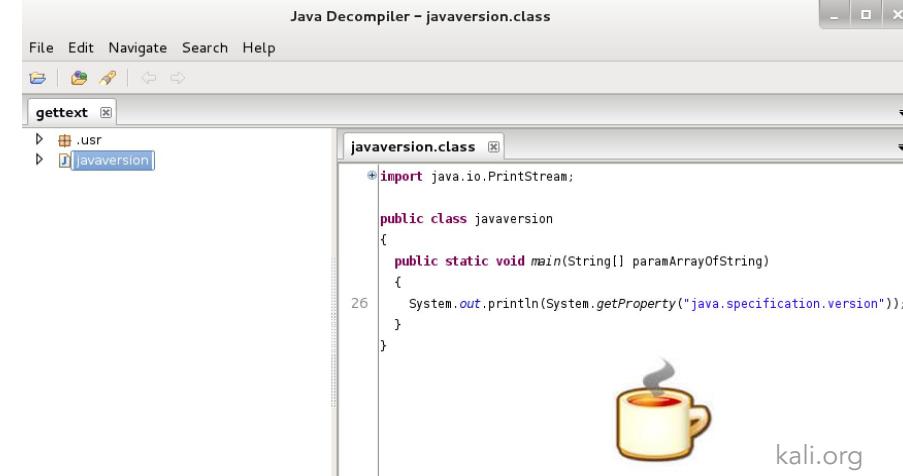
MDS checksum : 203605F4B264294E7861D4538E2BC9EA



[jd-gui-0.3.6.windows.zip](#)

Size : 770 KB

MDS checksum : AC391B87FBEB6A10C17EEE5BF085EB37



The screenshot shows the JD-GUI interface. On the left, there's a file tree with 'gettext' and 'javaversion.class'. The right pane displays the decompiled code for 'javaversion.class':

```
+import java.io.PrintStream;  
  
public class javaversion  
{  
    public static void main(String[] paramArrayOfString)  
    {  
        System.out.println(System.getProperty("java.specification.version"));  
    }  
}
```

A small coffee cup icon is in the bottom right corner.

kali.org

JD is a Decompiler for the Java programming language. JD is provided as a GUI tool as well as in the form of plug-ins for the Eclipse and IntelliJ IDEA integrated development environments.

<https://github.com/java-decompiler/jd-gui>

```
$ wget  
https://github.com/java-decompiler/jd-gui/releases/download/v1.4.0/jd-gui_1.4.0-0_all.deb  
$ sudo dpkg -i jd-gui_x.x.x-x_all.deb
```



@CHMODXX_



@CHMODXX

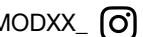
blog.chmodxx.net

DIVA - Damn Insecure (and) Vulnerable App

<https://github.com/payatu/diva-android> | <https://github.com/tjuxiang92/Android-Vulnerabilities/blob/master/diva-beta.apk>

- ✓ Created by Aseem Jakhar of Payatu (<http://payatu.com>)
- ✓ Open Source
- ✓ Allows us to perform RE demos without upsetting developers

Examples of: Insecure logging, hard-coding vulnerabilities, insecure data storage, input validation, access control vulnerabilities



@CHMODXX_ @CHMODXX blog.chmodxx.net

Drozer

- ✓ Released 2012 at Blackhat EU
- ✓ **Finds vulnerabilities / Provides exploits & payloads**
- ✓ **Agent** (runs on the device and facilitates testing), **Console** (CLI to interact with the device), **Server** (routes sessions between console & agents)

The logo for Drozer, featuring the word "drozer" in a lowercase, sans-serif font. The letters are white with a slight shadow, set against a solid orange background.

"Drozer allows you to assume the role of an Android app, and to interact with other apps, through Android's Inter-Process Communication (IPC) mechanism, and the underlying operating system." - MWR Labs

<https://labs.mwrinfosecurity.com/tools/drozer/>

```
$ git clone https://github.com/mwrlabs/drozer/  
$ cd drozer  
$ make deb  
$ sudo dpkg -i drozer-2.x.x.deb  
$ adb install drozer-agent-2.x.x.apk  
$ adb forward tcp:31415 tcp:31415  
$ drozer console connect
```

> COMMON ATTACK SURFACES



PERMISSIONS!



PERMISSIONS!

BROADCAST
RECEIVERS!



PERMISSIONS!

BROADCAST
RECEIVERS! IPCS!





PERMISSIONS!

**BROADCAST
RECEIVERS!**

**STORAGE!
S!**

The text is composed of several words in different colors and orientations. The word 'PERMISSIONS!' is in large, bold black letters at the top. Below it, 'BROADCAST' and 'RECEIVERS!' are written in pink and purple slanted letters. At the bottom, 'STORAGE!' is in large green letters, and 'S!' is in purple below it.



PERMISSIONS!

BROADCAST
RECEIVERS!

STORAGE!

INTENTS!



PERMISSIONS!
WEB STORAGE!
INTERVIEWS!
RECEIVED!
BROADCAST!



PERMISSIONS!

WEB STORAGE!

INTERVIEWS!

BROADCAST RECEIVERS!

CONTENT PROVIDERS!



PERMISSIONS!

LOGGING!

BROWSE HISTORY!

RECENT VIDEOS!

CONTENT PROVIDERS!

STORAGE!

INTENT



PERMISSIONS!
BROWSE LIST!
RECEIVE NEWS!
LOGGING!
CONTENT PROVIDERS!
DATA VALIDATION!
CONTENT VIEWS!
STORAGE!



PERMISSIONS!
W^{LIST}
BROW^{SERS!}
RECE^{IVE}
LOGGING!
CONTEN^T
DATA^{VALIDATION!}
VIEWS!
STORAGE!
CONTENT^{PROVIDERS!}
OBfuscation!



PERMISSIONS!
WIGGLY
BROWNS
RECIPIENTS
LOGGING!
CONTAMINATED
DATA VALIDATION!
VIEWERS!
STORAGE!
CONTENT PROVIDERS!
OBSCURATION!



PERMISSIONS!
WIGGLY SPHERES!
BROWNS! DEDICATORS!
RECHARGING!
LOGGING!
CONTENT VALIDATIONS!
DATA FALLOWS!
LOBFUSCATION!

A small white church emoji with a brown roof and a cross on top. It has three arched windows with stained glass and a central double door.A brown poop emoji with a smiling face, white eyes, and a wide-open mouth.

> FINDING & FIXING COMMON VULNS



QHINDO



Arthur Elmer

Aquatic Park

Can of Whoopie

Cartoon

David the G

Fabio Fazio

Flying Tacos

GeoSource

Kidzoo

Quigley Chat

Events

Music

PERMISSIONS

APPLICATION PERMISSION

- Developers often ask for more permissions than they need.
- Most users will accept anything they're asked.
- Err on the side of caution and do whatever you can to make the attack surface smaller.

```
dz> run app.package.info -a [PACKAGE-NAME]
```

What to Do:

- Follow the Principle of Least Privilege
- Define permissions with signature protection so no other applications can access components or request permissions
- Make sure the permissions you're requesting are really necessary -- let native apps handle functionality
 - Eg. Only need READ permissions? Don't grant READWRITE.



@CHMODXX_



@CHMODXX

blog.chmodxx.net

PERMISSIONS

FILE PERMISSIONS

- Only for files stored externally.
- You can define file permissions in `AndroidManifest` *and* in the code.
- Malware loves searching for files in SD Cards

What to Do:

- Don't give `MODE_WORLD_READABLE` or `MODE_WORLD_WRITABLE` permissions if you can help it
 - they allow other applications to access the file
- Share files between the Content Provider; avoid external storage where you can



@CHMODXX_ @CHMODXX blog.chmodxx.net

CHECKING THE MANIFEST

```
* Press ? for help

.. (up a dir)
</Desktop/DIVA/
> lib/
> res/
AndroidManifest.xml
classes.dex
diva-beta.apk
resources.arsc

5 <83>^F@^S<83>^F@</83>^F@<83>^F@C<83>
F@<83>^F@W<83>^F@i<83>^F@<83>^F@<9
><83>^F@j<83>^F@%<83>^F@C<83>^F@y<83>
F@H<84>^F@<84>^F@*<84>^F@G<84>^F@S
84>^F@n<84>^F@<89><84>^F@<8d><84>^F@E
84>^F@<84>^F@<84>^F@I<84>^F@<84>^F@B
@U<84>^F@<A<85>^F@<85>^F@<85>^F@<85>
<85>^F@Z<85>^F@)<85>^F@.<85>^F@<4<85>
F@<85>^F@<85>^F@B<85>^F@I<85>^F@P<
5>^F@<85>^F@<85>^F@<85>^F@j<85>^F@<85>^F@
s<85>^F@x<85>^F@<80><85>^F@<88><85>^F@
<90><85>^F@<88><85>^F@<85>^F@<85>^F@
`<85>^F@%<85>^F@<85>^F@<85>^F@<85>^F@<85>
F@Y<85>^F@<85>^F@<85>^F@<85>^F@<85>^F@<85>
86>^F@H<86>^F@N<86>^F@&P<86>^F@&[<86>^F@%<86
^F@<86>^F@&8<86>^F@<86>^F@<86>^F@<86>^F@<86
86>^F@q<86>^F@<86>^F@<86>^F@<86>^F@<93>
86>^F@<9b><86>^F@<E<86>^F@<86>^F@<86>^F@<86
^F@&A<86>^F@&I<86>^F@Y<86>^F@&@<86>^F@&T
86>^F@&[<86>^F@&B<87>^F@& <87>^F@&Q<87
^F@&[<87>^F@&X<87>^F@&[<87>^F@&K<87>^F@&
<87>^F@&I<87>^F@&91><87>^F@&I<87>^F@&[<87>^F@&
>^F@&3<87>^F@&%<87>^F@&E<87>^F@&0<87>^F@&
```

CHECKING THE MANIFEST

```
* values-sr/
* values-sv/
* values-sw/
* values-sw600dp-v13/
* values-ta-rIN/
* values-te-rIN/
* values-th/
* values-tl/
* values-tr/
* values-uk/
* values-un-rPK/
* values-uz-rUZ/
* values-v11/
* values-v12/
* values-v14/
* values-v17/
* values-v18/
* values-v21/
* values-v22/
* values-v23/
* values-vi/
* values-w300dp-v13/
* values-w480dp-v13/
* values-w500dp-v13/
* values-w600dp-v13/
* values-w720dp-v13/
* values-w820dp-v13/
* values-xlarge-lend-v4/
* values-xlarge-v4/
* values-zh-rCN/
* values-zh-rHK/
* values-zh-rTW/
* values-zu/
* values/
    attrs.xml
    bools.xml
    colors.xml
    dimens.xml
    drawables.xml
    ids.xml
    integers.xml
1 !!brut.androlib.meta.MetaInfo
2 apkFileName: divo-beta.apk
3 compressionType: false
4 doNotCompress:
5 - arsc
6 - res/drawable-hdpi-v4/abc_ab_share_pack_mtrl_alpha.9.png
7 - png
8 - res/drawable-hdpi-v4/abc_btn_switch_to_on_mtrl_00001.9.png
9 - res/drawable-hdpi-v4/abc_btn_switch_to_on_mtrl_00012.9.png
10 - res/drawable-hdpi-v4/abc_cab_background_top_mtrl_alpha.9.png
11 - res/drawable-hdpi-v4/abc_list_divider_mtrl_alpha.9.png
12 - res/drawable-hdpi-v4/abc_list_focused_holo.9.png
13 - res/drawable-hdpi-v4/abc_list_longpressed_holo.9.png
14 - res/drawable-hdpi-v4/abc_list_pressed_holo_dark.9.png
15 - res/drawable-hdpi-v4/abc_list_pressed_holo_light.9.png
16 - res/drawable-hdpi-v4/abc_list_selector_disabled_holo_dark.9.png
17 - res/drawable-hdpi-v4/abc_list_selector_disabled_holo_light.9.png
18 - res/drawable-hdpi-v4/abc_menu_hardkey_panel_mtrl_mult.9.png
19 - res/drawable-hdpi-v4/abc_popup_background_mtrl_mult.9.png
20 - res/drawable-hdpi-v4/abc_scrubber_primary_mtrl_alpha.9.png
21 - res/drawable-hdpi-v4/abc_scrubber_track_mtrl_alpha.9.png
22 - res/drawable-hdpi-v4/abc_spinner_mtrl_am_alpha.9.png
23 - res/drawable-hdpi-v4/abc_switch_track_mtrl_alpha.9.png
24 - res/drawable-hdpi-v4/abc_tab_indicator_mtrl_alpha.9.png
25 - res/drawable-hdpi-v4/abc_textfield_activated_mtrl_alpha.9.png
26 - res/drawable-hdpi-v4/abc_textfield_default_mtrl_alpha.9.png
27 - res/drawable-hdpi-v4/abc_textfield_search_activated_mtrl_alpha.9.png
28 - res/drawable-hdpi-v4/abc_textfield_search_default_mtrl_alpha.9.png
29 - res/drawable-ldrtl-hdpi-v4/abc_spinner_mtrl_am_alpha.9.png
30 - res/drawable-ldrtl-hdpi-v4/abc_spinner_mtrl_am_alpha.9.png
31 - res/drawable-ldrtl-xhdpi-v4/abc_spinner_mtrl_am_alpha.9.png
32 - res/drawable-ldrtl-xxhdpi-v4/abc_spinner_mtrl_am_alpha.9.png
33 - res/drawable-ldrtl-xxxhdpi-v4/abc_spinner_mtrl_am_alpha.9.png
34 - res/drawable-mdpi-v4/abc_ab_share_pack_mtrl_alpha.9.png
35 - res/drawable-mdpi-v4/abc_btn_switch_to_on_mtrl_00001.9.png
36 - res/drawable-mdpi-v4/abc_btn_switch_to_on_mtrl_00012.9.png
37 - res/drawable-mdpi-v4/abc_cab_background_top_mtrl_alpha.9.png
38 - res/drawable-mdpi-v4/abc_list_divider_mtrl_alpha.9.png
39 - res/drawable-mdpi-v4/abc_list_focused_holo.9.png
40 - res/drawable-mdpi-v4/abc_list_longpressed_holo.9.png
41 - res/drawable-mdpi-v4/abc_list_pressed_holo_dark.9.png
c:\holam\Desktop\DIVA\divo-beta.apktool.yml
```



@CHMODXX_



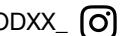
@CHMODXX

blog.chmodxx.net

IPCS

(INTERPROCESS COMMUNICATIONS)

- Services, Activities, BroadcastReceivers, ContentProviders
 - Endpoints aren't always secured
 - They act as data sinks *and* sources.
- **Broadcast messages allow any application to receive an intent!**
 - Malicious apps could gain access to another's data
- **ContentProviders:** could expose access to data and directory traversal or SQL injection attacks; when not permissions protected, any application can invoke
- **Activities:** could be used in a UI-redressing attack
- **BroadCast Receivers:** could be hijacked to intercept an Intent & its data; null values can also be sent to DoS applications
- **Services:** Could expose application-specific functionality



IPCS

(INTERPROCESS COMMUNICATIONS)

What to Do:

- Share files using **ContentProvider**; avoid external storage (like SDCards) where you can
- Android versions before 4.2 export content providers by default. Ensure this is false for any apps whose targeted SDK version is <= 16
- Even content providers that *aren't* exported can be accessed by privileged users
- Similarly, exported activities require no permissions for interaction
- When using ContentProviders, always ensure a permission is set for the required application
- Sanitize inputs or use prepared statements with ContentProviders to avoid SQL injection attacks
- Use explicit intents wherever possible
- Use custom permissions with services, too (can be checked by service when external service makes a request)
- Use the local broadcast manager for local intents
 - No other application can access the data
- `sendBroadcast(intent);` and `sendStickyBroadcast(intent);` are susceptible to IPC sniffing. Use intents signed with permissions so an unauthorized app can't receive the intent!
- Check the data being received from any broadcast and ensure that it's valid!



IPCS

USEFUL DROZER COMMANDS

```
dz> run app.provider.info -a [PACKAGE NAME]  
# list all exported content providers
```

```
dz> run app.provider.info -a [PACKAGE NAME] -u  
# list all non-exported content providers
```

```
dz> run app.activity.info -a [PACKAGE NAME]  
# list all exported activities
```

```
dz> run scanner.provider.injection -a  
# scan for sql injection vulns
```



@CHMODXX_



@CHMODXX

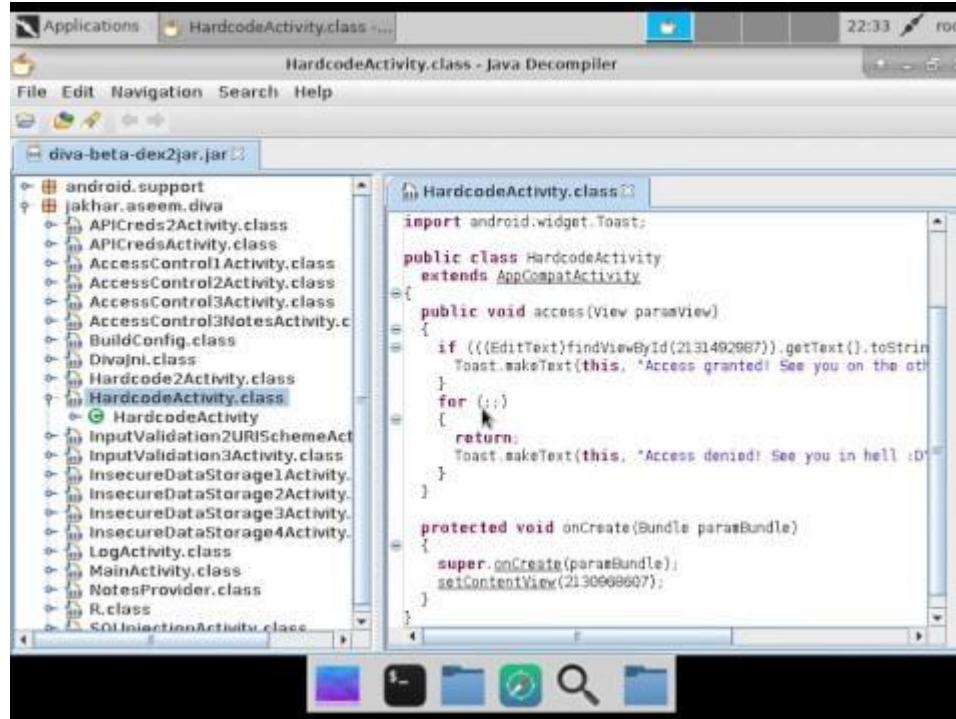
blog.chmodxx.net

ANALYZING SOURCE CODE

```
tor-m-kbal01:dex2jar-2.0 kristina.baloam$ ls
d2j-baksmali.bat           d2j-dex2smali.sh
d2j-baksmali.sh            d2j-jar2dex.bat
d2j-dex-recompute-checksum.bat d2j-jar2dex.sh
d2j-dex-recompute-checksum.sh d2j-jar2jasmin.bat
d2j-dex2jar.bat            d2j-jar2jasmin.sh
d2j-dex2jar.sh             d2j-jasmin2jar.bat
d2j-dex2smali.bat          d2j-jasmin2jar.sh
tor-m-kbal01:dex2jar-2.0 kristina.baloam$
```



ANALYZING SOURCE CODE



The screenshot shows the JD-GUI Java decompiler interface. The title bar reads "HardcodeActivity.class - Java Decomplier". The left pane displays the file structure of "diva-beta-dex2jar.jar", with "HardcodeActivity.class" selected. The right pane shows the decompiled Java code for "HardcodeActivity.class".

```
import android.widget.Toast;

public class HardcodeActivity extends AppCompatActivity
{
    public void access(View paramView)
    {
        if (((EditText)findViewById(2131492987)).getText().toString()
            .equals("Access granted"))
        {
            Toast.makeText(this, "Access granted! See you on the other side", 0).show();
        }
        else
        {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }

    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130908807);
    }
}
```



@CHMODXX_



@CHMODXX blog.chmodxx.net

DIVA SQL Injection

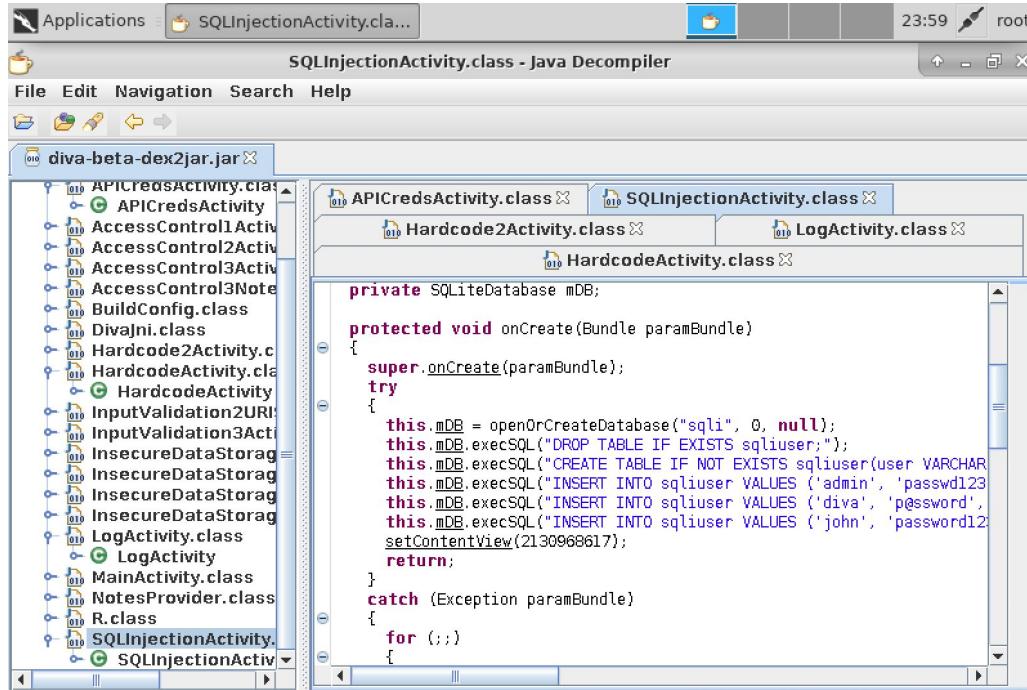
7. Input Validation Issues - Pa...

Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

Enter user name to search

SEARCH



The screenshot shows a Java Decomiler interface with the title "SQLInjectionActivity.class - Java Decompiler". The left pane displays a file tree for "diva-beta-dex2jar.jar" containing various Java classes like APIcredsActivity, AccessControl1Activity, etc. The right pane shows the decompiled code for the SQLInjectionActivity class. The code includes database operations such as opening a database, dropping a table if it exists, creating a table named "sqliuser" with columns "user" and "password", and inserting three rows of data: ('admin', 'passwd123'), ('diva', 'password'), and ('john', 'password12').

```
private SQLiteDatabase mDB;
protected void onCreate(Bundle paramBundle)
{
    super.onCreate(paramBundle);
    try
    {
        this.mDB = openOrCreateDatabase("sql", 0, null);
        this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
        this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'password'");
        this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password12");
        setContentView(2130968617);
        return;
    }
    catch (Exception paramBundle)
    {
        for (;;)
        {
    }
```



@CHMODXX_



@CHMODXX blog.chmodxx.net

DIVA SQL Injection

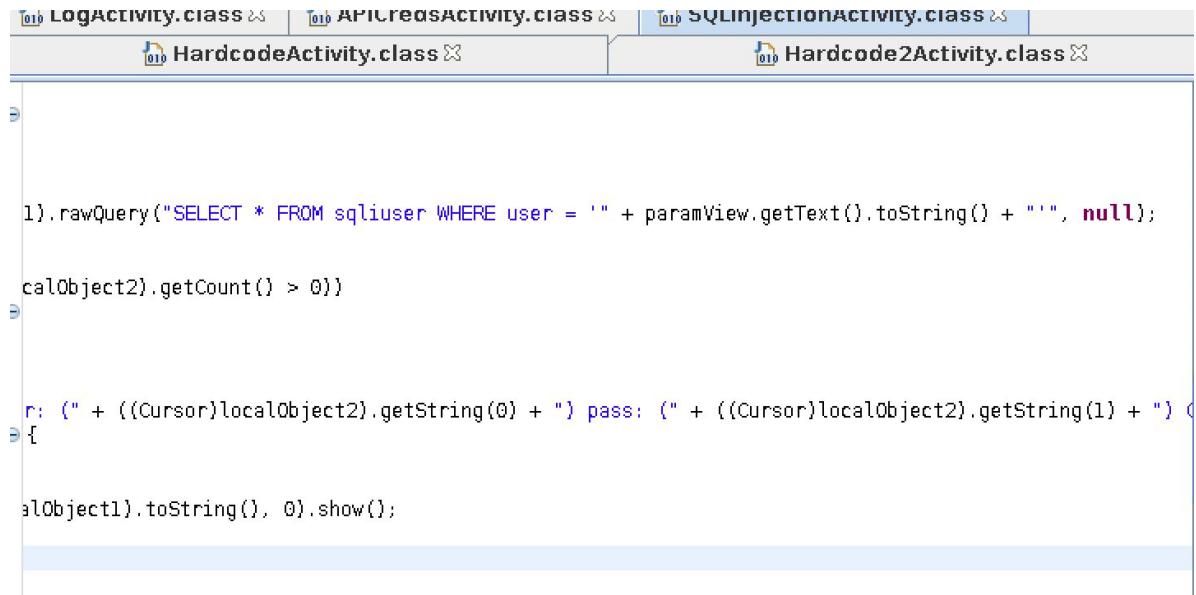
7. Input Validation Issues - Pa...

Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

Enter user name to search

SEARCH



The screenshot shows an Android Studio interface with multiple tabs at the top: LOGACTIVITY.class, APICREASACTIVITY.class, and SQLINJECTIONACTIVITY.class (which is currently selected). Below the tabs, two Java files are visible: HardcodeActivity.class and Hardcode2Activity.class. The SQLINJECTIONACTIVITY.java file contains the following code:

```
1).rawQuery("SELECT * FROM sqliuser WHERE user = '" + paramView.getText().toString() + "'", null);

localObject2).getCount() > 0))

if (" + ((Cursor)localObject2).getString(0) + ") pass: (" + ((Cursor)localObject2).getString(1) + ") {
    localObject1).toString(), 0).show();
```



@CHMODXX_



@CHMODXX blog.chmodxx.net



REAL WORLD EXAMPLE

Samsung Kies, Galaxy S3

- Kies was highly privileged: connects mobile phone to your PC
- Had a BroadCast receiver that restored APKs from the SDcard
- Tldr; Kies has a call chain that iterates through the sdcard/restore directory and installs every APK
- A researcher was able to add their app to the SD card by exploiting a WRITE_EXTERNAL_STORAGE privilege issue with the clipboard service on the S3, and then had Kies call that function with an intent

<http://sh4ka.fr/android>

INSECURE STORAGE

- **Apps are super easy to RE**
 - .apks are basically just .zip files
- **Data should be stored in either:**
 - /data/data/<package>
 - Only accessible by the application unless it gives permission or if the device is rooted
 - /sdcard
 - Accessible by everyone
- Process information can be dumped to access sensitive info.
- Don't embed any encryption key in source code.
- If an attacker has access to a phone, and the memory isn't cleared after the app is closed, they could access anything stored.
- WebViews allow HTML data to cache locally.



@CHMODXX_ @CHMODXX blog.chmodxx.net

INSECURE STORAGE

What to Do:

- Look for code that stores data locally: make sure it's not storing sensitive data
- When you absolutely *have* to store something client-side, make sure it's encrypted if it's sensitive
- When you're encrypting, use a *strong encryption algorithm*: avoid MD5/SHA1 hashing for passwords and instead use PBKDF2, bcrypt or scrypt.
- If you're using webviews, look at clearCache() or "no-cache" to prevent caching data altogether
- Re-initialize the Application class with dummy values once it closes to prevent saved information since it remains active even when the app is closed





REAL WORLD EXAMPLE

Skype, 2011

- Skype was storing contacts, profile, IM logs plus additional personal data at
`/data/data/com.skype.../files/<USERNAME>` in SQLite database and XML files
- Permissions were improperly set -- world readable and writable -- so any app could read them. Everything was unencrypted
- Skype also stored the usernames in static locations, so a malicious application could parse the `shared.xml` file, find the usernames and then use that value to find the files stored in data.

Reported by Justin Case (jcase),
<http://AndroidPolice.com>



REAL WORLD EXAMPLE

What's App, 2014

- Conversations stored on the phone's SD Card
- Researcher Bas Bosschert built a fake "malicious" app with a distracting load screen that requested the necessary permissions and downloaded user's WhatsApp data while the loading screen ran
- Database files were "encrypted", but easily decrypted with a Python script and a key found on an XDA Developers forum post

Reported by Bas Bosschert, [TechCrunch](#)

INSECURE COMMUNICATIONS

- **Web traffic inspection is an important part of the audit process**
 - (A surprising number of developers don't realize you can intercept web traffic -- especially on mobile)
- **Burp Suite** is a great (free) tool for setting up an intercepting proxy for mobile testing.
- **You can set up a proxy on an emulator.**
 - Set the Access Point Name to 10.0.2.2 and the port to the same as what's been specified by your Burp listener port.
- **New in Android P!**
 - TLS by default, but ability to opt out for legacy domains
 - <https://developer.android.com/training/articles/security-config>

Fix: Never send plaintext requests



@CHMODXX_



@CHMODXX

blog.chmodxx.net

WEBVIEWS

- **Renders web pages within the application**
- **WebView lets you break out of the app sandbox and bypass same origin.**
- **Also makes it possible to load malicious .js: any web page accessed by the frame in the app can call back to the application.** And can call back Java.
- **Loading cleartext content is the single biggest mistake you can make**
- **You see this a lot in apps with advertisements.**
- **How often does this happen?**
 - Stanford study in 2013:
 - 40k apps minimum using a “javascript bridge”
 - $\frac{1}{3}$ could be reached by untrusted content



WEBVIEWS

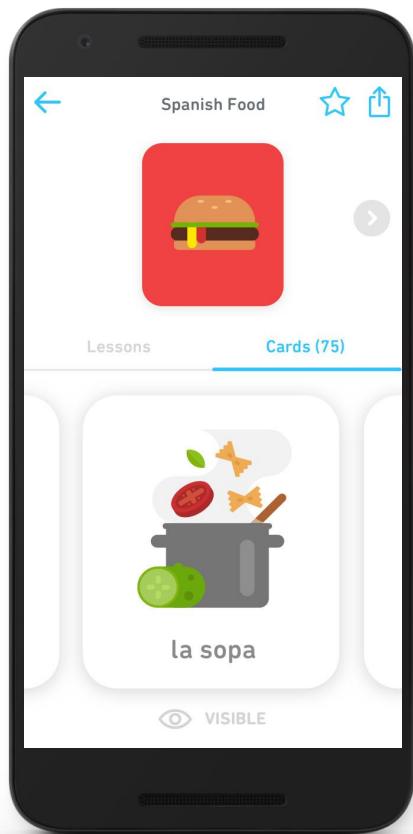


What to Do:

- Restrict users to the application domain
- Don't call `setJavaScriptEnabled()` until you absolutely have to process Javascript
- APIs 17+ require methods to be marked `@JavascriptInterface` for any method exposed to Javascript and this prevents malicious code from accessing the lower-level OS commands
- Create a whitelist of domains that are allowed to render content
- Send all traffic over SSL to prevent man-in-the-middle attacks where someone tries to inject script



@CHMODXX_ @CHMODXX blog.chmodxx.net



REAL WORLD EXAMPLE

TinyCards, 2018

RCE in TinyCards for Duolingo on Android

"TinyCards loads a website via webview when starting, but that site is loaded over http then redirected to https. A MITM attack that controls either the network or the DNS, can inject their own web content into the webview. You can confirm this by using an MITM proxy to capture the traffic." - @nightwatch-cybersecurity, HackerOne

Publicly Disclosed on HackerOne:
<https://hackerone.com/reports/281605>

LOGGING

- **Logging is great for debugging.***
 - *It's also great for hackers.
- **Even system processes (eg. ActivityManager) log detailed messages.**
- Even though the READ_LOGS permission was removed for 3P Applications after 4.1, rooted devices can still access it.

```
$ logcat  
# running from shell shows sys and app logs  
$ adb logcat  
# same as above, just direct
```





REAL WORLD EXAMPLE

Firefox, 2012

- Logged browsing activity, including plain text URLs and even session IDs
- Malicious application or attacker could use session IDs to hijack a victim's session

Reported by Neil Bergman

DIVA Insecure Logging

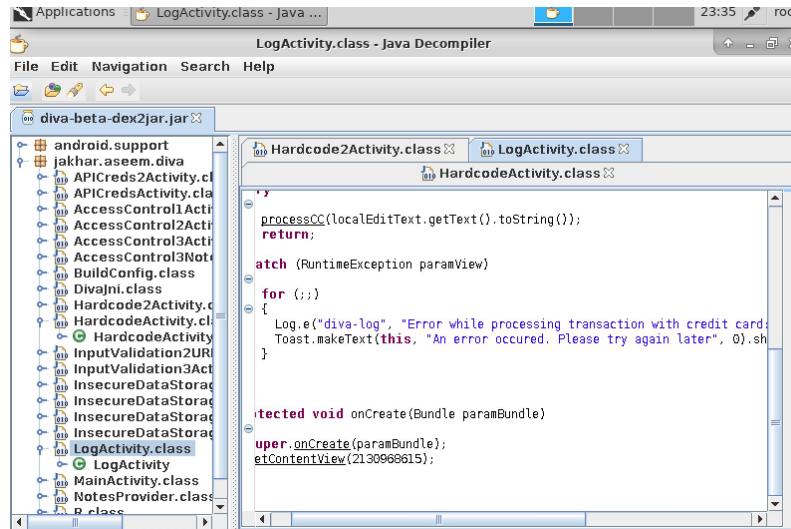
1. Insecure Logging

Objective: Find out what is being logged where/how and the vulnerable code.

Hint: Insecure logging occurs when developers intentionally or unintentionally log sensitive information such as credentials, session IDs, financial details etc.

Enter your credit card number

CHECK OUT



The screenshot shows the Java Decomiler interface with the file 'LogActivity.class - Java Decomiler' open. The code editor displays the following Java code:

```
processCC(localEditText.getText().toString());
return;

catch (RuntimeException paramView)
{
    Log.e("diva-log", "Error while processing transaction with credit card");
    Toast.makeText(this, "An error occurred. Please try again later", 0).show();
}

protected void onCreate(Bundle paramBundle)
{
    super.onCreate(paramBundle);
    setContentView(213096615);
}
```

```
D/MobileDataStateTracker( 469): default:
setPolicyDataEnable(enabled=true)

D/LightsService( 469): Excessive delay setting light:
86ms

D/dalvikvm( 1695): GC CONCURRENT freed 136K, 6% free
3845K/4060K, paused 7ms+4ms, total 93ms

E/diva-log( 1695): Error while processing transaction
with credit card: 0000000000

E/SoundPool( 469): error loading
/system/media/audio/ui/Effect_Tick.ogg
```



@CHMODXX_



@CHMODXX

blog.chmodxx.net

OBFUSCATION

It's easy to RE Android apps.

You can make it harder by obfuscating your code.

- **Pros:** Harder for people to steal your stuff or exploit
- **Cons:** Ongoing maintenance can be tricky.

What to Do:

- ProGuard obfuscates your code *lexically*: meaningful names replaced by machine-generated garble
- Using native code makes decompilation harder: attacker has to resort to assembly level reversing
 - BUT be aware: more susceptible to issues like buffer overflows
- Java reflection: code that's able to inspect other code -- makes it harder to trace what's happening in your app



PRIVATE KEYS

- **Used for:**
 - Signing apps
 - Encrypting https traffic
- **Private keys are included in apps ALL. THE. TIME.**
- **Java keystore**
 - Container for public/private keys and certificates
 - Password protection is optional (!!!)
 - No container-level encryption
 - Private keys housed within share same password as keystore container by default

What to Do:

- **DON'T STORE YOUR PRIVATE KEYS IN YOUR APP**
- **Cloud KMS:** Google offers cloud storage for secrets (<https://cloud.google.com/kms/>)
- If you don't trust Google, keep it somewhere else. Safe. Separate from the app.
- Google I/O 2018 announced "**StrongBox**": resistant to shared resource attacks, side channel attacks, physical attacks
 - Only some new devices that ship with Android P

PRIVATE KEYS



June 2017, an IT security journalist finds a private key in a CISCO app.

Will Dorman of Carnegie Mellon does a study analyzing apps for exposed private keys.

- Looked at 1.7M APKs

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119
Google Play signing private keys	1,948
Apple Push Services private keys	87
Apple iPhone Developer private keys	21
Apple iPhone Enterprise private keys	68

PRIVATE KEYS

June 2017, an IT security journalist finds a private key in a CISCO app.

Will Dorman of Carnegie Mellon does a study analyzing apps for exposed private keys.

- Looked at 1.7M APKs

PKCS#12 Password Cracking Statistics

Strategy	Count	Percent
Total	2119	100%
rockyou.txt password list	870	41.4%
Strings from app code	729	34.4%
Manual analysis	18	0.8%



PRIVATE KEYS



June 2017, an IT security journalist finds a private key in a CISCO app.

Will Dorman of Carnegie Mellon does a study analyzing apps for exposed private keys.
- Looked at 1.7M APKs

Java Keystore Password Cracking Statistics

Strategy	Count	Percent
Total	3215	100%
rockyou.txt password list	453	14.1%
Strings from app code	35	1.1%
hashcat-naive	1714	53.3%

Watch the BSidesSF Talk

<https://www.youtube.com/watch?v=-VjK0FMmGm4>

Carnegie Mellon University
Software Engineering Institute

Keep It Like a Secret: When Android Apps Contain Private Keys
© 2017 Carnegie Mellon University

Approved for public release and unlimited distribution.

228

TAMPER DETECTION

What to Do:

- **Attackers can download an APK, modify it, re-sign it**
 - The certificate hash would change, so it'd be obvious it wasn't the same developer (UNLESS YOU INCLUDE YOUR PRIVATE KEY)
 - But the attacker could still re-upload the app as a clone and fool people into downloading it
- **You can add signature checks to your code...but you'd have to be sneaky.**
 - Determined attackers could just figure out where you're checking for the signature and remove it.
- Avoid client-side checks
- Google's SafetyNet has some nifty tamper-detection features
 - Can detect whether a device is rooted (with some level of certainty)
 - Can determine whether a device has malware (to an extent)
- Android P's "Keystore Attestation API": signed statement from secure hardware that the device hasn't been tampered with
- You can run system calls to check whether your application is being accessed by the Android Debug Bridge or whether the app is running on an emulator



@CHMODXX_



@CHMODXX

blog.chmodxx.net

BE PARANOID



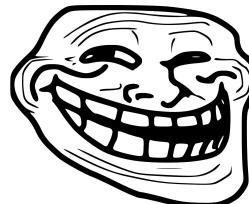
BE PARANOID

ALL THE TIME



BE PARANOID

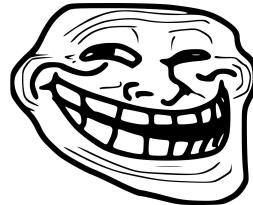
ALL THE TIME



JK.

BE PARANOID

ALL THE TIME



JK. Sort of.

BASIC RULES

- ✓ Never trust the client
- ✓ Follow the Principle of Least Privilege
- ✓ Turn on the security linter in Android Studio
- ✓ NEVER STORE YOUR PRIVATE KEY IN THE APP
- ✓ Be as explicit as possible about your app's intentions
- ✓ **Seriously, never trust the client**



How to Turn on Android Security Linter
File > Settings > Editor > Inspections
<http://tools.android.com/tips/lint-checks>

> CONTINUED LEARNING



RESOURCES



Android Application Security Overview

<https://source.android.com/security/overview/app-security>

Android Developers: App Security Best Practices

<https://developer.android.com/topic/security/best-practices>

OWASP Mobile Security Testing Guide (MSTG)

<https://github.com/OWASP/owasp-mstg>

Full DIVA Solution

<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-apps-diva-part-1/#gref>

Android REing Series

<https://www.youtube.com/c/chmodxx>

BOOKS VIDEOS & COURSES

Stanford | Continuing Studies





Thank you!