ANDROID VULNERABILITIES

Tools for this Workshop

- GenyMotion
- Drozer
- Reverse Engineering
  - APKTool
  - JADX

What you will Gain!

- Insecure Logging
- Hardcoding Issues
- Insecure Data Storage
- Input Validation
- Access Control
- Have some Fun!

Android Debugging Bridge (ADB)

An awesome tool!

ADB

- Tool used to debug your Android Apps
- Java - System.out.println("**{Print Value}**");
- Android - Log.d("**{Key}**", "**{Print Value}**");

ADB Basic Commands

- adb devices - List all Devices
- adb push **{local} {android}** - Put file onto device
- adb pull **{android} {local}** - Take file from device
- adb install **{file.apk}** - Install an Application
- adb uninstall **{package name}**
- adb shell **{linux command}** - Run a Linux shell
- adb logcat - View device log

Challenge 1

Insecure Logging

Hints!

adb logcat - View Android Log

grep **{text}** - Search Line that matches text

Look for what you type

Insecure Logging

- adb shell ps | grep -E 'NAME|diva'

```
USER      PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
u0_a59    1667  244   586464 43508 ffffffff b766307b S jakhar.aseem.diva
```

- adb shell logcat | grep **{pid}**

- Log stays even when you exit the app

- Use an **Obfuscator**

  - Hang on! Will cover this in the next challenge

  - http://stackoverflow.com/questions/5553146/disable-logcat-output-completely-in-release-android-app

Reverse Engineering

Sounds badass huh? It isn’t that tough

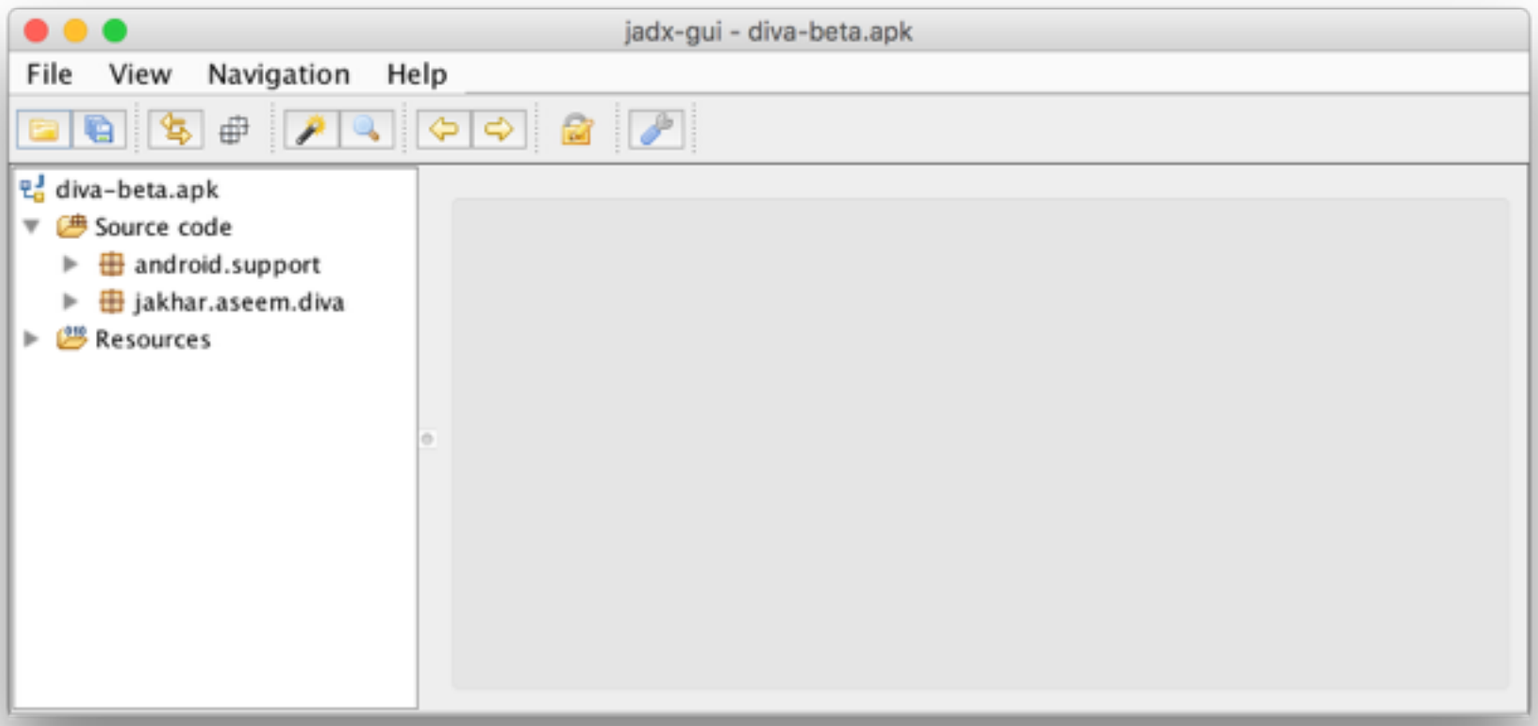
Tools

- APK Tool - Decompile (smali) & Recompile
  - APK Studio - Recompilation
- **JADX** - Provides Deobfuscation
- Dex2Jar & JD-GUI - Command Line Based
  - d2j-dex2jar.sh **{app.apk}**
  - Open Jar with jd-gui.jar

Set up JADX

- Go to **Tools/Reverse Engineering/jadx-0.6.0/bin/jadx-gui**

- Select **diva-beta.apk** in APK folder

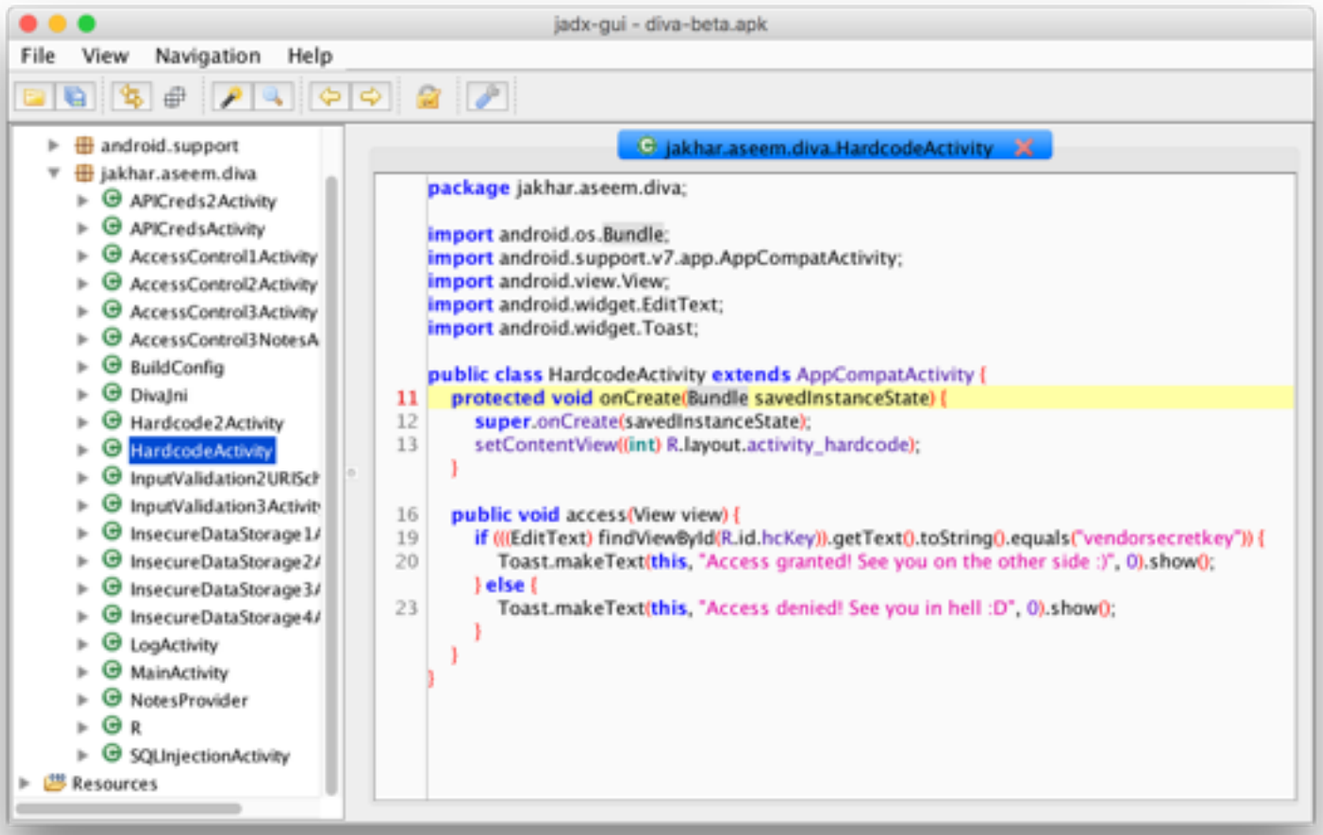


Challenge 2

Hardcoding Issues

Hardcoding Issues

- **Never** store Sensitive Strings in Java
- Use an Obfuscator to slow Reverse Engineering



Obfuscators

- ProGuard - http://proguard.sourceforge.net/
- yGuard - http://www.yworks.com/products/yguard
- DexGuard - https://www.guardsquare.com/dexguard
    - String Encryption

http://proguard.sourceforge.net/index.html#alternatives.html

What it does

Original Source Code Before Obfuscation

```
private void CalcPayroll (SpecialList employeeGroup) {
  while(employeeGroup.HasMore()) {
    employee = employeeGroup.GetNext(true);
    employee.UpdateSalary();
    DistributeCheck(employee);
  }
}
```

Before

Recompiling APKs

- adb uninstall jakhar.aseem.diva
- apktool d diva-beta.apk
- cd diva-beta/smali/jakhar/aseem/diva
- Edit **HardcodeActivity.smali** using a text editor
- apktool b diva-beta
- cd diva-beta/dist
- keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -validity 10000
- jarsigner -verbose -keystore my-release-key.keystore diva-beta.apk alias_name
- jarsigner -verify diva-beta.apk

Storage on Android

How it saves your data

Data Storage

- Shared Preferences
    - Store private primitive data in key-value pairs
- SQLite Databases
    - Store structured data in a private database
- Internal Storage
    - Store private data on the device memory
- External Storage
    - Store public data on the shared external storage

Android Data Folders

- /data/app - APK Files
- /data/data - Application Data Directory
- /data/system - System Data Directory

http://freeandroidforensics.blogspot.sg/2014/11/some-artifacts-in-datasystem-directory.html

Challenge 3,4,5,6

Insecure Data Storage
I’ve lost count of the number of challenges here

Hint: check **/data/data**

Shared Preferences Challenge 3

- adb shell
- cd /data/data/jakhar.aseem.diva/shared_prefs
- cat jakhar.aseem.diva_preferences.xml

```
Editor spedit =
PreferenceManager.getDefaultSharedPreferences(this).edit();
EditText pwd = (EditText) findViewById(R.id.ids1Pwd);
spedit.putString("user", ((EditText)
findViewById(R.id.ids1Usr)).getText().toString());
spedit.putString("password", pwd.getText().toString());
spedit.commit();
```

SQLite
Challenge 4

- cd /data/data/jakhar.aseem.diva/databases
- sqlite3 ids2
- **sqlite>** .tables
- **sqlite>** select * from myuser;
- **sqlite>** .quit

```
this.mDB = openOrCreateDatabase("ids2", 0, null);
this.mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password
VARCHAR);");
…
this.mDB.execSQL("INSERT INTO myuser VALUES ('" + ((EditText)
findViewById(R.id.ids2Usr)).getText().toString() + "', '" + ((EditText)
findViewById(R.id.ids2Pwd)).getText().toString() + "');");
this.mDB.close();
```

Internal Storage Challenge 5

- cd /data/data/jakhar.aseem.diva/
- cat uinfo-*tmp

```
File uinfo = File.createTempFile("uinfo", "tmp", new
File(getApplicationInfo().dataDir));
FileWriter fw = new FileWriter(uinfo);
fw.write(usr.getText().toString() + ":" +
pwd.getText().toString() + "\n");
fw.close();
```

External Storage Challenge 6

- cd /mnt/sdcard
- cat .uinfo.txt

```
File uinfo = File.createTempFile("uinfo", "tmp", new
File(getApplicationInfo().dataDir));
uinfo.setReadable(true);
uinfo.setWritable(true);
FileWriter fw = new FileWriter(uinfo);
fw.write(usr.getText().toString() + ":" +
pwd.getText().toString() + "\n");
fw.close();
```

Lessons Learnt

- **NEVER** store sensitive information on a phone
- Encrypt the data
  - Shared Preferences - https://github.com/scottyab/secure-preferences
  - SQLite - https://github.com/sqlcipher/android-database-sqlcipher
- **Obfuscate** your Code!

Data Inputs

Don’t take what people say as what it is

Data Inputs

- SQL Injection
  - Attacker injects own SQL statements
    - Web, Mobile & Any Application that uses a Database.
- File Traversal
  - file://**{Directory}**
  - Try it on Chrome

Challenge 7, 8

Input Validation

SQL Injection Challenge 7

- No Santization on Input was used.

```
Cursor cr = this.mDB.rawQuery("SELECT * FROM sqliuser WHERE
user = '" + srchtxt.getText().toString() + "'", null);
```

SELECT * FROM sqliuser WHERE user = '**1' or '1' ='1' --**'
SELECT * FROM sqliuser WHERE user = '**1' or '1' ='1**'

File Traversal Challenge 8

- Let us read the file we created during the **external storage challenge**

- file:///**mnt/sdcard/.uinfo.txt**

- file:///**data/data/jakhar.aseem.diva/shared_prefs/jakhar.aseem.diva_preferences.xml**

Input Validation

- Always sanitize your inputs
- Use **PreparedStatements**

```
SQLiteDatabase db = dbHelper.getWritableDatabase();
SQLiteStatement stmt = db.compileStatement("SELECT * FROM
Country WHERE code = ?");
stmt.bindString(1, "US");
stmt.execute();
```

Access Control

Let me get into what you don’t expose me to

Types of Intents

- Every screen, process or message are called intents in Android

- **Activity** - Single screen in an app

- **Service** - Component that performs background operations without a user interface

- **Broadcast** - Message that any app can receive

Intents

- **Explicit Intents** - launch a specific app component, such as a particular activity or service in your app

```
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

- **Implicit Intents** - Any app on device can perform the action

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

Drozer

- Comprehensive security audit and attack framework for Android

- Exposes Activities that are not well protected

- Metasploit for Android

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

https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-drozer-user-guide-2015-03-23.pdf

Let’s set it up

- adb install agent.apk
- Open drozer app and select **ON**
- adb forward tcp:31415 tcp:31415
- drozer console connect

Let’s Roll!

- run app.package.info -a jakhar.aseem.diva
- run app.package.attacksurface jakhar.aseem.diva
- run app.activity.info -a jakhar.aseem.diva
- run app.activity.start --component jakhar.aseem.diva jakhar.aseem.diva.MainActivity
- help app.activity.start
- run app.provider.info -a jakhar.aseem.diva

Challenge 9,10

Access Control

Challenge 9

- adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS

- run app.activity.start --component jakhar.aseem.diva jakhar.aseem.diva.APICredsActivity

Challenge 10

- Never assume that previous Intent always passes a secure value.

- Ensure a session key is set up for sensitive data

Doesn't Work

```
am start -a jakhar.aseem.diva.action.VIEW_CREDS2 -n
jakhar.aseem.diva/.APICreds2Activity -—ez check_pin false
```

Works

```
run app.activity.info -a jakhar.aseem.diva

run app.activity.start --component jakhar.aseem.diva
jakhar.aseem.diva.APICreds2Activity --extra boolean check_pin false
```

Content Providers

- Sharing data between applications through the single ContentResolver interface.

- For example, the contacts data is used by multiple applications

```
getContentResolver().query(NotesProvider.CONTENT_URI, new
String[]{"_id", "title", "note"}, null, null, null)
```

https://developer.android.com/reference/android/content/ContentProvider.html

Challenge 11

Content Providers

Challenge 11

- ADB
  - content query --uri content://jakhar.aseem.diva.provider.notesprovider/notes
- Drozer
  - run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/
  - run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/ --selection “_id=5”
- **Never** set Exported to **True** unless required.

http://www.androidpentesting.com/2015/04/all-you-need-to-know-about-sql.html

Optional Challenges

Are you up for the **Challenge?**

Challenge 12

JNI Hardcode Vulnerability

Use **objdump** & **readelf**

JNI Hardcode Vulnerability

- Java Native Interface (JNI) interact with native code (written in C/C++)
- adb pull /data/data/jakhar.aseem.diva/lib/libdivajni.so libdivajni.so
- objdump -s -j .rodata *.so
    - s - show the full content
    - j - specifying the segment name
- readelf -x .rodata *.so
    - x - specifying the section name

https://github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c

Challenge 13

JNI Input Validation

Use **adb logcat**

JNI Input Validation

- Buffer Overflow

```
const char * pcode = (*env)->GetStringUTFChars(env, jcode, 0);

int ret = 0;
char code[CODESIZEMAX];

strcpy(code, pcode);
```

https://github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c

Real Application Vulnerability

Here’s where the knowledge is put to good use!

Blackboard

- Install it into the emulator and login
- Find your username and password in the console

Hints

Burpsuite to monitor login traffic

Google **Android AccountManager**

Solution

- adb shell
- cd /data/system/users/0
- sqlite3 accounts.db
- select * from accounts;

Android Security

- **Handling Credentials**

  - In general, we recommend minimizing the frequency of asking for user credentials—to make phishing attacks more conspicuous, and less likely to be successful. Instead use an authorization token and refresh it.

  - Where possible, **username and password should not be stored on the device.** Instead, **perform initial authentication** using the username and password supplied by the user, and then **use a short-lived, service-specific authorization token**.

  - Services that will be accessible to multiple applications should be accessed using AccountManager. If possible, use the **AccountManager** class to invoke a **cloud-based service** and **do not store passwords on the device**.

  - https://developer.android.com/training/articles/security-tips.html#WebView

We’ve Conquered it!

There are tons of Tools!

- Covering today's topic and much more!
- https://github.com/tjunxiang92/Android-Vulnerabilities
- https://santoku-linux.com/
- https://developer.android.com/training/best-security.html