Mobile Hacking Android
AGENDA

- Einleitung
  - Ziele
  - Einführung Terminologie

- Schwachstellen

- Tools
ACTIVITIES

- activity represents a single screen with a user interface
  - email app might have one activity that shows a list of new emails
  - another activity to compose an email,
  - and another activity for reading emails

- each one is independent of the others

- different app can start any one of these activities (if the email app allows it)

- camera app can start the activity in the email app that composes new mail, in order for the user to share a picture
SERVICES

- *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes

- does not provide a user interface
  - service might play music in the background while the user is in a different app
  - might fetch data over the network without blocking user interaction with an activity
  - another component, such as an activity, can start the service and let it run or bind to it in order to interact with it
EINFÜHRUNG - TERMINOLOGIE

CONTENT PROVIDERS

- *content provider* manages a shared set of app data
- store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access
- through the content provider, other apps can query or even modify the *data* (if the content provider allows it)
  - Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query part of the content provider (such as `ContactsContract.Data`) to read and write information about a particular person
BROADCAST RECEIVERS

- **broadcast receiver** is a component that responds to system-wide broadcast announcements
  - broadcast announcing that the screen has turned off, the battery is low, or a picture was captured
  - let other apps know that some data has been downloaded to the device and is available for them to use
  - although broadcast receivers don't display a user interface
  - More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work
  - broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object
EINFÜHRUNG - TERMINOLOGIE

INTENTS

- *activities, services, and broadcast receivers*—are activated by an asynchronous message called an intent

- Intents bind individual components to each other at runtime

- An intent is created with an Intent object, which defines a message to activate either a specific component or a specific type of component—an intent can be either explicit or implicit, respectively

- For activities and services, an intent defines the action to perform
  - for example, to "view" or "send" something

- may specify the URI of the data to act on
  - among other things that the component being started might need to know
ZUSAMMENFASSUNG

- Activity:
  - different app can start any one of these activities (if the email app allows it)

- Service:
  - another component, such as an activity, can start the service and let it run or bind to it in order to interact with it

- Content providers:
  - through the content provider, other apps can query or even modify the data (if the content provider allows it)

- Broadcast receivers:
  - More commonly, though, a broadcast receiver is just a "gateway" to other components

Quelle: https://developer.android.com/guide/components/fundamentals.html
EINFÜHRUNG - TERMINOLOGIE

ZUSAMMENFASSUNG

- **Activity:**
  - different app can start any one of these activities (if the email app allows it)

- **Service:**
  - another component, such as an activity, can start the service and let it run or bind to it in order to interact with it

- **Content providers:**
  - through the content provider, other apps can query or even modify the data (if the content provider allows)

- **Broadcast receivers:**
  - More commonly, though, a broadcast receiver is just a “gateway” to other components

Quelle: https://developer.android.com/guide/components/fundamentals.html
EINFÜHRUNG - TERMINOLOGIE

ANDROID MANIFEST

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly.

```xml
<activity>
    <meta-data android:name="android:name" android:value=".ViewStatement" />
    <meta-data android:name="android:label" android:value="@string/title_activity_view_statement" />
</activity>

<provider
    android:name=".TrackUserContentProvider"
    android:authorities="com.android.insecurebank2.TrackUserContentProvider"
    android:exported="true"
>
</provider>

<receiver
    android:name=".MyBroadcastReceiver"
    android:exported="true"
>
    <intent-filter>
        <action android:name="theBroadcast" />
    </intent-filter>
</receiver>
```
DAS SETUP / SCHÄRFE DEINE TOOLS

adb
Android Debug Bridge version 1.0.32
Revision 09a0d98bebce-android
-a
directs adb to
-d
directs command
which returns an error
-e
directs command
which returns an error

adb install InsecureBankv2.apk
adb server is out of date. killing...
daemon started successfully
400 KB/s (3503142 bytes in 8.533s)
pkg: /data/local/tmp/InsecureBankv2.apk
Success

Marko Winkler / Mobile Hacking - Android
15.02.2017
DAS SETUP / SCHÄRFÉ DEINE TOOLS

InsecureBankv2

Marko

...
SCHWACHSTELLEN
ACTIVITY EXPORTED

ACTIVITY

```xml
<activity
    android:name=".PostLogin"
    android:exported="true"
    android:label="@string/title_activity_post_login">
    ...
</activity>

> am start -n com.android.insecurebankv2/.PostLogin
```
Activity Exported
ACTIVITY EXPORTED

ACTIVITY

Transfer

View Statement

Change Password

Device not Rooted!!
<receiver
    android:name=".MyBroadcastReceiver"
    android:exported="true">
    <intent-filter>
        <action android:name="theBroadcast" />
    </intent-filter>
</receiver>

public class MyBroadcastReceiver extends BroadcastReceiver {
    String usernameBase64ByteString;
    public static final String MYPREFS = "mySharedPreferences";

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub
        String phn = intent.getStringExtra("phoneNumber");
        String newpass = intent.getStringExtra("newpass");

        if (phn != null) {
            try {
                SharedPreferences settings = context.getSharedPreferences(MYPREFS, Context.MODE_WORLD_READABLE);
                String username = settings.getString("EncryptedUsername", null);
                byte[] usernameBase64Byte = Base64.decode(username, Base64.DEFAULT);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }
}
public class MyBroadcastReceiver extends BroadcastReceiver {
    String usernameBase64ByteString;
    public static final String MYPREFS = "mySharedPreferences";

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub
        String phn = intent.getStringExtra("phonenumber");
        String newpass = intent.getStringExtra("newpass");
        if (phn != null) {
            try {
                SharedPreferences settings = context.getSharedPreferences(MYPREFS, Context.MODE_WORLD_READABLE);
                final String username = settings.getString("EncryptedUsername", null);
                byte[] usernameBase64Byte = Base64.decode(username, Base64.DEFAULT);
                // Further processing...

            } catch (Exception e) {
                // Handle exception
            }
        }
    }
}
public class TrackUserContentProvider extends ContentProvider {

  // This content provider vuln is a modified code from www.androidpentesting.com

  static final String PROVIDER_NAME = "com.android.insecurebankv2.TrackUserContentProvider";
  // The Content provider that handles all the tracked user history
  static final String URL = "content://" + PROVIDER_NAME + "/trackerusers";
  static final Uri CONTENT_URI = Uri.parse(URL);
  static final String name = "name";
  static final int uriCode = 1;
  static final UriMatcher uriMatcher;
  private static HashMap < String, String > values;
  private SQLiteDatabase db;
  static final String DATABASE_NAME = "myDb";
  static final String TABLE_NAME = "names";
  static final int DATABASE_VERSION = 1;
  static final String CREATE_DB_TABLE = " CREATE TABLE " + TABLE_NAME + " (id INTEGER PRIMARY KEY AUTOINCREMENT, " + " name TEXT NOT NULL);";
public class TrackUserContentProvider {

    // This content provider is used to store and retrieve user data.
    static final String CONTENT_AUTHORITY = "com.example.trackusers.contentprovider";
    static final String CONTENT_BASE_URI = "content://com.example.trackusers/
    static final String CONTENT_ITEM_BASE_URI = "content://com.example.trackusers/
    static final String CONTENT_PROVIDER_NAME = "com.example.trackusers"

    // Define constants for the tables and columns
    public static final String TABLE_NAME = "trackusers";
    public static final String COLUMN_ID = "id";
    public static final String COLUMN_NAME = "name";

    // Define content query selection strings
    public static final String CONTENT_QUERY = "SELECT * FROM " + TABLE_NAME + " WHERE " + COLUMN_ID + " = ?";

    // Define the table schema
    public static final String TABLE_SCHEMA = "CREATE TABLE " + TABLE_NAME + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " + COLUMN_NAME + " TEXT NOT NULL)";

    // Constructor
    public TrackUserContentProvider() {
        // Initialize the database
        SQLiteDatabase db = null;
    }

    // Methods for inserting, updating, and deleting data
    public void insert(String name) {
        // Insert user name into the table
    }
    public void update(int id, String name) {
        // Update user name with the new name
    }
    public void delete(int id) {
        // Delete user with the given id
    }

    // Method for getting all user names
    public Cursor getAllUsers() {
        // Return all users from the database
    }
}
INTENTS

```java
package com.android.dns.sniff intents;

import ...

public class MainActivity extends ActionBarActivity {
    Button bypassLogin;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        IntentFilter filter = new IntentFilter("theBroadcast");
        MyReceiver receiver = new MyReceiver();
        registerReceiver(receiver, filter);
        TextView t1 = (TextView) findViewById(R.id.textView);

        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            t1.setText("Phone Number is: " + extras.getString("phone");
        }
    }

    class MyReceiver {
        // Receiver implementation
    }
}
```
INTENTS

Change Password

Phone Number is: 1555215554 and New Password is: Dinesh@123!

Intent Sniffing
SENSITIVE DATA

HARDCODED STRINGS

http://resources.infosecinstitute.com/android-hacking-security-part-9-insecure-local-storage-shared-preferences/

```java
public class MyBroadcastReceiver extends BroadcastReceiver {
    String usernameBase64ByteString;
    public static final String MYPREFS = "mySharedPreferences";

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub

        String phn = intent.getStringExtra("phonenumber");
        String newpass = intent.getStringExtra("newpass");

        if (phn != null) {
            try {
                SharedPreferences settings = context.getSharedPreferences(MYPREFS, Context.MODE_WORLD_READABLE);
                final String username = settings.getString("EncryptedUsername", null);
                byte[] usernameBase64Byte = Base64.decode(username, Base64.DEFAULT);
                usernameBase64ByteString = new String(usernameBase64Byte, "UTF-8");
                final String password = settings.getString("superSecurePassword", null);
                CryptoClass crypt = new CryptoClass();
                String decryptedPassword = crypt.aesDecryptedString(password);
                String textPhoneno = phn.toString();

                String textMessage = "Updated Password from: " +decryptedPassword+" to: ":"+newpass;
                SmsManager smsManager = SmsManager.getDefault();
                System.out.println("For the changepassword - phonenumber: "+textPhoneno+" password is: "+textMessage);
                smsManager.sendTextMessage(textPhoneno, null, textMessage, null, null);
            } catch (Exception exception) {
                // Handle exception
            }
        }
    }
}
```
TOOLS

ANALYSE

- https://ibotpeaches.github.io/Apktool/
  - reverse engineering Android apk files
- https://github.com/skylot/jadx
  - Dex to Java Decompiler
  - Read/write the Dalvik Executable (.dex) file
  - Convert .dex file to .class files
  - disassemble dex to smali files and assemble dex from smali files

SCA

- https://github.com/linkedin/qark
  - QARK is an easy to use tool capable of finding common security vulnerabilities in Android applications
DefCon

- Vortrag *backdoor*ing the *frontdoor*
  - Q: „Wie hast du die iPhone App geknackt?“
  - A: „Ich habe die Android App decompiled...“
QUELLE

HTTPS://GITHUB.COM/DINESHSHETTY/ANDROID-INSECUREBANKV2
HTTPS://DEVELOPER.ANDROID.COM/GUIDE/COMPONENTS/FUNDAMENTALS.HTML