Innopolis University

Secure System and Network Engineering

Advanced Security

Laboratory 7 report:

Android

Sergey Grebennikov, Ilnar Sibgatullin

Innopolis, 20 February 2018

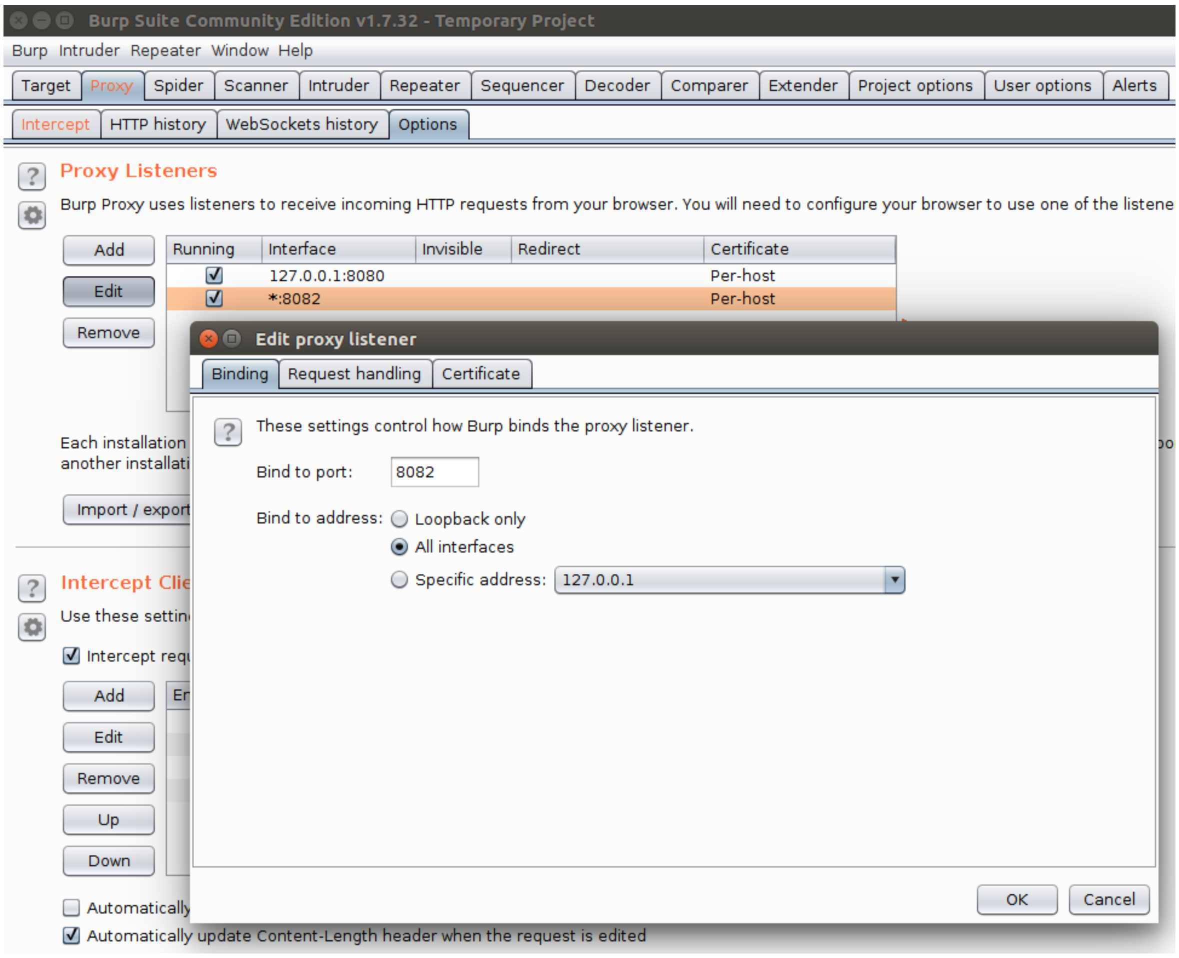
Introduction

The goal of this lab is to take a deeper look to an `Android` application performing its static and dynamic analysis.

1 Preparation

Android application package `weather.apk` was downloaded to a mobile phone with `Android 4.1.2`. To install it "Unknown sources" in the Security settings of the device was enabled.

`Burp Suite Community Edition v1.7.32` was installed and `Burp Proxy Listener` was configured to listen on all interfaces using port 8082 (Figure 1).


Burp Suite Community Edition v1.7.32 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Add
Edit
Remove
Import / export
Intercept Clie
Add
Edit
Remove
Up
Down
Edit proxy listener
Binding Request handling Certificate
OK
Cancel

Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listene

Running	Interface	Invisible	Redirect	Certificate
✓	127.0.0.1:8080			Per-host
✓	*:8082			Per-host

Each installation
another installati

Use these settin
✓ Intercept requ

Automatically
✓ Automatically update Content-Length header when the request is edited

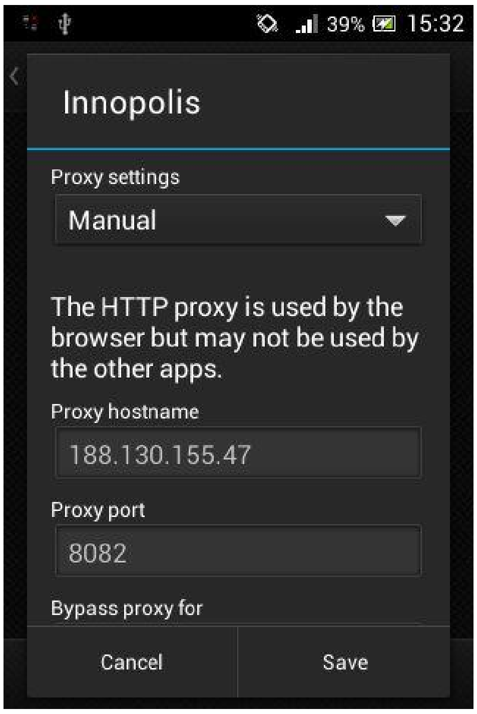
These settings control how Burp binds the proxy listener.

Bind to port: 8082

Bind to address: Loopback only [✓ All interfaces] Specific address: 127.0.0.1

Figure 1: `Burp Proxy Listener` configuration

`Innopolis` Wi-Fi network was modified to use proxy as show in the Figure 2.



Innopolis

Proxy settings

Manual

The HTTP proxy is used by the browser but may not be used by the other apps.

Proxy hostname

188.130.155.47

Proxy port

8082

Bypass proxy for

Figure 2: Network proxy configuration on `Android` device

2 Questions

2.1 Set-up of MITM proxy

Provide proof/commands of how this is done and traffic being MITM'ed. Is it possible to obtain some useful information from what is captured?

Using the Burp proxy server, we managed to intercept the GET and POST requests that were sent by the phone's web browser (Figure 3). This allowed us to get some valuable information (Figure 4).

2.2 Static analysis

2.2.1 Describe the framework

Which framework is used for this Android application? How does this work?

Apache Cordova (formerly PhoneGap) mobile application development framework was used for this Android application.

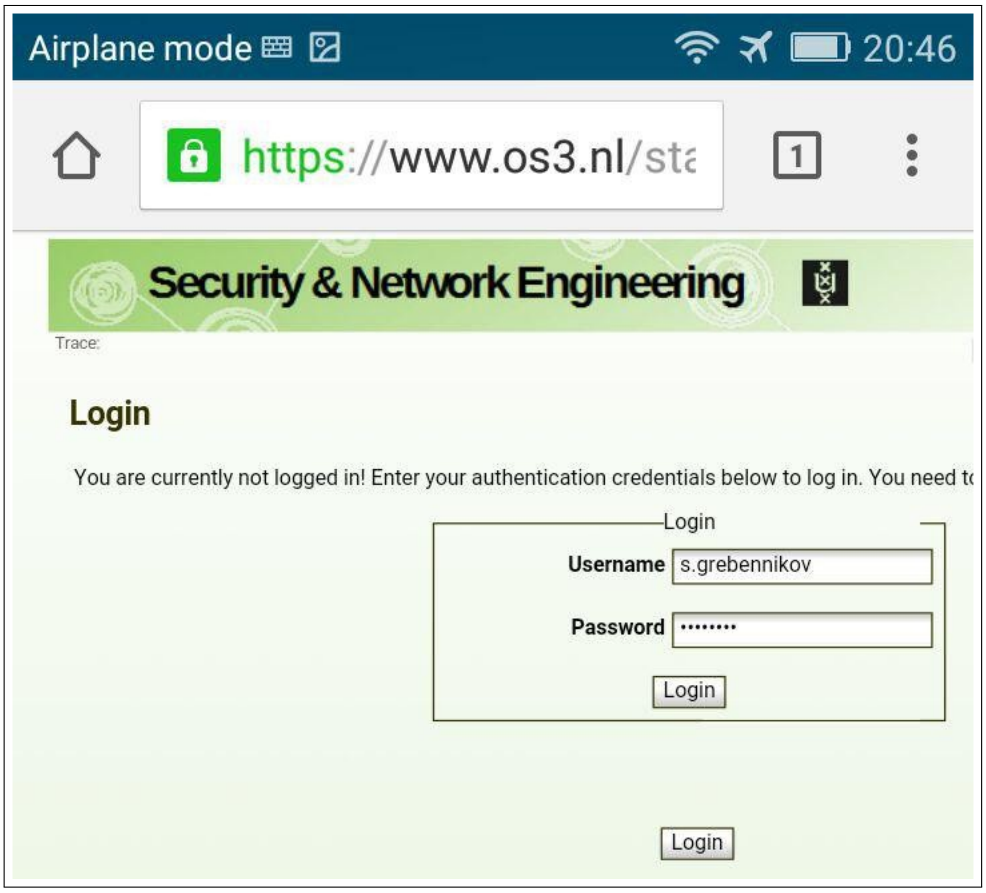
The core of Apache Cordova applications use CSS3 and HTML5 for their rendering and JavaScript for their logic. HTML5 provides access to underlying hardware such as the accelerometer, camera, and GPS.

Apache Cordova can be extended with native plug-ins, allowing developers to add more functionalities that can be called from JavaScript, making it communicate directly between the native layer and the HTML5 page. These plugins allow access to the device's accelerometer, camera, compass, file system, microphone, and more.

2.2.2 Unzip the .apk

APK files are a type of archive file, specifically in zip format packages based on the JAR file format. `weather.apk` was unzipped using the following command

```
$ unzip weather.apk
```


Airplane mode 20:46

https://www.os3.nl/sta

Security & Network Engineering

Trace:

Login

You are currently not logged in! Enter your authentication credentials below to log in. You need t

Login

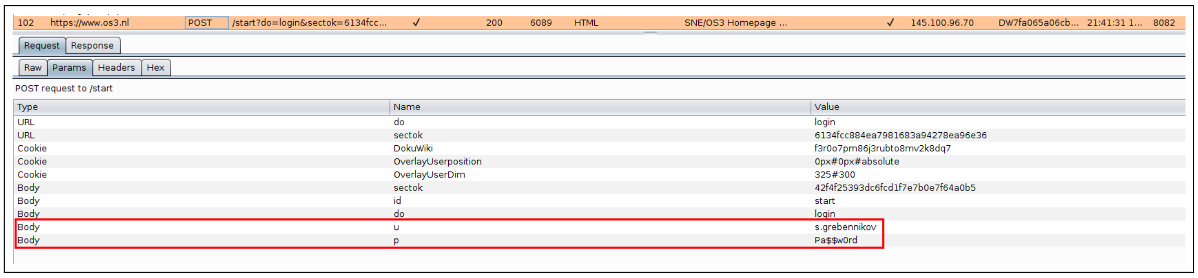
Username s.grebennikov

Password ••••••••

Login

Login

Figure 3: An attempt to log in to the *www.os3.nl* site.


```
102  https://www.os3.nl  POST  /start?do=login&sectok=6134fcc...  ✓  200  6089  HTML  SNE/OS3 Homepage ...  ✓  145.100.96.70  DW7fa065a06cb...  21:41:31 1...  8082
```

Request | Response

Raw | Params | Headers | Hex

POST request to /start

Type	Name	Value
URL	do	login
URL	sectok	6134fcc884ea7981683a94278ea96e36
Cookie	DokuWiki	f3r0o7pm86j3rubto8mv2k8dq7
Cookie	OverlayUserposition	0px#0px#absolute
Cookie	OverlayUserDim	325#300
Body	sectok	42f4f25393dc6fcd1f7e7b0e7f64a0b5
Body	id	start
Body	do	login
Body	u	s.grebennikov
Body	p	Pa$$w0rd

Figure 4: Intercepted login and password

What is the .DEX file? Which files are present?

After unzipping we got the files and folders that are shown in the Figure 5.

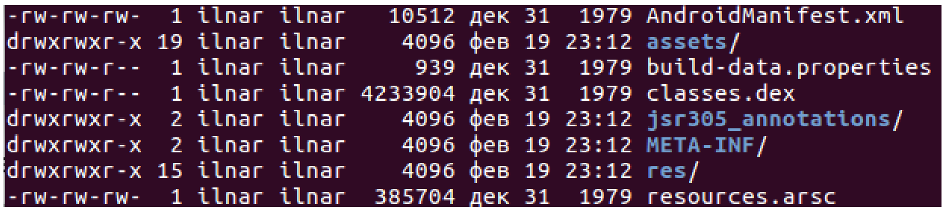

```
-rw-rw-rw-  1 ilnar ilnar   10512 дек 31  1979 AndroidManifest.xml
drwxrwxr-x 19 ilnar ilnar    4096 фев 19 23:12 assets/
-rw-rw-r--  1 ilnar ilnar     939 дек 31  1979 build-data.properties
-rw-rw-r--  1 ilnar ilnar 4233904 дек 31  1979 classes.dex
drwxrwxr-x  2 ilnar ilnar    4096 фев 19 23:12 jsr305_annotations/
drwxrwxr-x  2 ilnar ilnar    4096 фев 19 23:12 META-INF/
drwxrwxr-x 15 ilnar ilnar    4096 фев 19 23:12 res/
-rw-rw-rw-  1 ilnar ilnar  385704 дек 31  1979 resources.arsc
```

Figure 5: Unzipped `weather.apk`

`classes.dex` file is the classes compiled in the `dex` file format understandable by the Dalvik virtual machine and by the Android Runtime. **.DEX**-file stands for Dalvik Executable format.

In `META-INF` directory the following files are present:

- `MANIFEST.MF` manifest file;

- `CERT.RSA` the certificate of the application;
- `CERT.SF` file with the list of resources and SHA-1 digest of the corresponding lines in the `MANIFEST.MF` file.

Also, there are the `resources.arsc` file containing precompiled resources; the `res` directory containing resources not compiled into `resources.arsc` and `assets` directory containing applications assets.

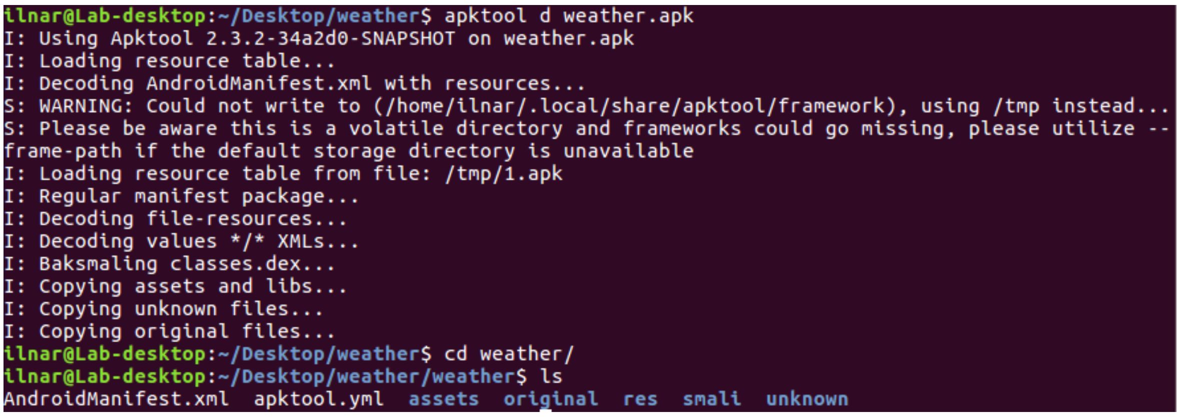
`AndroidManifest.xml` is an additional Android manifest file, describing the name, version, access rights, referenced library files for the application.

2.2.3 Decompile/disassemble with the following tools:

- **With APKtool**
  To decompile with APKtool the following command was used:

  ```
  $ apktool d weather.apk
  ```

  It resulted extracting the files and folder that are shown in the Figure 6.



```
ilnar@Lab-desktop:~/Desktop/weather$ apktool d weather.apk
I: Using Apktool 2.3.2-34a2d0-SNAPSHOT on weather.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (/home/ilnar/.local/share/apktool/framework), using /tmp instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --
frame-path if the default storage directory is unavailable
I: Loading resource table from file: /tmp/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
ilnar@Lab-desktop:~/Desktop/weather$ cd weather/
ilnar@Lab-desktop:~/Desktop/weather/weather$ ls
AndroidManifest.xml  apktool.yml  assets  original  res  smali  unknown
```

Figure 6: Decoding with `apktool`

- **With enjarify**
  `enjarify` is a tool for translating Dalvik bytecode to equivalent Java bytecode. This allows Java analysis tools to analyze Android applications. The following command was used:

  ```
  $ python3 -O -m enjarify.main ../weather.apk
  ```

  The result of command:

  ```
  1000 classes processed
  2000 classes processed
  Output written to weather-enjarify.jar
  ```

What is the relation between DEX/smali/Java?

DEX is a binary bytecode. This is the format that the platform actually understands. However, it's not easy to read or modify binary code, so there are tools out there to convert to and from a human readable representation. The most common human readable format is known as **Smali**. This is essentially the same as the disasssembler. **Java** is a high-level programming language which is used by programmers to write a code.

2.2.4 Analyze the AndroidManifest

The content of `AndroidManifest` file is shown in the Appendix A.

What is the minimum supported Android version and API level? Which permissions are used? What do they mean?

The information about minimum supported Android version and API level can be found in `apktool.yml` file.

```
!!brut.androlib.meta.MetaInfo
apkFileName: weather.apk
compressionType: false
doNotCompress:
- arsc
- txt
isFrameworkApk: false
packageInfo:
  forcedPackageId: '127'
  renameManifestPackage: null
sdkInfo:
  minSdkVersion: '15'
  targetSdkVersion: '16'
sharedLibrary: false
unknownFiles:
  build-data.properties: '8'
  jsr305_annotations/Jsr305_annotations.gwt.xml: '8'
usesFramework:
  ids:
  - 1
  tag: null
version: 2.3.2-34a2d0-SNAPSHOT
versionInfo:
  versionCode: '1'
  versionName: '1.100'
```

Thus, minimum supported API level is 15 (`minSdkVersion:'15'`) which corresponds to `Android 4.0`.

The following permissions are used:

```
android.permission.CAMERA
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
android.permission.READ_PHONE_STATE
android.permission.INTERNET
android.permission.RECEIVE_SMS
android.permission.RECORD_AUDIO
android.permission.MODIFY_AUDIO_SETTINGS
android.permission.VIBRATE
android.permission.READ_CONTACTS
android.permission.WRITE_CONTACTS
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.ACCESS_NETWORK_STATE
android.permission.GET_ACCOUNTS
com.android.browser.permission.READ_HISTORY_BOOKMARKS
android.permission.WAKE_LOCK
android.permission.RECORD_VIDEO
android.permission.READ_EXTERNAL_STORAGE
android.permission.FLASHLIGHT
```

These permissions are Android permissions; they grant access to device features. To maintain security for the system and users, Android requires apps to request permission before the apps can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request.

What is the package name?

The package name is `io.appery.project464000`.

```
package="io.appery.project464000"
```

What is the meaning of the `debuggable` flag here? What is the security impact of using this flag?

The flag `android:debuggable="true"` is located in the manifest file. It means that android will manage all logs file regarding the application.

Shipping application with debug means that anyone with physical access to the device can execute arbitrary code under that application's permission. If the application holds sensitive data, it will be fairly straightforward to extract that sensitive data from the application. Doing the same on nondebuggable application would require the attacker to first obtain root privilege or find an exploit in the application itself.

The `allowBackup` flag is missing here. By setting this flag to `true` (or omitting it), what is the security impact?

The `allowBackup` flag actually is not missing in our `AndroidManifest` file and set to "true".

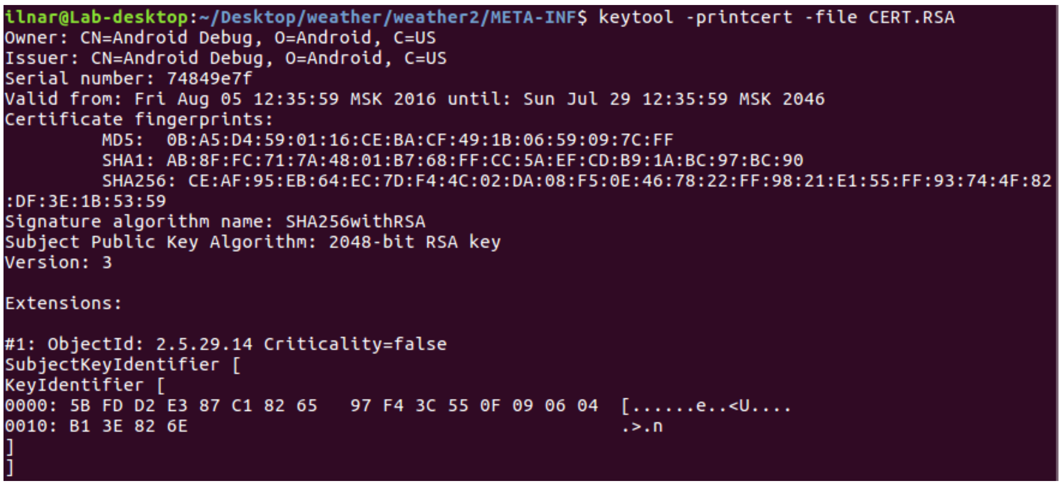
It sets up the automatic backuping of the application data. By default Google Drive could provider is used to perform backup and restore of an application data. Also, It is possible to set a custom cloud backup.

Android backups rely on the Android Debug Bridge (ADB) command to perform backup and restore. Alongside with enabled `debugging` it carries potential security flaws. It gives an attacker opportunity to inject malicious code into your backup data.

2.2.5 Who was the APK signed by?

Android requires that all APKs be digitally signed with a certificate before they can be installed. Certificate information is stored in `CERT.RSA` file. To retrieve it the following command was run.

```
keytool -printcert -file CERT.RSA
```



```
ilnar@Lab-desktop:~/Desktop/weather/weather2/META-INF$ keytool -printcert -file CERT.RSA
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 74849e7f
Valid from: Fri Aug 05 12:35:59 MSK 2016 until: Sun Jul 29 12:35:59 MSK 2046
Certificate fingerprints:
         MD5:  0B:A5:D4:59:01:16:CE:BA:CF:49:1B:06:59:09:7C:FF
         SHA1: AB:8F:FC:71:7A:48:01:B7:68:FF:CC:5A:EF:CD:B9:1A:BC:97:BC:90
         SHA256: CE:AF:95:EB:64:EC:7D:F4:4C:02:DA:08:F5:0E:46:78:22:FF:98:21:E1:55:FF:93:74:4F:82
:DF:3E:1B:53:59
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 5B FD D2 E3 87 C1 82 65   97 F4 3C 55 0F 09 06 04  [......e..<U....
0010: B1 3E 82 6E                                        .>.n
]
]
```

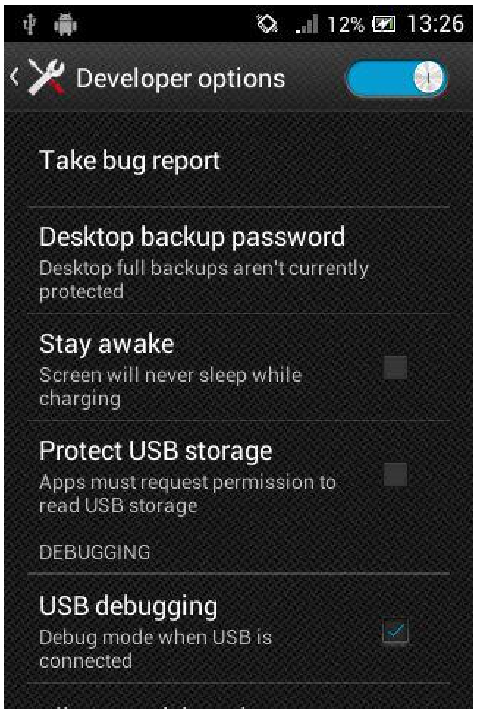
Figure 7: APK certificate information

As we can see in the Figure 7 APK was singed by Android Debug.

2.3 Dynamic analysis

Android Debug Bridge (`adb`) command-line tool was used to perform dynamic analysis. It provides access to a Unix shell that can be used to run a variety of commands on a device.

To use `adb` with a phone connected over USB, **USB debugging** in the device system settings, under **Developer options** was enabled (Figure 8).



12% 13:26

Developer options

Take bug report

Desktop backup password
Desktop full backups aren't currently protected

Stay awake ☐
Screen will never sleep while charging

Protect USB storage ☐
Apps must request permission to read USB storage

DEBUGGING

USB debugging ✓
Debug mode when USB is connected

Figure 8: USB debugging enabled

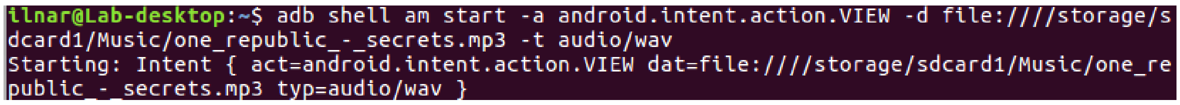
What can you achieve by calling intents? Manually create an intent from the ADB shell to play some music. Use Frida to log system calls (e.g. `write()` calls)

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

To play a music file on the phone the following intent from ADB shell was started.

```
adb shell am start -a android.intent.action.VIEW -d file:////storage/
    sdcard1/Music/one_republic_-_secrets.mp3 -t audio/wav
```

The output of the command is shown in the Figure 9.



```
ilnar@Lab-desktop:~$ adb shell am start -a android.intent.action.VIEW -d file:////storage/s
dcard1/Music/one_republic_-_secrets.mp3 -t audio/wav
Starting: Intent { act=android.intent.action.VIEW dat=file:////storage/sdcard1/Music/one_re
public_-_secrets.mp3 typ=audio/wav }
```

Figure 9: `adb shell` intent to play a music

The specified `mp3` file playback started on the phone (Figure 10).

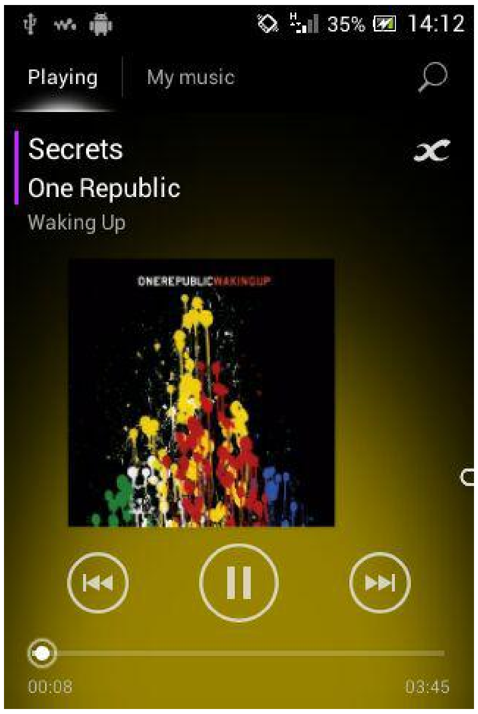


Figure 10: Music file playback

Frida is a dynamic code instrumentation toolkit. It lets you inject snippets of JavaScript or your own library into native apps on Windows, macOS, GNU/Linux, iOS, Android, and QNX.

It was installed using `pip` as follows.

```
# pip install frida
```

Then the latest `frida-server` for Android was download from `https://build.frida.re/frida/android/arm/bin/` and pushed to the phone.

```
$ adb push ~/Downloads/frida-server /data/local/tmp/
$ adb shell "chmod 755 /data/local/tmp/frida-server"
$ adb shell "/data/local/tmp/frida-server &"
```

To start logging `write()` calls the following command was used.

```
frida-trace -U -n io.appery.project464000 - i write
```

Conclusion

Static and dynamic analysis of the Android application `weather.apk` were performed.

Resources

1. `https://support.portswigger.net/`
2. `https://en.wikipedia.org/wiki/Android_application_package`
3. `https://en.wikipedia.org/wiki/Apache_Cordova`
4. `https://developer.android.com/studio/command-line/adb.html`

Appendices

A Appendix

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
 ↪    android:hardwareAccelerated="true" package="io.appery.project464000" platformBuildVersionCode="23"
 ↪    platformBuildVersionName="6.0-2704002">
    <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:resizeable="true"
     ↪    android:smallScreens="true" android:xlargeScreens="true"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-feature android:name="android.hardware.location.gps"/>
    <uses-permission android:name="android.permission.RECORD_VIDEO"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.FLASHLIGHT"/>
    <uses-feature android:name="android.hardware.camera" android:required="true"/>
    <application android:allowBackup="true" android:debuggable="true" android:hardwareAccelerated="true" android:icon="@drawable/icon"
     ↪    android:label="@string/app_name" android:supportsRtl="true" android:theme="@android:style/Theme.Holo.Light.NoActionBar">
        <activity android:configChanges="keyboardHidden|orientation|screenSize" android:label="@string/activity_name"
         ↪    android:launchMode="singleTop" android:name="io.appery.project464000.MainActivity"
         ↪    android:theme="@android:style/Theme.DeviceDefault.NoActionBar" android:windowSoftInputMode="adjustResize">
            <intent-filter android:label="@string/launcher_name">
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <receiver android:enabled="true" android:name="com.google.android.gms.analytics.AnalyticsReceiver">
            <intent-filter>
                <action android:name="com.google.android.gms.analytics.ANALYTICS_DISPATCH"/>
            </intent-filter>
        </receiver>
        <service android:enabled="true" android:exported="false" android:name="com.google.android.gms.analytics.AnalyticsService"/>
        <receiver android:enabled="true" android:exported="true"
         ↪    android:name="com.google.android.gms.analytics.CampaignTrackingReceiver">
            <intent-filter>
                <action android:name="com.android.vending.INSTALL_REFERRER"/>
            </intent-filter>
        </receiver>
        <service android:enabled="true" android:exported="false"
         ↪    android:name="com.google.android.gms.analytics.CampaignTrackingService"/>
        <provider android:authorities="io.appery.project464000.provider" android:exported="false" android:grantUriPermissions="true"
         ↪    android:name="android.support.v4.content.FileProvider">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/provider_paths"/>
        </provider>
        <activity android:clearTaskOnLaunch="true" android:configChanges="keyboardHidden|orientation|screenSize"
         ↪    android:exported="false" android:name="com.google.zxing.client.android.CaptureActivity"
         ↪    android:theme="@android:style/Theme.NoTitleBar.Fullscreen" android:windowSoftInputMode="stateAlwaysHidden">
            <intent-filter>
                <action android:name="com.google.zxing.client.android.SCAN"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity android:label="Share" android:name="com.google.zxing.client.android.encode.EncodeActivity">
            <intent-filter>
                <action android:name="com.phonegap.plugins.barcodescanner.ENCODE"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity android:label="Share" android:name="com.google.zxing.client.android.HelpActivity">
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity android:exported="false" android:name="com.google.android.gms.common.api.GoogleApiActivity"
         ↪    android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
        <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>
    </application>
</manifest>
```