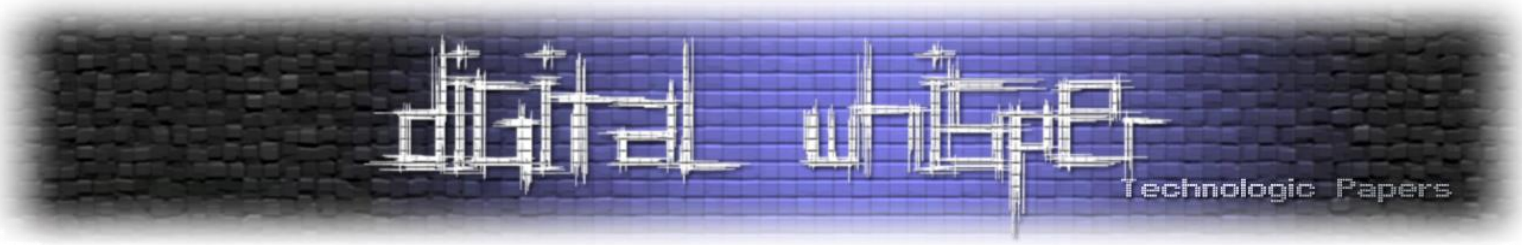


משטחי תקיפה באפליקציות Android - חלק א'

מאת 0x3d5157636b525761

הקדמה

זהו המאמר הראשון של סדרת מאמרים שתציג משטחי תקיפה (Attack surfaces) לאנדרואיד. נתמקד בעיקר בקוד האפליקציות (זה שכתוב ב-Java) אבל יש סיכוי שנבצע "זליגה" מפעם לפעם לקוד native פגיע.

בהמשך המאמר נציג כיצד ניתן לבצע פעולות מרוחקות על אפליקציית WheresMyDroid הפופולרית (בין 10 ל-50 מיליון התקנות), כולל שדרוג שלה (שאמור לעלות כסף) וכן פעולות שונות על המכשיר.

השתלשלות האירועים

- 20.04.2016 - גילוי הנקודות (שתוצגנה בהמשך) ב-WheresMyDroid.
- 21.04.2016 - פנייה אל מפתחי האפליקציה. לצערי הם לא הגיבו.
- 01.05.2016 - פנייה נוספת אל מפתחי האפליקציה. עדיין לא הגיבו.
- 07.05.2016 - פרסום (Public disclosure).

רקע בסיסי על אנדרואיד

בשלב זה נספק רקע בסיסי (ביותר) על אנדואיד. הלו המכירים את הארכיטקטורה מוזמנים לדלג הלאה.

אפליקציות ו-Dalvik

משתמשי אנדרואיד מפעילים אפליקציות. בניגוד לתהליך מסורתי על מערכת הפעלה, "אפליקציה" יכולה לרוץ מכמה תהליכים, ובדרך כלל יש לה יותר מ-Entry point יחיד. אפליקציות מגיעות ב-Archive עם סיומת APK (בפועל, ניתן לפתוח אותן כמו כל zip רגיל).

בתוך ה-APK יש מספר פריטים מעניינים:

- **AndroidManifest.xml** - זהו קובץ המכיל את כל ה-Metadata של האפליקציה, ובפרט את ההרשאות שבו האפליקציה משתמשת (כגון כתיבה ל-SD או קריאה של SMS) וכן את ה-Entry points שלה.

  לאפליקציה יכולות להיות מספר Entry points מסוגים שונים:

  - **Activity** - ה-Entry point הנפוץ ביותר, המציין אלמנט GUI שמשתמש יכול לעבוד איתו (די דומה ל-form).
  - **Service** - חלק באפליקציה שנועד לבצע פעולות ממושכות ברקע (למשל, לנגן מוזיקה).
  - **Content Provider** - כל מה שעלול לספק תוכן לאפליקציות אחרות. אף על פי שאנדרואיד מגיעה עם מספר Content providers שלה, אין מניעה מלייצר כאלה.
  - **Broadcast Receiver** - כל מה שיכול להתעדכן כתוצאה מ-Content provider אחר.
- **classes.dex** - כאן נמצאת (כמעט) כל הלוגיקה של האפליקציה. כל ה-class-ים נמצאים ממש כאן. קוד לאנדרואיד נכתב באופן טיפוסי בג'אווה (מעל מימוש בשם Apache Harmony), אך מקומפל ל-bytecode מיוחד בשם Dalvik (או "ART" על פלטפורמות חדשות). אנדרואיד מכילה JVM מיוחד בשם DVM שיודע להריץ את ה-bytecode הזה.
- **META-INF** - מכילה חתימות דיגיטליות (באופן זהה לקבצי JAR). אנדרואיד מחייבת כל APK להיות חתום עם חתימה כלשהי.
- **lib** - בתיקיה זו ישבו קבצי SO למיניהם. אנדרואיד מאפשרת ממשק בין Java לקוד native (שנכתב בדרך כלל ב-C או C++). ממשק זה ידוע בתור JNI, ומשומש באפליקציות רבות (כגון Facebook, Chrome ו-WhatsApp). אף על פי שאנדרואיד רצה על מעבדי ARM, תחת lib תופענה תתי תיקיות המציינות את שם הארכיטקטורה (כגון x86, ARMv7 וכדומה). יש לציין שרוב האפליקציות לא מכילות סתם ככה קוד native - הסיבות לקוד כזה הוא בדרך כלל ניסיון להשיג ביצועים טובים יותר (כגון מימוש RTP של WhatsApp) או שימוש בחבילות סגורות (כגון BoringSSL עבור Chrome).

מערכת ההרשאות

כמו בכל מערכת לינוקס, לכל קובץ ישנן הרשאות. ההרשאות הללו מציינות אילו ישויות יכולות לבצע קריאה (Read), כתיבה (Write) או הרצה (eXecute) של הקובץ. הישויות הן בעל הקובץ (User), קבוצת הבעלות של הקובץ (Group) ואחרים (Others). את ההרשאות משנים על ידי הפקודה chmod (ישנן פקודות מתאימות גם לשינוי ה-owner של הקובץ, למשל).

מתחת לפני השטח, אנדרואיד עצמה מייצרת User עבור כל אפליקציה. בטרמינולוגיה של אנדרואיד, נוצר Application ID חדש ("AID"), אף על פי שמדובר במשתמשי לינוקס לכל דבר ועניין. רוב ההרשאות של האפליקציה (כפי שצוינו בקובץ ה-manifest שלה) תתורגמנה לשייכות לקבוצות כאלה ואחרות (למשל,

ישנה קבוצה שכל מי ששייך אליה יכול לכתוב אל כרטיס ה-SD). בנוסף, בוצעו שינויים מסויימים בקרנל כדי לתמוך בהרשאות מסויימות (למשל, אכיפה על כך שרק מי ששייך לקבוצה מסויימת יוכל לפתוח socket).

תקשורת בין תהליכים (IPC)

אנדרואיד "הפשיטה" את רוב מנגנוני ה-IPC המסורתיים של לינוקס ומימשה דרייבר בשם ה-Binder. דרייבר זה אחראי לחלק גדול של ה-IPC במערכת. הארכיטקטורה של ה-Binder די מרתקת (ומזכירה במקצת ממשקי COM), ולמזלנו רוב כותבי האפליקציות לא צריכים לדבר עם הדרייבר ישירות. ישנן אבסטרקציות רבות לעבודה מול ה-Binder, כאשר האבסטרקציה הנפוצה ביותר ידועה בתור Intent.

ניתן לשלוח Intent אל אפליקציה ספציפית, בין רכיבים שונים של אפליקציה, או פשוט "לכל מי שמוכן לקבל את ה-Intent". כמובן, Intent יכול להכיל בתוכו מידע, וכך למעשה שליחת Intent-ים מעל ה-Binder נותנת לכותב אפליקציה המון כוח לבצע IPC כמעט ללא טיפול בכאבי ראש בדמות סנכרון או סיראליזציה.

כאשר שולחים Intent אל אפליקציה ספציפית ה-Intent ידועה בתור Explicit intent, וכאשר שולחים "לכל מי שעונה על קריטריון כלשהו" אז ה-Intent ידועה בתור Implicit intent. באופן כללי, Implicit intents נפתרים על ידי הגדרת פילטרים (Intent filters), שבדרך כלל מוגדרים ב-AndroidManifest.xml.

מנגנונים נוספים

ישנם מנגנונים נוספים מעניינים במערכת שכדאי לציין כבר בשלב זה:

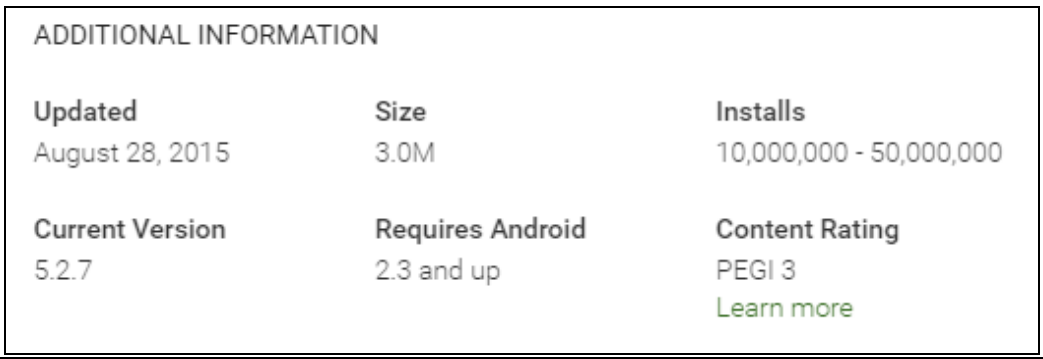
- לאנדרואיד יש גרסא משלה ל-SELinux (ידועה בשם SEAndroid). זה אומר שגם משתמש Root לא כל-יכול.
- באנדרואיד, הרשאות מסויימות יכולות להינתן רק לאפליקציות שמותקנות תחת ה-System partition (בניגוד לאפליקציות רגילות שמותקנות תחת ה-Data partition). האפליקציות הללו ידועות בתור System-apps, ועקרונית אין דרך להסיר אותן או להתקין חדשות כאלה אלא אם כן מבצעים Rooting למכשיר. ההרשאות המיוחדות הללו נותנות כוח רב לאפליקציות System - ביניהן היכולת להתקין APK חדש ללא התרעה ("התקנה שקטה") וכן קריאת לוגים של המערכת (logcat), שאליהם נכתב המון מידע שימושי ומעניין.

משטח תקיפה מבוסס SMS

משטח התקיפה הראשון שבו נתבונן הוא SMS. נדמה לעיתים כי מתכנתי אפליקציות לא חושבים על SMS כעל משטח תקיפה לגיטימי (בניגוד לגלישה לדפי אינטרנט, למשל), ולכן פעמים רבות סומכים על תוכן ה-SMS ללא עוררין.

לצורך ההדגמה, נבחן את אפליקציית WheresMyDroid, המותקנת על 10 מיליון - 50 מיליון מכשירים. מטרת האפליקציה היא לאתר את המכשיר במקרה והוא אבד או נגנב.

להלן פרטי האפליקציה, כפי שמופיעים ב-Google play:



הורדת ה-APK יכולה להתבצע במספר דרכים:

1. שליפת ה-APK מתוך ה-filesystem לאחר ההתקנה (דורש אסקלציה).
2. ביצוע Man in the middle על ה-Google services (דורש התקנה של סרטיפיקט ועדיין לא טריוויאלי).
3. הורדת ה-APK מאתר צד ג' בחינם.

ישנם חוקרים שלא מביעים אמון באתרי צד ג' מסוג זה, אך מנסיוני האישי אני יכול להגיד בפה מלא שרובם מהימנים. עם זאת, הייתי ממליץ תמיד על סביבת עבודה נקייה (כלומר, לא לחקור אפליקציה על המכשיר האישי שלנו). אני משתמש באופן אישי ב-apk-pure.com, אבל ישנם שירותים דומים נוספים כמובן.

לאחר שהורדנו את ה-APK, ניתן להתחיל לחקור אותו באופן סטטי. ישנם כלים טובים לכך:

- **apktool** - כלי מצויין זה מתרגם את ה-AndroidManifest.xml ל-XML אמיתי (ה-manifest המקורי הוא בינארי על אף הסיומת שלו), שולף resources וכן מבצע תרגום של classes.dex ל-SMALI code. אף על פי שנציין מהו SMALI code במאמרים הבאים, ניתן להגיד כי היחס בין SMALI ל- Dalvik bytecode הוא כמו היחס בין אסמבלי לשפת מכונה. נוסיף ונציין כי apktool יודע לבנות מחדש קבצי APK (repacking).
- **dex2jar** - מתרגם APK שלם ל-JAR. כמובן, חלק מהמידע הולך לאיבוד (למשל, ה-manifest).
- **jd-gui** - לא באמת כלי עבור אנדרואיד, אבל כלי זה מהווה Java disassembler ויודע לעבוד עם קבצי JAR.
- **JEB** - כלי חקירה all-in-one עבור אנדרואיד. לא חינמי (בניגוד לכלים הקודמים שהזכרתי).

אני התרגלתי להשתמש בכלים החינמיים, אבל JEB נחשב לאיכותי ביותר - שווה לבדוק אותו.

כעת ניתן לגשת לניתוח הראשוני. השלב הראשון יהיה להריץ apktool:

```
D:\research\WheresMyDroid>java -jar D:\1337\apktool\apktool_2.1.0.jar d -o apktoolout WheresMyDroid_v5.2.7.apk
I: Using Apktool 2.1.0 on WheresMyDroid_v5.2.7.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\USER\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

D:\research\WheresMyDroid>dir apktoolout
 Volume in drive D has no label.
 Volume Serial Number is 8A1F-2AE0

 Directory of D:\research\WheresMyDroid\apktoolout

05/07/2016  11:17 AM    <DIR>          .
05/07/2016  11:17 AM    <DIR>          ..
05/07/2016  11:17 AM            12,049 AndroidManifest.xml
05/07/2016  11:17 AM               406 apktool.yml
05/07/2016  11:17 AM    <DIR>          assets
05/07/2016  11:17 AM    <DIR>          original
05/07/2016  11:17 AM    <DIR>          res
05/07/2016  11:17 AM    <DIR>          smali
               2 File(s)         12,455 bytes
               6 Dir(s)  179,415,449,600 bytes free
```

לאחר הרצה מוצלחת, נוכל להתבונן ב-AndroidManifest.xml:

```
<uses-permission android:name="com.android.vending.BILLING"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.BATTERY_STATS"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
```

המון הרשאות - נראה מבטיח! שימו לב במיוחד להרשאות SEND_SMS ו-RECEIVE_SMS שישמשו אותנו.

ניתן לראות גם כי קיים Broadcast receiver בשם SMSReceiver שייקרא בכל פעם שנקבל SMS:

```
<receiver android:name=".receivers.SMSReceiver">
    <intent-filter android:priority="999">
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

כעת כדאי לבחון את הקוד של ה-Receiver הזה באמצעות dex2jar (ולאחר מכן jd-gui):

```
D:\research\WheresMyDroid>D:\1337\dex2jar\dex2jar-0.0.9.15\d2j-dex2jar -o d2j.jar WheresMyDroid_v5.2.7.apk
dex2jar WheresMyDroid_v5.2.7.apk -> d2j.jar
```

הפונקציה onReceive תיקרא עם ה-Intent הרלוונטי. להלן הפלט של dex2jar (הקוד קטוע אך ממשיך עוד):

```
public void onReceive(Context paramContext, Intent paramIntent)
{
  String str1 = "";
  String str2 = "";
  boolean bool = false;
  try
  {
    Bundle localBundle1 = paramIntent.getExtras();
    Object[] arrayOfObject;
    SmsMessage[] arrayOfSmsMessage;
    if (localBundle1 != null)
    {
      arrayOfObject = (Object[])localBundle1.get("pdus");
      arrayOfSmsMessage = new SmsMessage[arrayOfObject.length];
    }
    label494:
    label627:
    label628:
    for (int i = 0;; i++)
    {
      String str10;
      String str11;
      String str12;
      String str13;
      String str14;
      String str15;
      String str16;
      String str17;
      if (i >= arrayOfSmsMessage.length)
      {
        if (bool) {
          break label494;
        }
        Intent localIntent1 = new Intent(paramContext, SMSHandlerService.class);
        Bundle localBundle2 = new Bundle();
        localBundle2.putString("FROM", str1);
        localBundle2.putString("MESSAGE", str2);
        localIntent1.putExtras(localBundle2);
        paramContext.startService(localIntent1);
        SharedPreferences localSharedPreferences = GF.getSavePref(paramContext);
```

הרבה APK-ים משתמשים בשיטות Obfuscation שונות להסתרת המטרות האמיתיות של ה-class-ים שלהם. בדרך כלל זה כולל פתיחה של המחרוזות רק בזמן ריצה וכן שינוי של ה-class-ים לשמות כגון a, b, c וכדומה. למזלנו, כאן לא ננקטה גישה זו ולכן קל לנו מאד להבין את מטרות ה-class-ים השונים.

לעיתים קוד הפלט של dex2jar יוצא מכוער או אפילו לא מדוייק, ולעיתים התרגום לא מצליח (ואז צריך להתחיל לתרגם ידנית קוד SMALI לקוד Java).

אם כן, מתודת ה-onReceive תקבל את ה-SMS בתוך שדה ה-extra של ה-Intent. ניתן לראות כי עבור כל הודעה יישלח Intent נוסף - הפעם אל class בשם SMSHandlerService, עם שני שדות: FROM ו-MESSAGE.

הבה ונבחן את הקוד הרלוונטי ב-SMSHandlerService:

```
protected void onHandleIntent(Intent paramIntent)
{
  Log("--onHandleIntent--");
  setupAnalytics();
  this.pref = GF.getSavePref(this);
  loadSettings();
  Log("Get bundle passed from SMS Receiver");
  Bundle localBundle1 = paramIntent.getExtras();
  this.from = localBundle1.getString("FROM");
  this.message = localBundle1.getString("MESSAGE");
  Log(2, "From: " + this.from + "\n" + " - Message: " + this.message);
  if ((this.from == null) || (this.message == null)) {
    Log("from or message is null");
  }
  String str;
  do
  {
    return;
    if ((this.from.contains("@")) || (this.message.contains("@")))
    {
      Intent localIntent = new Intent(this, SMSEmailHandlerService.class);
      Bundle localBundle2 = new Bundle();
      localBundle2.putString("FROM", this.from);
      localBundle2.putString("MESSAGE", this.message);
      localIntent.putExtras(localBundle2);
      startService(localIntent);
      return;
    }
    str = GF.trimText(this.message);
    if (str.startsWith("wmd open")) {
      openApp(this.message);
    }
    this.ringAttWord = GF.trimText(this.ringAttWord);
    this.gpsAttWord = GF.trimText(this.gpsAttWord);
    this.camAttWordBack = GF.trimText(this.camAttWordBack);
    this.camAttWordFront = GF.trimText(this.camAttWordFront);
    this.lockAttentionWord = GF.trimText(this.lockAttentionWord);
    this.unlockAttentionWord = GF.trimText(this.unlockAttentionWord);
    this.wipeAttentionWord = GF.trimText(this.wipeAttentionWord);
  } while (!containsAttentionWord(str));
  if (this.whtblkListEnabled.booleanValue())
  {
    if (whtblkCheck(this.whtblkWhiteEnabled))
    {
      Log("we can continue");
      checkAttWord(str);
      return;
    }
    Log("rejected we can't continue");
    return;
  }
  checkAttWord(str);
}
```

נשים לב לקוד הברור שיצא - במידה והתקבל Intent מייל (מזוהה על ידי סימן ה-@), אז מטפל בו ה-SMSEmailHandlerService. אחרת, הקוד הנוכחי יטפל בו. אם נתעלם מפקודת wmd open, נשים לב שדי הרבה member-ים מאותחלים, ולאחר מכן בודקים האם ההודעה מכילה את אחת מהמילים הללו

(בפונקציה הפנימית containsAttentionWord). לאחר מכן קוראים אל checkAttWord. נבחן את המתודה הזו (הקוד לא מלא):

```
private void checkAttWord(String paramString)
{
  Log("checkAttWord");
  if (paramString.contains("wmdprounlock"))
  {
    Log("Version changed to pro");
    GF.getSavePref(this).edit().putInt("version", 1).putInt("nag_count", 0).commit();
    return;
  }
  if (paramString.startsWith("wmdinstalled"))
  {
    Log("Message equals respond string");
    Analytics.Event(tracker, "feature_used", "sms_action", "installed_request");
    GF.sendMessage(this, this.from, getString(2131165286));
    return;
  }
  if (paramString.startsWith(this.ringAttWord))
  {
    Log("Message equals ring attention word");
    ringFeature(this.from);
    return;
  }
  if (paramString.startsWith(this.gpsAttWord))
  {
    Log("Message equals GPS attention word");
    gpsFeature(this.from);
    return;
  }
  if (paramString.startsWith(this.camAttWordBack))
  {
    Log("Message equals camera back attention word");
    cameraBackFeature(this.from);
    return;
  }
  if (paramString.startsWith(this.camAttWordFront))
  {
    Log("Message equals camera front attention word");
    cameraFrontFeature(this.from);
    return;
  }
  if (paramString.startsWith(this.lockAttentionWord))
  {
    Log("Message equals remote lock attention word");
    lockFeature(this.from, paramString);
    return;
  }
  if (paramString.startsWith(this.unlockAttentionWord))
  {
    Log("Message equals unlock attention word");
    unlockFeature(this.from);
    return;
  }
```

אז מה יש לנו?

- SMS ניקלט על ידי SMSReceiver.
- SMSReceiver שולח Intent אל SMSMessageHandler עם שדות ה-FROM ו-MESSAGE.
- מתודת onHandleIntent מתבצעת, וכתלות בתוכן ההודעה עלולה להתבצע אחת מהפעולות הבאות:
  - עדכון האפליקציה לגרסת Pro (ההודעה "wmdupgrade").
  - בדיקה האם האפליקציה מותקנת (ההודעה "wmdinstalled").
  - צלצול.
  - שליפת נתוני GPS.
  - לקיחת תמונה (באופן שקט) מתוך המצלמה הקדמית או האחורית (דורש גרסת Pro).
  - נעילה ושחרור של הטלפון (דורש הפיכת האפליקציה ל-Device administrator).
  - ביצוע Wipe למכשיר (דורש הפיכת האפליקציה ל-Device administrator).
- ההודעות שגורמות להפעולות (להוציא את השתיים הראשונות שהן hard-coded) קונפיגורביליות, אך מקבלות ערכים default-יים:
  - צלצול: "WMD Ring".
  - שליפת מיקום: "WMD GPS".
  - תמונה: "WMD Camera Front" ו-"WMD Camera Back".
  - נעילה ושחרור: "WMD Lock" ו-"WMD Unlock".
  - ביצוע Wipe למכשיר: "WMD Wipe".
- את הערכים הללו ניתן לשלוף בקלות מתוך strings.xml (שמכיל את ה-String resources של האפליקציה).

תובנות נוספות:

- האפליקציה לא מתריעה לאחר ההתקנה על כך שהערכים הם default-יים.
- שדרוג ל-Pro אמור לעלות כסף. עם זאת, נראה שניתן לשדרג בחינם על ידי שליחת "wmdupgrade".
- חלק מהפיצ'רים באפליקציה מאופשרים רק לאחר שדרוג ל-Pro. תוקף יכול לשדרג מרחוק כמובן.
- חלק מהפיצ'רים דורשים מהאפליקציה להיות Device administrator, מה שלא מתבצע כברירת מחדל.
- אם נסכם, עבור התקנה "רגילה" של האפליקציה ללא שינויים, תוקף יכול לבצע (בסיכוי טוב):
  - שדרוג ל-Pro.
  - צילום שקט מהמצלמות. התמונה עולה באופן שקט ואוטומאטי לשרת מרוחק, ולינק נשלח לתוקף.
  - שליפת נתוני GPS. הנתונים נשלחים באופן שקט כ-SMS החוצה לתוקף.
- נציין כי יש אפשרויות של האפליקציה לביצוע whitelist, אך היא לא דולקת כברירת מחדל.

בעיות נוספות

תחת אנדרואיד, רכיבים (ו-Service-ים ביניהם) יכולים להיות exported או לא להיות exported. כאשר רכיב הוא exported אז הוא יכול לקבל Intent-ים מאפליקציות אחרות, וכאשר הוא לא exported אז הוא לא. ניתן לראות כי SMSHandlerService הוא Exported, ולכן אפליקציה סוררת (ללא הרשאות קריאת SMS-ים, ללא הרשאות מצלמה וכדומה) יכולה לשלוח Intent שייתפס על ידי ה-SMSHandlerService ויבצע פעולות בשמה.

מסקנות

1. בתחילת מאמר זה סקרנו (באופן גס מאד) את היכולות של אנדרואיד. לאחר מכן, התמקדנו במשטח תקיפה מבוסס SMS-ים, והצגנו כיצד אפליקציה תמימה למראה יכולה לשמש ככלי ריגול לכל דבר.
2. כמו רוב החולשות האפליקטיביות באנדרואיד, הבעיה נבעה ממתכנת שסומך לחלוטין על הקלט הנכנס, בצירוף חוסר הבנה על כך שניתן לבצע Reversing לאפליקציה עצמה. נפוץ מאד לראות סיסמאות hard-coded באפליקציות לאנדרואיד, והמצב הנוכחי הוא שאין תהליך מסודר שמבצע Review על קוד של אפליקציה.
3. אפליקציה עם ערכים דיפולטיים שלא מתריעה בפני המשתמשים שלה על כך שיש לעדכן את הערכים הללו היא אפליקציה בעייתית (באופן דומה לקוד הסודי לתא הקולי הסלולרי או הסיסמא לראוטרים הביתיים של חלק גדול מאיתנו).

אף על פי שסדרת המאמרים תופץ גם בעברית, אני מזמין את הקורא השקדן לקרוא את הפוסט המקורי בבלוג שלי, בכתובת: http://securitygodmode.blogspot.com