

# **APK TESTING REPORT**

## **Mobile Ecosystem Security**

# APK INFORMATION

**APK Name :-** Assignment.apk

**Package Name :-** com.intrepidusgroup.learner

**Description :-** Assignment.apk has eight challenges which are designed as:-



# TESTING PLATFORM

**Host Machine :-** Microsoft Windows

**Virtual Machine :-** Kali Linux

Assignment.apk has been installed in Android Virtual Environment(GenyMotion)

## TOOLS

**Adb :-** Android Debug Bridge is a tool used to communicate with the phone

**Apk Tool :-** 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.

**MobSF:-** Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.

**Drozer :-** Drozer (formerly Mercury) is the leading security testing framework for Android.

**Burpe Suite :-** Intercept tool by Portswigger

## Assignment.apk Analysis Using MobSF

Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.

MobSF is used to analyse the Java Code of the Application.

```
File Edit View Terminal Tabs Help
MobSFv3.0
[INFO] 23/May/2020 13:26:36 - Mobile Security Framework v3.0.9 Beta
REST API Key: c50b5eeaf2c578f4c4827fa7b535ba680bd4b9ca964da740648745328798274
3
[INFO] 23/May/2020 13:26:36 - OS: Linux
[INFO] 23/May/2020 13:26:36 - Platform: Linux-4.19.0-kali4-amd64-x86_64-with-
glibc2.29
[INFO] 23/May/2020 13:26:36 - Dist: kali 2020.2 kali-rolling
[INFO] 23/May/2020 13:26:36 - MobSF Basic Environment Check
[INFO] 23/May/2020 13:26:36 - Checking for Update.
[INFO] 23/May/2020 13:26:37 - No updates available.
[2020-05-23 10:51:26 -0400] [18653] [CRITICAL] WORKER TIMEOUT (pid:18655)
[2020-05-23 10:51:27 -0400] [18919] [INFO] Booting worker with pid: 18919
[2020-05-23 12:15:47 -0400] [18653] [CRITICAL] WORKER TIMEOUT (pid:18919)
[2020-05-23 12:15:48 -0400] [18970] [INFO] Booting worker with pid: 18970
[INFO] 23/May/2020 16:17:32 -
```

The screenshot displays the MobSF web interface with the following sections:

- RECENT SCANS**: A navigation menu with options for RECENT, STATIC, DYNAMIC, API, and ABOUT. A search bar for MDS is also present.
- APP SCORES**: Shows a hammer icon, an average CVSS score of 7.4, a Security Score of 45/100, and 0/285 Trackers Detection.
- FILE INFORMATION**:
  - File Name: Assignment.apk
  - Size: 0.27MB
  - MDS: 693051abdf0f7034a64527762240af21
  - SHA1: 37f2bc34af48b99fc34d382dd2dcb8c2ac0a01ca
  - SHA256: 5ee56c83641c6ec766e677d34df3b8c1c15c2fc92b96
  - 85de371963be1be7420c
- APP INFORMATION**:
  - App Name: Learner
  - Package Name: com.intrepidusgroup.learner
  - Main Activity: com.intrepidusgroup.learner.LessonSelectorActivity
  - Target SDK: 16, Min SDK: 14, Max SDK: 16
  - Android Version Name: 1.0
  - Android Version Code: 1

## Connect to Phone

To connect to phone and install the apk we will use the following commands as shown below :-

```
unable to connect to 192.168.42.102:5555: connection timed out
root@kali:~/Downloads/platform-tools# adb connect 192.168.42.102
connected to 192.168.42.102:5555
root@kali:~/Downloads/platform-tools# adb devices
List of devices attached
192.168.42.102:5555    device
```

192.168.42.102 = IP of Mobile Phone

```
# adb install Assignment.apk
```

## Lesson 1: Android Logging Secret

Analysed the java code of [com/intrepidusgroup/learner/Lesson1Activity.java](#) and the instruction given in lesson 1 we get to know that create challenge function is followed by fillLogWithGarbage() function which is responsible for creating logs appending with "LEARNER"

```
private void fillLogWithGarbage() {
    int nonSecureRandomInt = new Random().nextInt(100);
    for (int i = 0; i < 100; i++) {
        if (i == nonSecureRandomInt) {
            Log.d("LEARNER",
String.valueOf(getResources().getString(R.string.challengeString)) + this.challenge);
        } else {
            Log.d("LEARNER",
String.valueOf(getResources().getString(R.string.garbageLogString)) +
generateChallenge());
        }
    }
}
```

Analyzing the log with Learner as a tag

```
root@kali:~/Downloads/platform-tools# adb logcat LEARNER
```

```
O/LEARNER ( 2746): This is garbage. The challenge value is:jeot0tuxkx  
O/LEARNER ( 2746): This is garbage. The challenge value is:6beid6g74h  
O/LEARNER ( 2746): This is garbage. The challenge value is:4vlsnalou1  
O/LEARNER ( 2746): This is garbage. The challenge value is:lbcvnb7lw  
O/LEARNER ( 2746): This is garbage. The challenge value is:7mx7n666mc  
O/LEARNER ( 2746): This is not garbage. The challenge value is:p2l03wg4bl  
O/LEARNER ( 2746): This is garbage. The challenge value is:wtujs8q8h3  
O/LEARNER ( 2746): This is garbage. The challenge value is:o4aayljmbe  
O/LEARNER ( 2746): This is garbage. The challenge value is:csidr92kg  
O/LEARNER ( 2746): This is garbage. The challenge value is:in4e7jxa5u  
O/LEARNER ( 2746): This is garbage. The challenge value is:77l6ln0k45
```

p2l03wg4bl

Congrats! The challenge is completed.

## Lesson 2: Screwy File Permission

In java source of file `com/intrepidusgroup/learner/Lesson2Activity.java` the code for generating the file name is visible according to which the file name is current date with phone number and extension as `.txt` But, Here the need is to find the file in the apk package following the hint

```
root@vbox86p:/data/data/com.intrepidusgroup.learner # cd files
root@vbox86p:/data/data/com.intrepidusgroup.learner/files # ls
2020052315555218135.txt
```

Location of file

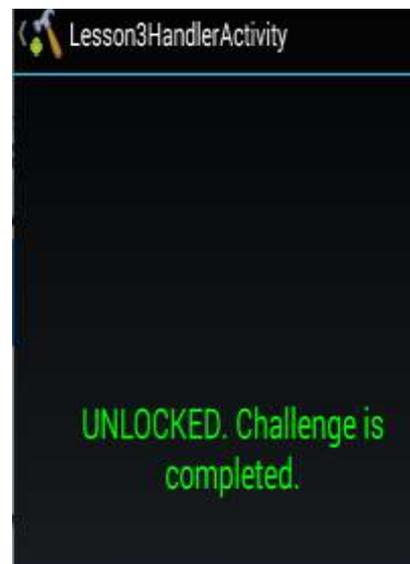


## Lesson 3: URI Handler Crazyiness

Just By analyzing the file Just by com/intrepidusgroup/learner/Lesson3HandlerActivity.java we will get the answer for this challenge

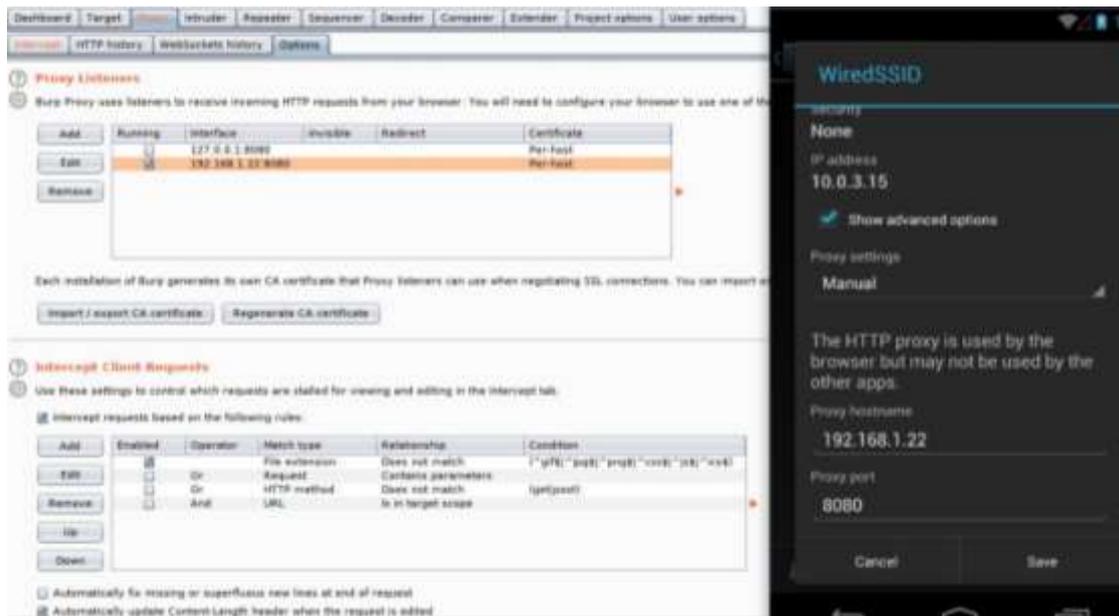
```
public class Lesson3HandlerActivity extends Activity {
    final String unlockkey = "crazyurihandler";

    /* access modifiers changed from: protected */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lesson3_handler);
        getActionBar().setDisplayHomeAsUpEnabled(true);
        Intent uriReqIntent = getIntent();
        String uri = uriReqIntent.getScheme();
        String path = uriReqIntent.getData().getPath();
        Log.d("LEARNER", "Received request for URI " + uri + " at path: " + path);
        TextView lockedStatus = (TextView) findViewById(R.id.textViewLocked);
        if (path.contains("crazyurihandler")) {
```

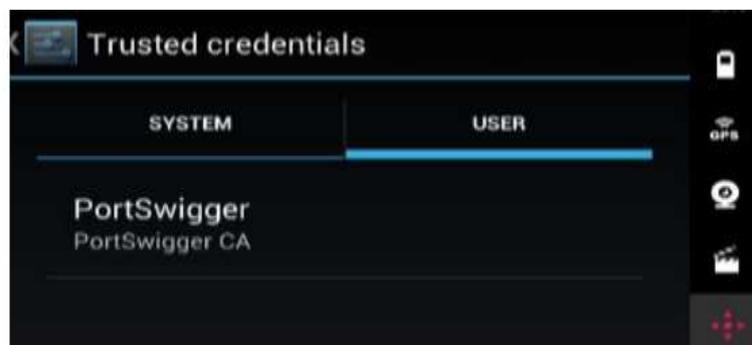


## Lesson 4: SSL Man in the Middle Attack

This challenge requires Burp Suite to be configured with the custom phone. Download the burp certificate from <http://burp> from phone's browser as cert.der



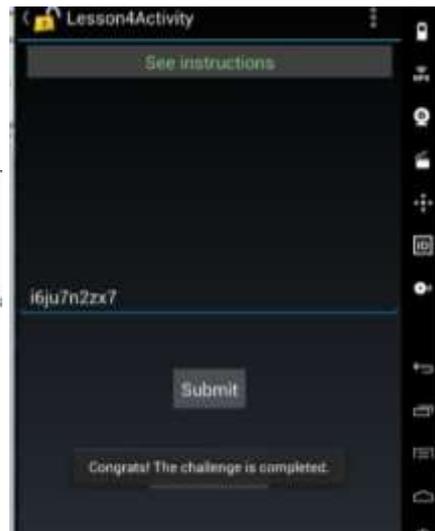
After adding the PortSwigger Certificate as a trusted one in System < Security < Trusted Credentials, Open the Learner application and click on resend code while keeping burp suite intercept as on. The Secret Token Header is the key for this level, i.e. i6ju7n2zx7



```

Request to https://intrepidusgroup.com:443 [69.163.179.108]
Forward Drop Intercept is on Action
Raw Headers Hex
GET / HTTP/1.1
SecretTokenHeader: i6ju7n2zx7
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.1; Custom Build/JR0035)
Host: intrepidusgroup.com
Connection: close
Accept-Encoding: gzip, deflate

```



## Lesson 6: Encryption vs Encraption

Analysing the `com/intrepidusgroup/learner/Lesson6Activity.java` code we get to know the AES Encryption is used with key “intrepidlearner1”

```

private String encryptNumberWithAES(String number) throws
NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
IllegalBlockSizeException, BadPaddingException {
    byte[] numberToEncryptBytes = number.getBytes();
    byte[] key = "intrepidlearner1".getBytes();
    Cipher c = Cipher.getInstance("AES");
    c.init(1, new SecretKeySpec(key, "AES"));
    return Base64.encodeToString(c.doFinal(numberToEncryptBytes), 0);
}

public void lesson6OnSubmitClick(View v) {
    if (checkSecretToken(((TextView)
        findViewById(R.id.lesson6EditText1)).getText().toString())) {

```

We had used online tool to decrypt the phone number that is provided to us.

# Encryption

Enter text to be Encrypted

0611933556

OR

Browse...

No file selected

Select Mode

ECB

Enter Secret Key

intrepidlearner1

Output Text Format:  Base64

Hex

Encrypt

AES Encrypted Output:

KzPMH5t9W4z7qRJwU6wu4w==

Lesson6Activity

See instructions

Challenge phone number:  
0611933556

KzPMH5t9W4z7qRJwU6wu4w==

Congrats! The challenge is completed.

Lesson 7: Shared with the World

Drozer is an open source software, a framework for Android security assessments. “Drozer allows you to assume the role of an Android app, and to interact with other apps, through Android’s Inter-Process Communication (IPC) mechanism, and the underlying operating system.”

```
public class lesson7ContentProvider extends ContentProvider {
    private static final String AUTHORITY = "com.intrepidusgroup.learner.contentprovider";
    private static final String BASE_PATH = "iglearnerdb";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.cursor.item/user";
    public static final String CONTENT_TYPE = "vnd.android.cursor.dir/users";
    public static final Uri CONTENT_URI = Uri.parse("content://com.intrepidusgroup.learner.contentprovider/iglearnerdb");
    private static final int USERS = 10;
    private static final int USER_ID = 20;
    private static final UriMatcher sURIMatcher = new UriMatcher(-1);
    private Lesson7DatabaseHelper database;
}
```

Download the drozer.apk and install in the phone and forward the tcp ports.

```
root@kali:~/Downloads/platform-tools# adb forward tcp:31415 tcp:31415
root@kali:~/Downloads/platform-tools# drozer console connect
```

The screenshot shows a terminal window on the left and an Android application interface on the right. The terminal window displays the following commands and output:

```
dz> run app.provider.query content://com.intrepidusgroup.learner.ctentprovide
r/iglearnerdb
Could not get a ContentProviderClient for content://com.intrepidusgroup.learn
er.ctentprovider/iglearnerdb.
dz> run app.provider.query content://com.intrepidusgroup.learner.contentprovi
der/iglearnerdb
|_id | user      | password                                     | description
|----|-----|-----|-----|
| 1  | johndoe  | you found the password:zod4w3tnox          | John Doe's description
```

The Android application interface on the right shows a screen titled "Lesson7" with the following text:

In this lesson, our application has spinned up a data provider but in an insecure fashion. Your goal is to find the password stored within the database. You'll know when you've found it.

Hint: we really like mercury.

See instructions

## Lesson 8: Malicious intent

Analysing the Manifest.xml file will help in this case .The file includes nodes of each of the files including Activities, Services ,Content Providers and Broadcast Receiver that makes an application and intent filters Permission determines how they coordinate with each of the other Application.

```
/* access modifiers changed from: protected */
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lesson8_aux);
    getActionBar().setDisplayHomeAsUpEnabled(true);
    TextView lockedStatus = (TextView) findViewById(R.id.lesson8TextViewLocked);
    Intent intent = getIntent();
    if
(!intent.getAction().equals(getResources().getString(R.string.lesson8IntentActionString)) || !intent.getExtras().containsKey(getSecret())) {
        lockedStatus.setText(R.string.wrongIntentString);
        lockedStatus.setTextColor(-65536);
        return;
    }
    lockedStatus.setText(R.string.debugString);
    lockedStatus.setTextColor(-16711936);
}
```

In **Lesson8AuxActivity.java** we can see that class Lesson8AuxActivity calls for action with an extra string defined in function getSecret() and the secret is current date.

```
private String getSecret() {
    return getDate();
}

private String getDate() {
    DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
    Date currDate = new Date();
    Log.d("LEARNER", "Current date is " + dateFormat.format(currDate));
    return dateFormat.format(currDate);
}
}
```

Drozer is being used

Following are the steps to call the action “com.intrepidusgroup.learner.custom.intent.action.SEND” :

```
root@kali:~/Downloads/platform-tools# adb forward tcp:31415 tcp:31415
root@kali:~/Downloads/platform-tools# drozer console connect
```

```
For more information on how to formulate an Intent, type 'help intents'.
Last Modified: 2012-11-06
Credit: MWR InfoSecurity (@mwrlabs)
License: BSD (3 clause)

optional arguments:
  -h, --help
  --action ACTION          specify the action to include in the Intent
  --category CATEGORY [CATEGORY ...]
                          specify the category to include in the Intent
  --component PACKAGE COMPONENT
                          specify the component name to include in the Intent
  --data-uri DATA_URI    specify a Uri to attach as data in the Intent
  --extra TYPE KEY VALUE
                          add an field to the Intent's extras bundle
  --flags FLAGS [FLAGS ...]
                          specify one-or-more flags to include in the Intent
  --mimetype MIMETYPE     specify the MIME type to send in the Intent
dz>
```

## Command

```
dz> run app.activity.start --component com.intrepidusgroup.learner com.intrep
idusgroup.learner.Lesson8AuxActivity --action com.intrepidusgroup.learner.cus
tom.intent.SEND --extra string 20200524 intent
dz>
```

