AppUse ותקיפת צד שרת של אפליקציות Android

מאת ברק טוילי

הקדמה

כולנו יודעים שהטלפונים החכמים מכילים הרבה מאוד מידע רגיש עלינו, מכרטיסי אשראי ועד להתכתבויות ב-WhatsApp, מה מיקומנו, תמונות וכו'.

כיום אנו רואים התפתחות ממשית בתחום הטלפוניה, בנקים וחברות כרטיסי אשראי מפתחות אפליקציות לניהול החשבון דרכן, אפליקציות צ'ט אשר מכילות היסטוריית שיחות שלנו, והרבה מאוד מידע חשוב שלנו מנוהל ע"י הטלפון החכם.

ארכיטקטורת מערכת ההפעלה אנדרויד מאפשרת למתכנת לנהל את המידע בצורה רחבה, לייצר רכיבים נגישים לאפליקציות באותו המכשיר, לשמור מידע במקומות מסוכנים ובקלות ניתן לנהל את המידע בצורה שגויה. בנוסף לכך, מתכנתים רבים אשר תכנתו עד היום צד שרת, צריכים לפתח אפליקציה אשר היא צד לקוח ואינם מודעים לסכנות האפשריות, ובכך מגדילים את מרחב התקיפה המוכר לתוקף עד לפני תקופת הטלפונים החכמים.

כאשר בודק מבצע מבדק חדירה לאפליקציית Android, הבדיקה מתחלקת ל-2 מרחבים עיקריים:

- **תקיפת צד הלקוח** - במרחב זה נבדוק חולשות בצד לקוח כגון שמירת מידע רגיש בצורה מסוכנת, שמירת סיסמאות בקוד, מניפולציות על broadcast receivers, Activities וכו'.
- **תקיפת צד השרת** - במרחב זה נבדוק חולשות אפליקטיביות בצד השרת כגון XSS, SQLI, Authorization Bypass, Authentication Bypass, וכו'...

על כל תת נושא מהנ"ל אפשר לכתוב מאמר שלם, לכן במאמר זה אתמקד רק בדרך שבה התוקף מקבל את המידע המירבי ביותר על צד השרת דרך אפליקציית ה-Android, איך הוא מנצל מידע זה, ונראה איך אנו תוקפים את צד השרת של האפליקציה בעזרת אפליקציה, בנוסף, לעיתים חושפים Web services רק עבור אפליקציות מובייל וזו הצורה היחידה למצוא אותם.

להבנת המאמר נדרשת היכרות בסיסית עם אפליקציות אנדרויד או נסיון עבודה עם האמולטור.

במידה ויש צורך בהסבר מעמיק על הטכניקות השונות, על האיומים ו/או דרכי ההגנה, אנו (AppSec Labs) מקיימים קורסים בנושא Android Application Hacking ו-Android Secure Coding באופן קבוע, לקבלת מידע נוסף, ניתן לפנות למייל: info@appsec-labs.com

הקמת סביבת עבודה

בכדי לבצע בדיקה על אפליקציית Android בודק ישתמש ב-Android SDK , אשר מכיל את האמולטור של Android שמבצע אמולציה מלאה של מערכת ההפעלה Android, בנוסף, על מנת לבצע בדיקה על אפליקציה יש הרבה מאוד כלים עוצמתיים העוזרים בביצוע הבדיקה כגון:

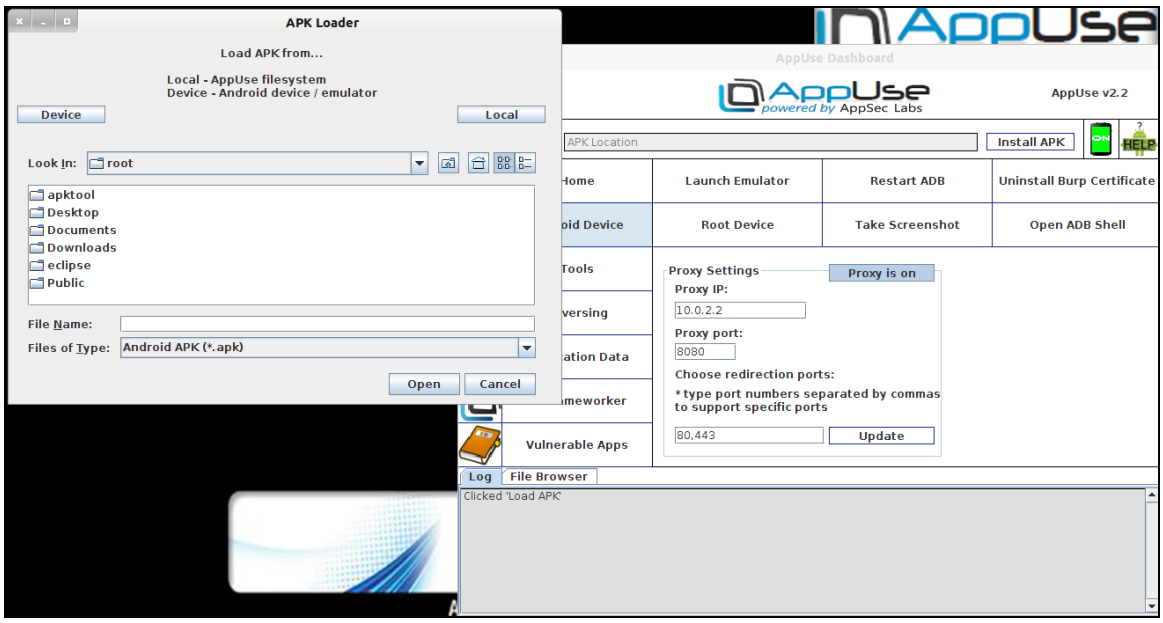
- Apktool
- Dex2jar
- Jdgui
- Burp
- apkAnalyzer
- sqlitebrowser
- ועוד כלים רבים.

אפיוז - סביבת עבודה לאנדרואיד

ב-AppSec Labs פיתחנו כלי בשם AppUse, אפיוז הינו מכונה וירטואלית (VM) אשר מספקת לבודק סביבה מקונפגת עם כל הכלים ההכרחיים. במערכת הוספנו סקריפטים והתאמות אשר חוסכות לבודק זמן רב מזמן הבדיקה. בנוסף לכך, מוטמע במערכת "אפיוז-דשבורד" אשר מאפשר להפעיל את כל הכלים בצורה קלה ומהירה ומייעל משמעותית את עבודת הבודק! ממליץ בחום לכל מי שמתעסק בתחום להוריד את אפיוז מהכתובת הבאה:

https://appsec-labs.com/AppUse

תמונה להמחשה של אפיוז-דשבורד:



במאמר אסביר כיצד לבצע תקיפה לצד השרת בעזרת אפיוז-דשבורד וגם בצורה ידנית בעזרת הכלים הרלוונטים.

קצת על האמולטור

האמולטור, כשמו הוא, מבצע אמולציה של מערכת ההפעלה של Android, האמולטור נוצר בכדי לתת לבודקים ולמפתחים לבדוק ולפתח אפליקציות ללא צורך פיזי בטלפון חכם.

האמולטור מגיע עם מספר כלים אשר מאפשרים לבצע פעולות שונות על האמולטור, לא אתעכב על כל אחד מהם, אך יש מספר רב של מקורות המסבירים על כל אחד מהם:

- http://developer.android.com/tools/help/emulator.html - Emulator
- http://developer.android.com/tools/help/adb.html -ADB

שלבי התקפה

התקפת צד השרת של אפליציית Android מורכבת מכמה שלבים:

- השגת אפליקצית היעד - איך תוקף משיג את האפליקציה המותקפת? מה הצעדים הדרושים לכך?
- Android Manifest - מה התוקף יכול ללמוד מקובץ ה-Manifest של האפליקציה המותקפת?
- Decompile APK and investigating the code - איך עושים Decompile לאפליקציה? ואיך נחקור את הקוד בכדי לאסוף נתונים חשובים (עם איזה שרת האפליקציה מדברת? ודרך איזה פורט?)
- יירוט התעבורה לפרוקסי - אילו אפשרויות יש בידי התוקף בכדי ליירט את התעבורה לפרוקסי?
- ניצול החולשה הקיימת בצד שרת - לאחר שיירטנו את התעבורה, דוגמא לזיהוי חולשה אפליקטיבית והסבר כיצד ננצל את החולשה.

השגת אפליקציית היעד

בכדי להשיג את האפליקציה המותקפת לבודק יש כמה אפשרויות:

1. השגת האפליקציה ע"י שמה או שם החבילה שלה (Package) דרך Google Play - במידה והאפליקציה זמינה ב-Google Play, יש אפשרות להוריד את האפליקציה למכשיר אמיתי או להשתמש באתרים המספקים שירותים המאפשרים להוריד APK ישירות מהגוגל פליי (באפיוז דאשבורד ניתן להשתמש בשירותים אלה ע"י הכפתור אשר נמצא תחת Tools -> Download APK From Google Play), כגון:
   - http://apk-dl.com/
   - http://apps.evozi.com/apk-downloader/
2. השגת האפליקציה המתוקנת על מכשיר - במידה והאפליקציה מותקנת על מכשיר זמין, יש באפשרות הבודק למשוך את האפליקציה מתוך המכשיר.

הכלי adb אשר מגיע עם ה-SDK המאפשר לבודק לבצע אינטראקציה עם האמולטור. כמה פקודות חשובות והסבר קצר עליהם ניתן למצוא כאן:
https://appsec-labs.com/blog/adb-common-commands/

במאמר אצא מנקודת הנחה שאתם יודעים כבר להשתמש בפקודות adb shell/pull/push הסטנדרטיות.

בנוסף, חשוב לציין כי במערכת ההפעלה Android יש 2 סוגי אפליקציות:

- **אפליקציות משתמש** - האפליקציות הנורמטיביות שכולנו מורידים מה-Google Play ומתקינים אותם, לדוגמא אפליקציית בנק שמאפשרת למשתמש לנהל את חשבונו. אפליקציות אלה נמצאות תחת התיקייה: /data/app/
- **אפליקציות מערכת** - אפליקציות של ה-ROM, אשר אחראיות לתפקוד המלא של ה-ROM, אפליקציות אלה בעלות הרשאות גבוהות יותר בדרך כלל ולא ניתנות למחיקה, לדוגמא אפליקציית SystemUI אשר מציגה למשתמש את ה-GUI של מערכת ההפעלה. אפליקציות אלה נמצאות תחת התיקיה: /system/app/

אבחנה וחקירה ראשונית - Android Manifest

כל אפליקציית אנדרויד מכילה קובץ מאניפסט אשר הינו בפורמט XML הנקרא AndroidManifest.xml.

קובץ המאניפסט מכיל מידע רב על האפליקציה כגון: שם האפליקצייה (Package name), פירוט הרכיבים במערכת (Activities, broadcast receivers, content providers), הצהרה על שימוש בהרשאות מסויימות, הצהרה על הרשאות אישיות ועוד. בכדי לצפות בקובץ המניפסט של אפליקציה, אנו צריכים להשתמש בכלים שעושים Decode (פיענוח) לקבציי האפליקציה, 2 האפשרויות הנפוצות ביותר לעשות זאת הן ע"י 2 הכלים, apktool ו-apkparser:

בכדי לצפות במניפסט בעזרת apktool נשתמש בפקודה הבאה:

```
apktool decode <apk name>
```

לאחר מכן תיווצר תיקיה בשם ה-APK ומשם נוכל לגשת אל קובץ ה-AndroidManifest.xml

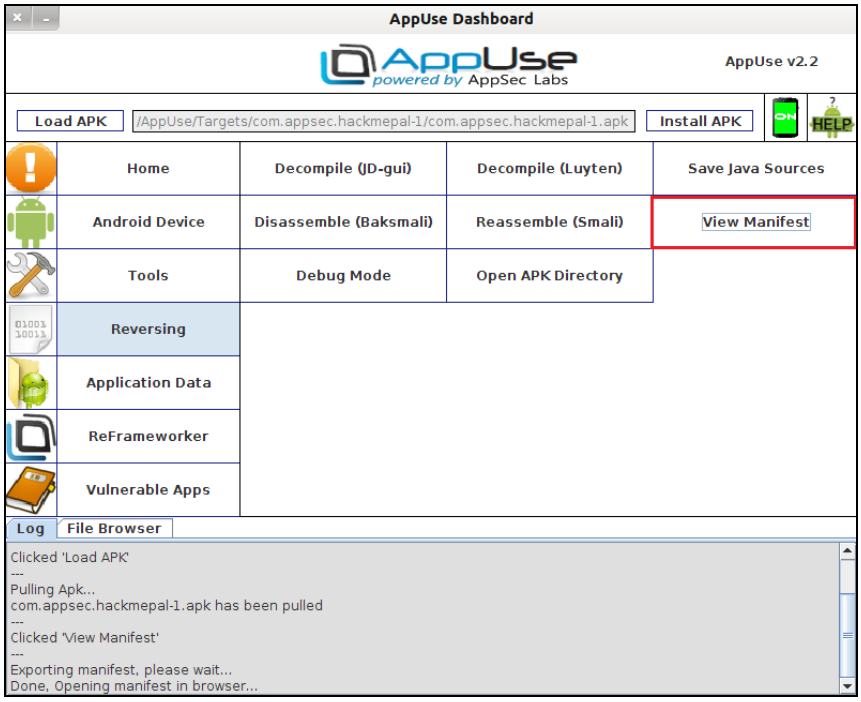
בכדי לצפות במניפסט בעזרת apkparser נשתמש בפקודה הבאה:

```
java -jar APKParser.jar <apk name>
```

במידה ואנו רוצים לייצא את המאניפסט לקובץ ולצפות בו בדפדפן אנו יכולים להשתמש בפקודה:

```
java -jar APKParser.jar APK_NAME.apk > AndroidManifest.xml
```

הדרך הקלה יותר לצפות במניפסט של האפליקציה תהיה באמצעות AppUse, בכך שנטען את ה-APK לתוך אפיוז דרך כפתור ה-Load APK ונלחץ על הכפתור View Manifest תחת מדור ה-Reversing:



מקובץ המאניפסט ניתן ללמוד הרבה מאוד על האפליקציה, הקובץ מספק לתוקף מידע רב על האפליקציה, מבלי לבצע פעולות מורכבות מידי.

בתמונה הבאה תוכלו לראות חלק מקובץ המניפסט של האפליקציה HackMePal (אפליקציית hack me אשר פותחה ע"י AppSec Labs לבדיקות ותרגולים, האפליקציה מדמה מערכת עשירה של פעולות אשראי וטרנזקציות כספיות):

```
−<manifest android:versionCode="1" android:versionName="1.1" package="com.appsec.hackmepal">
    <uses-sdk android:minSdkVersion="16" android:targetSdkVersion="17"> </uses-sdk>
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"> </uses-permission>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"> </uses-permission>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"> </uses-permission>
  −<application android:theme="@7F070001" android:label="@7F060000" android:icon="@7F020014" android:debuggable="true"
   android:allowBackup="true">
   −<activity android:label="@7F060000" android:name="com.appsec.hackmepal.Splash2">
     −<intent-filter>
        <action android:name="android.intent.action.MAIN"> </action>
        <category android:name="android.intent.category.LAUNCHER"> </category>
      </intent-filter>
    </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.List"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.AcDetail"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.AcHistory" android:exported="true"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Transfer" android:exported="true"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Register"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.AcAnother"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.UserList"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Admin"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Admin_UserList"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.More"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Requestmoney"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.AutoPay"> </activity>
    <activity android:label="@7F060000" android:name="com.appsec.hackmepal.Messaging"> </activity>
```

בעזרת התמונה למעלה, אנו יכולים לראות כי האפליקציה משתמשת בהרשאות לגשת לאינטרנט, לקרוא ולכתוב קבצים אל ה-SDCard. נוכל לראות זאת ע"י קריאת ההרשאות שהאפליקציה מבקשת תחת התגית uses-permission, נוכל גם לראות כי יש לה 2 Activities (מסכים) שהם exported (ניתן להפעיל אותם מאפליקציות אחרות) מה שיכול להוות בעיה מבחינת אבטחת מידע מכיוון שאפליקציות זדוניות יוכלו לתקשר עם מסכים (Activities) של האפליקציה ולבצע מתקפות שונות, כפי שאנו רואים, אנו אכן מקבלים מידע רב מהאפליקציה.

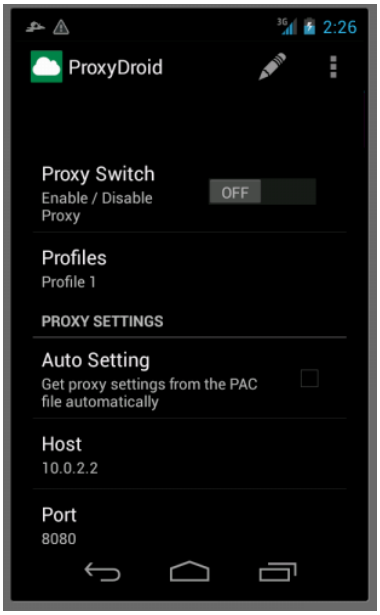
כמו שאנו רואים, האפליקציה משתמשת בהרשאות אינטרנט, זאת אומרת היא יכולה לתקשר עם שרת חיצוני, במידה והאפליקציה לא הייתה משתמשת בהרשאה זו, לא היינו יכולים לבצע תקיפות לצד השרת כי הוא לא היה קיים. במילים אחרות, מההרשאה אנחנו למדים שכנראה שקיים צד-שרת.

פרוקסי

לאחר שווידאנו שככל הנראה האפליקציה אכן מתקשרת עם שרת חיצוני, אנו נרצה ליירט את התעבורה שלה, בכדי לבצע מניפולציות שונות על הבקשות היוצאות ונכנסות. על מנת לעשות זאת, יש כמה וכמה אפשרויות להעביר את התעבורה דרך פרוקסי.

מערכת האנדרויד מספקת שירות מובנה של פרוקסי מאוד מצומצם ומוגבל, לכן נבנו כמה אפליקציות אשר מספקות הגדרת פרוקסי על המערכת למכשירים שהם ROOTED. אפליקציה מאוד מוכרת הינה ProxyDroid, אשר מספקת למשתמש אפשרות להגדיר פרוקסי על המכשיר בצורה קלה ונוחה.

בתמונה הבאה ניתן לראות את הגדרות האפליקציה:

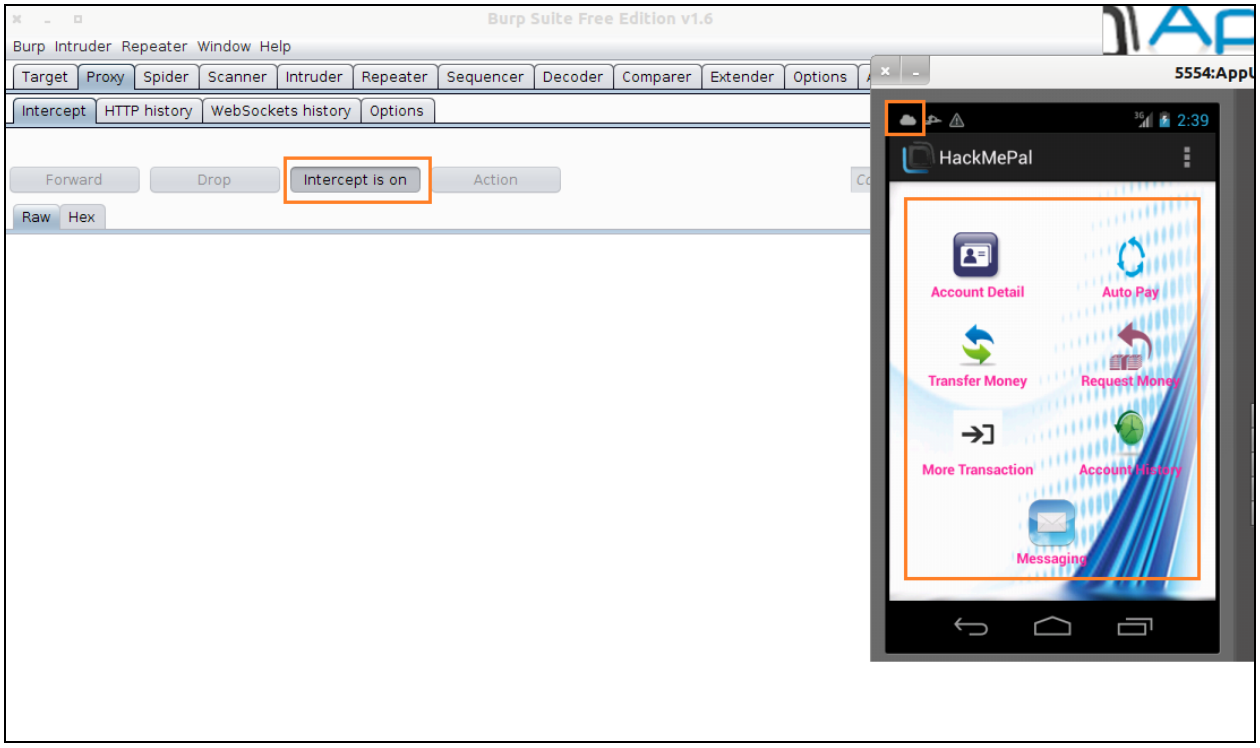


כפי שנין לראות, ה-HOST, זאת אומרת ה-IP של שרת הפרוקסי, הינו 10.0.2.2 כתובת זו מסמנת לאמולטור את הכתובת של המארח של האמולטור (המחשב ממנו רץ האמולטור).

בנוסף, ניתן לראות כי באפליקציה אין אפשרות להגדיר את התעבורה בפורטים ספציפים אשר יועברו לפרוקסי, הפורטים הקבועים הם 443 ו-80, זאת אומרת שבמידה ונרצה ליירט תעבורה אשר יוצאת בפורט אחר, לא נוכל לעשות זאת בעזרת האפליקציה הזו, בנוסף, האפליקציה תומכת רק בפרוטוקול HTTP, ולא תומכת בפרוטוקולים בינארים, ולכן לא נהיה אפקטיביים במיוחד.

לאחר שאנו נפעיל את שרת הפרוקסי (לדוגמא burp suite) ונדליק את הפרוקסי של ProxyDroid בכדי ליירט את התעבורה מהאמולטור אל burp, ננסה לבצע התחברות באפליקציה HackMePal (admin/admin) בכדי לתפוס את בקשת ההתחברות ולבצע מניפולציות עליה.

בתמונה הבאה, אנו רואים כי הצלחנו לבצע התחברות (ריבוע למטה מימין) למרות שהburp מקונפג ליירט את הבקשות (מלבן בתמונה משמאל) ושה-ProxyDroid מוגדר כראוי (סמל הענן בתוך הריבוע למעלה מימין):



מכאן אנו מסיקים כי האפליקציה לא מתקשרת עם השרת תחת הפורט 443 או 80, אלא פורט אחר אשר לא נתמך ע"י ProxyDroid.

שני עניינים שאנו צריכים לחקור:

- באיזה פורט האפליקציה מתקשרת עם השרת?
- איך נורה לאמולטור להעביר לפרוקסי בקשות בפורטים ספציפיים?

נתחיל מהעניין הראשון.

אבחנה וחקירה של קוד האפליקציה

כל אפליקציית אנדרואיד מכילה את הקובץ classes.dex. לאחר כתיבת קוד ב-JAVA וקימפול של הקוד, ל-bytecode, הוא מומר לקוד שה- JVM (Java Virtual Machine) מכיר בו בפורמט class.

בכתיבת אפליקציית אנדרויד קוד זה מומר לקובץ שה-Dalvik מכיר בו בפורמט dex., בכדי לרוץ על מכשיר אנדרוידאיד. קובץ זה מכיל את כל קוד האפליקציה ובעצם את לוגיקת האפליקציה.

בכדי לצפות בקוד המקור נצטרך לבצע מספר פעולות, הוצאה של הקובץ classes.dex מתוך ה-apk. הקובץ אשר מכיל את קוד ה-java של האפליקציה. לאחר מכן נמיר את קובץ ה-dex לקובץ jar, ואז נבצע decompile בעזרת כלים קיימים היודעים להחזיר קוד מקור קריא בקבצי JAR, ובכך נראה קוד שקרוב מאוד לקוד המקור של האפליקציה.

על מנת לחלץ ולהמיר את קובץ classes.dex ל-JAR נשתמש בכלי dex2jar :

```
unzip <apk name> classes.dex
```

```
dex2jar classes.dex
```

לאחר מכן ייווצר קובץ בשם classes_dex2jar.jar, נוכל לטעון אותו אל תוך כלי פענוח לJAVA, לדוגמא JD-GUI או Luyten .

כעת, יש לנו גישה לקוד המקור של האפליקציה, העובדה שתוקף יכול להשיג את קוד מקור האפליקציה חושפת בפניו מידע רב אודות האפליקציה ואודות הפונקציונליות שלה כגון סיסמאות אשר שמורות בקוד, לאילו קבצים האפליקציה כותבת ומה, איך והיכן היא שומרת נתונים רגישים וגישה לרכיבים אשר נרשמים באופן דינאמי בתוך הקוד ולא נמצאים במאניפסט (לדוגמא, dynamic broadcast receiver).

לאחר חקירה ואיבחון של הקוד, ניתן לזהות כי האפליקציה הנבדקת שלנו מתקשרת תחת הפורט 6543:

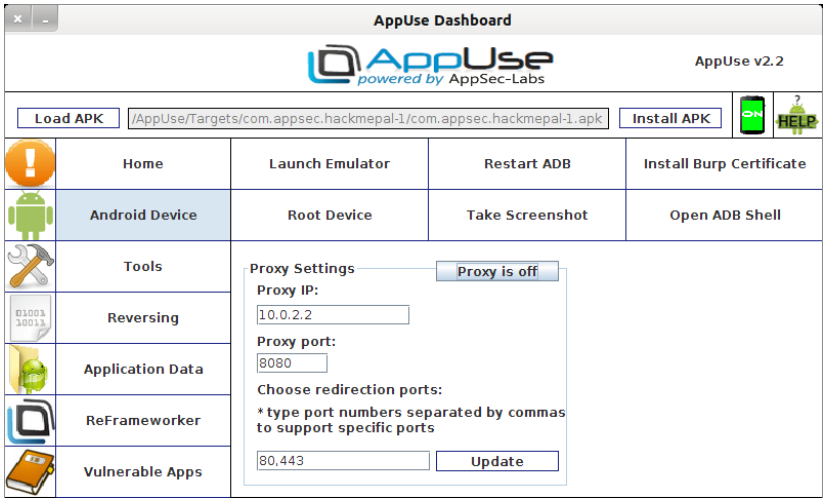
```
public class BrReciever extends BroadcastReceiver
{
  String amount;
  Context ctx;
  String host = "10.0.2.2";
  String port = "6543";
  String refuid;
  String uid;

  public void onReceive(Context paramContext, Intent paramIntent)
  {
    this.uid = paramIntent.getExtras().getString("uid");
    this.refuid = paramIntent.getExtras().getString("refuid");
    this.amount = paramIntent.getExtras().getString("amount");
    this.ctx = paramContext;
    new TransferWithRest(null).execute(new Void[0]);
  }
```

לאחר שאנו יודעים את הפורט שהאפליקציה מתקשרת תחתו, ניגש אל העניין השני.

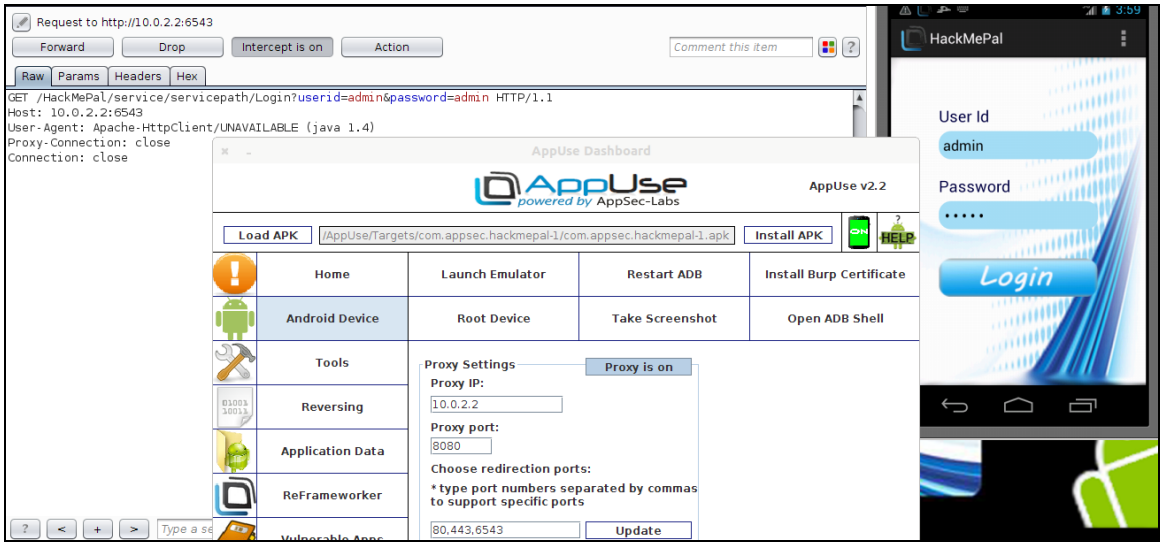
אפיוז פרוקסי

כאשר נתקלנו בבעיה, ש-ProxyDroid לא מיירט את התעבורה פרט לפורטים 80 ו-443, ניסינו לחפש כלי אחר שעושה את זה ללא הצלחה, לכן, בנינו כלי בעצמנו. אפליקציית AppUse, אשר מותקנת על אפיוז, הינה כרגע חסרת GUI והיא מאפשרת למשתמש להגדיר מ-AppUse Dashboard את הפרוקסי, ומאפשרת למשתמש לקבוע אילו פורטים ייורטו אל הפרוקסי שלו. התמונה הבאה מראה את מסך הגדרת הפרוקסי ב-Dashboard, כפי שניתן לראות, אפשר לקבוע את הפורטים אשר ייורטו בעזרת רשימת פורטים מופרדים בפסיק:



לאחר שהוספנו את הפורט הרצוי לרשימה ומילאנו את הפרטים של הפרוקסי שלנו, נבצע ROOT על המכשיר בעזרת הכפתור Root Device תחת המדור Android Device (מכיוון שמכשיר ROOTED הינו תנאי מקדים בכדי להפעיל את הפרוקסי) לאחר מכן, נפעיל את הפרוקסי על האמולטור ע"י לחיצה על הכפתור "Proxy is off" ולאחר מכן ננסה לבצע להתחברות לאפליקציה שלנו ובכך, נוכל ליירט את הבקשה ולבצע עליה מניפולציות.

התמונה הבאה ממחישה את המצב הנ"ל:



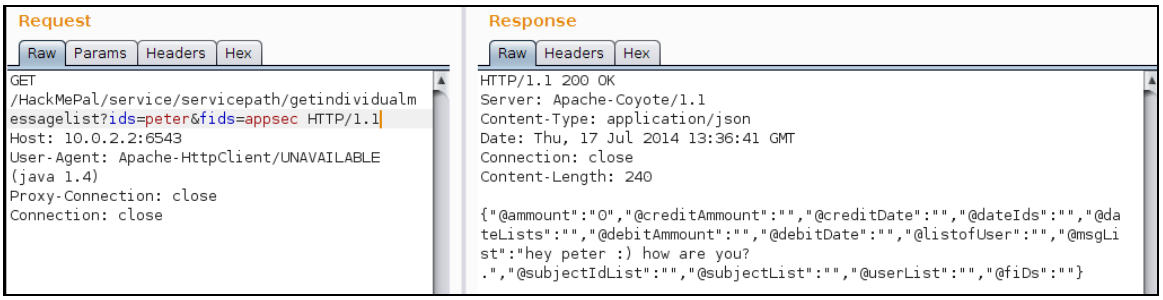
ניצול חולשות בצד השרת

לאחר שהצלחנו ליירט את התעבורה אל הפרוקסי שלנו, הגיע הזמן לבצע מניפולציות על בקשות אשר נשלחות מהלקוח אל צד השרת, נוכל לנסות לנצל מתקפות צד שרת כגון:

- XSS
- SQLI
- Authentication/Authorization bypass
- Parameter tampering
- DoS attacks
- ועוד.

אתן רק דוגמא אחת של ניצול חולשה ולא אכנס ליותר מידי פרטים בגלל שאני לא רוצה להתעמק במתקפות שונות אלא רק להדגים כיצד להגיע למצב שנוכל לתקוף את צד השרת.

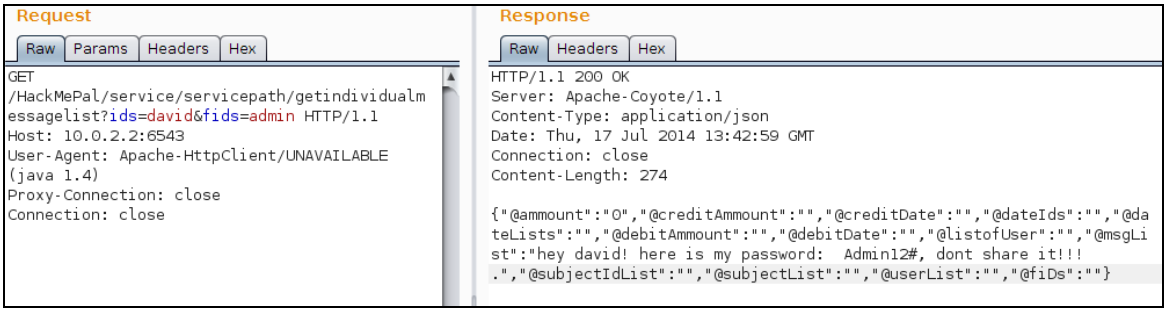
לאחר חקירה של הבקשות היוצאות מהאפליקציה נוכל להבחין בבקשה הבאה בעת כניסה לחלון שליחת הודעת צ'ט פרטית למשתמש באפליקציה:



כפי שנראה בתמונה, השרת שולח בקשה אל השרת בכדי לקבל את ההיסטוריה של שיחת 2 המשתמשים, ניתן לראות כי שם המשתמש של המשתמש הנוכחי וחברו לצ'ט מועברים כפרמטרים אל השרת (ids=peter,fids=appsec).

מתוך מבנה הבקשה ניתן להסיק במהירות שסביר שקיימת חולשה וננסה לבצע מניפולציה על הפרמטרים במטרה לעקוף את מנגנון האוטוריזציה ובעזרתו נוכל לראות שיחות של משתמשים אחרים במערכת.

נשנה את ערך הפרמטרים של שמות המשתמשים וננסה לראות שיחה בין המשתמש admin למשתמש david, ונראה האם השרת אוכף מנגנון אוטוריזציה:



כמו שניתן לראות בתמונה למעלה, השרת אכן הגיב וסיפק לנו את היסטוריית ההודעות של admin ו-david, ובכך הצלחנו להבין כי האפליקציה לא אוכפת מנגנון אוטוריזציה או שפשוט עקפנו אותו ☺.

חשוב לציין כי הדוגמא הנ"ל הינה חלק מאפליקציית HackMePal, אשר עוצבה להיות פגיעה ככל האפשר למתקפות ע"י חברת אפסק, אך יחד עם זאת, במהלך בדיקות חדירה רבות שבצעתי נתקלתי בלא מעט מצבים דומים מאוד למצב זה, ככל הנראה מכיוון שהמפתחים עצמם לא היו מודעים ולא לקחו בחשבון כי הנתונים אשר מגיעים אליהם יכולים להשתנות ע"י תוקף וסמכו על המשתמש, הנחת יסוד שהיא שגויה לחלוטין בתחום אבטחת המידע.

סיכום

לסיכום, במאמר זה הסברתי על כל שלבי העבודה של תוקף בזמן ביצוע התקפת צד שרת מאפליקציית אנדרויד, כיצד אנו משיגים את האפליקציה המותקפת ורואים כי היא אכן מתקשרת עם שרת חיצוני ע"י חקירת המאניפסט, כיצד אנו משיגים את קוד האפליקציה וחוקרים אותו, מה החסרונות של הפרוקסי באנדרויד וכיצד אנו מגדירים אותו בצורה נכונה.

תוקפים ובודקים אשר מגיעים מעולם האפליקטיבי ומנסים לבצע תקיפות על צד השרת מאפליקציית מובייל מבינים שלבצע תקיפת לצד שרת מתוך האפליקציה זו לא תמיד משימה קלה, כמו שניתן לראות.

אני ממליץ בחום לכל מי שמבצע / רוצה להיכנס לתחום בדיקות אבטחה ב-Android להשתמש באפיוז, זוהי פלטפורמה חזקה ומתעדכנת שתחסוך לך הרבה זמן בתפעול וביצוע בדיקות על אפליקציות אנדרואיד.

לינק לאפיוז:

http://appsec-labs.com/appuse

על הכותב

שמי ברק טוילי, אחראי פרויקט AppUse והמפתח העיקרי שלו, חוקר ויועץ אבטחת אפליקציות בחברת AppSec Labs. להצעות/הערות/הארות/בקשות ניתן לשלוח מייל לכתובת: appuse@appsec-labs.com, או למייל האישי: barak@appsec-labs.com.