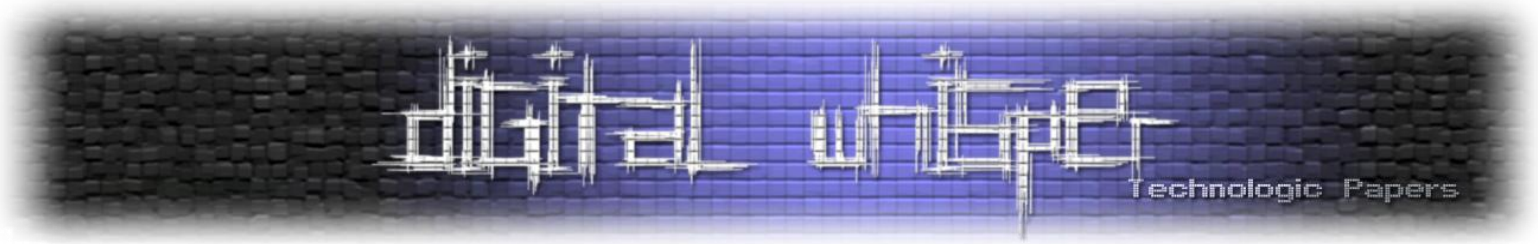

האם אנדרואידים חולמים על תולעים אלקטרוניות?

מאת אייל גל (codeScriber)

הקדמה

לפני שנתחיל אני חייב לציין שהכותרת תיראה ללא ספק הרבה יותר טוב באנגלית ומבוססת על ספרו של הסופר Philip K. Dick: "Do Android Dream of Electric Sheeps".

למי שעוד לא קלט את הקטע, במאמר זה אני אנסה להציג מספר נקודות מעניינות למחשבה למשתמשים ולמפתחים במערכת ההפעלה "אנדרואיד". אני אינני איש אבטחה , אך אני נמצא בעולם התכנות לסלו לר כ-4 שנים, בזמן הזה יצא לי להתקל בדי הרבה בעיות וקוד תוכנה עבור מערכות שונות כגון סימביאן, QT, J2ME, WM, וכמובן אנדרואיד. כמו בכל דבר בחיים , חלק מאותן מערכות הפעלה /פלטפורמות היו יותר קלות לעבודה מהאחרות, חלקן נותנות יותר מרחב פעולה מאחרות, אבל כולן מכוונות למכשירים הקטנים שרובנו מחזיקים כיום בכיס המכנס. מתכנתים ל-Web או לאפליקציות שולחניות בטח ישאלו את השאלה המתבקשת: מה בעצם ההבדל בין מה שאנחנו עושים לעבודה של מפתח לסלולר? אז אני אתחיל בהצגה קצרה של פלטפורמת אנדרואיד ואיך היא עובדת , ואיך כותבים למערכת כזו , בגדול. משם נמשיך למודל האבטחה הממומש באנדרואיד, כיצד גוגל מתכננים להגן על המשתמשים מבלי להגביל מפתחים ביכולות פיתוח והפצה של אפליקציות.

1. גוגל מכים שנית: מה היא אנדרואיד?

אנדרואיד היא מערכת הפעלה "שלמה", אני לא מסווג אותה כמערכת הפעלה לסלולר, כי זה לא בהכרח נכון, היא תוכננה ועוצבה למכשירים "מעוטי יכולת" כגון מכשירים סלולים אבל לא מזמן HP השיקו NETBOOK עם אנדרואיד עליו , בנוסף קיים פורט של ה- GIT Repository עבור x86. אנדרואיד בנויה כולה על בסיס לינוקס, ולכן היא מכילה קרנל של לינוקס , סט כלים בסיסיים של לינוקס, ועוד מספר ספריות שנותנות שרותים כגון ספריית שמע, תצוגה (Open–GL) וכדומה.

השכבה התחתונה של המערכת כתובה כמובן בשפת C על גבי לינוקס ומיועדת למעבדי ARM (מאחר והיא נועדה לשימוש בעיקר תחת מכשירים סלולרים, שזה המעבד המוביל בהם כיום, אך כמו שציינתי, יש פורט ל-x86). מעל לשיכבה זו רץ Dalvik ה-virtual machine של גוגל, למה אני לא כותב פשוט JAVA virtual machine? כי זה לא, זה למעשה לא מריץ Java Byte Code אלא Byte Code שונה שנקרא DEX והומצא ע"י גוגל, ההנחה המרכזית היא שזה כך מפני סיבות יעילות, אבל הסיבה האמיתית שלא להשתמש ב-VM כזה או אחר של SUN או IBM.

אם כך, **בעצם כל אפליקציה של אנדרואיד רצה בעצם תחת מכונה ויר טואלית משלה** וברמת העיקרון, היא מנותקת מכל שאר האפליקציות באותה המערכת. זה הבט אחד של האבטחה של אנדרואיד, אך כמובן שאם זאת הייתה **כל האמת** זאת הייתה ללא ספק מערכת די משעממת ולא מאפשרת פיתוחים מעניינים במיוחד.

דבר נוסף שכדאי לדעת זה שבעיקרון באנדרואיד כל האפליקציות, כולל אלו שהגיעו עם המערכת ונחשבות כ-"System Apps" שוות, כלומר **שוות לאפליקציות צד שלישי** . אני מוצא את זה לא מדויק, אבל זה הרבה יותר הוגן מההפרדה במערכות אחרות כגון WM או Iphone.

הקוד של אנדרואיד אומנם פתוח לחלוטין- מהקוד של ה-SDK ועד לאמולטור, אבל גם אם אתם כותבים תוכנית ב-SDK אתם לא יכולים לגשת לכלל ה-API הקיימים, לעיתים תראו בקוד המקורי annotations כאלה: "@hidden" שבגדול אומר שגם אם המתודה Public אתם עדיין לא יכולים לגשת אליה, יש לגוגל לא מעט API פנימיים שהדרך היחידה לשנות אותם בעצמכם היא לייצר Image חדש לטלפון ולהתקין אותו, דבר שאפשרי כיום בעיקר עבור HTC והפורטים של 86X אך לא עבור יצרנים אחרים פשוט בגלל הדרך בה צורבים ROM חדש למכשיר...).

2. איך נראית תוכנית\אפליקציה באנדרואיד?

תוכנית שנכתבה לאנדרואיד, בשונה מתוכניות במערכות אחרות מורכבת מקומפוננטות, לא DLL או כמה JAR-ים, אלא, באופן לוגי היא מורכבת מכמה סוגי קומפוננטות שנציג עכשיו, את התיאור המלא שלהם תוכלו למצוא ב:

http://developer.android.com/guide/topics/fundamentals.html

ישנם ארבעה סוגי קומפוננטות: Activity, Service, BroadcastRecievers ו-ContetProvider.

Activity: היא היחידה הויזואלית של אנדרואיד, דרכה מתקיימת האינטראקציה עם המשתמש. יחידה זו מאפשרת יצירת ממשק משתמש , קבלת קלט מהמשתמש והצגת הפלט , היחידה הזו בד"כ מייצגת מסך יחיד, כלומר אם יש אפליקציה מרובת מסכים, כל מסך כזה ייוצג ע"י activity נפרדת. ה-Activities מסודרות במחסנית עבור כל אפליקציה כך שכשהמשתמש לוחץ על "back" הוא אוטומטית מסיר יחידה אחת מהמחסנית ומקבל את היחידה הקודמת לה שם. יש פרמטרים שמאפשרים לקבוע איך Activity תתנהג במידה ואפליקציה אחרת (לא הזאת שבה היא מוגדרת) תרצה להריץ אותה, לא נכנס לזה פה, אך ניתן לקרוא על זה כאן:

http://developer.android.com/guide/topics/fundamentals.html

(תחת הכותרת: Affinities and new tasks ואילך.)

Serivce: כשמה כן היא, היא הקומפוננטה אשר אחראית על נתינת שרות אשר מורץ ברקע. אין אינטראקציה ויזואלית ישירה עם המשתמש ובד"כ התוצאה הישירה של אינטראקציה עם שרות היא Intent שמופץ במערכת או Dialog או Notification שמוצג למשתמש. דוגמא טובה לשרות כזה היא מערך הניהול של ה-SMS שמחכה ל-SMS וברגע שכאלה מגיעים הוא שולח Intent לכל מי שמאזין ומודיע למשתמש על SMS חדש שהגיע. במידה ויבחר המשתמש לקרוא את אותו SMS הוא יגיע ל-Activity שזה תפקידה, תפקיד ה-Service עצמו תם. ל-Services יש בדרך כלל נטייה לקבל עדיפות גבוה יותר להשארות בזיכרון כשהמערכת נדרשת לשחרר משאבים

BroadcastReciever: זוהי קומפוננטה זעירה יחסית אשר נדרשת להיות יעילה. תפקידה פשוט מאוד: להגיב במיידי על EVENT מסוים ששודר במערכת. ישנו מספר לא קטן של אירועי מערכת שכבר קיימים באנדרואיד (כגון: אירוע על סיום ה-BOOT, אירוע על קבלת SMS, אירוע על לחיצת הכפתור של המצלמה ע"י המשתמש וכו') הקומפוננטה הזו היא חלק ממנגנון ה-IPC באנדרואיד.

ContentProvider: באנדרואיד לכל אפליקציה יש גישה למערכת קבצים פרטית משלה ואין לה גישה לקבצים של אפליקציות אחרות, זאת כמובן מפאת שקולי אבטחה, לכן אנדרואיד מאפשרת לאפליקציות לגשת למידע השייך לאפליקציות אחרות ע"י ממשק אשר נקרא "ContetProvider". ממשק זה פועל באופן הבא:

- התוכנית מבקשת מאנדרואיד לבצע שאילתה על URI מסיים עם פרמטרי WHERE, ORDER BY וכו, הממשק מוגבל במקצת יחסית ל-DB מכיוון שמאחוריו יכול לעמוד כל מימוש כולל רשת, מערכת קבצים, HASHTABLE וכדומה.

- לאחר השאילתה, במידה ויש לכם את ההרשאות לגשת למידע שאתם רוצים (שנקבע על פי ה-URI ששולחים לשאילתה) תקבלו את התשובה באובייקט Cursor שהוא אותו אובייקט שמתקבל משאילתת SQL.

דבר נוסף שלא קשור באופן ישיר לכלל הקומפוננטות, אך בהחלט קשור לכולן הוא האובייקט: Intent. אוביקט Intent הוא בעצם IPC Message אשר רץ במערכת בין הקומפוננטות השונות ומעביר הודעות מאחת לשניה.

Intent כולל בד"כ ACTION שאומר מה צריך לעשות כמו VIEW או EDIT למשל, הוא יכול להכיל הגדרת CLASS ספציפי שאותו הוא נדרש להריץ, הוא יכול להכיל URI ובנוסף יש לו יכולת להכיל נתונים שונים מסוג : Int, Boolean, Char, String, Parcelable כאשר האחרון הוא Interface שמוגדר ב-SDK לסריאליזציה של אובייקטים.

ה-Intent יכול להריץ Activity מתוך Activity אחרת מצד אחד, אך הוא גם יכול להחזר תשובה (Result) ל-activity הקוראת. הוא יכול לשדר EVENT מסויים כגון הגעה של SMS ולהכיל בתוכו את הפרטים של אותו SMS, תוכן, HEADER וכדומה.

דבר נוסף שחשוב להזכיר בחלק זה עבור אפליקציית אנדרואיד שלמה, הוא מה שמחבר את כל החלקים לאפליקציה אחת, פרט לקובץ APK שבה ארוזה האפליקציה כמובן ☺.

AndroidManifest.xml: זהו בעצם meta-file שמגדיר את האפליקציה, מה שמה, שם החבילה שמכילה אותה, איזה קומפוננטות יש בתוכה ? (Services Activities etc'...) איזה הרשאות יש לכל אחת ואחת מהקומפוננטות ואילו הרשאות יידרשו מאפליקציה חיצונית בכדי לגשת בהצלחה לקומפוננטות של האפליקציה.

בנוסף לכל Activity או Serivce יש מספר פרמטרים שאפשר לכוון ע"י ה-manifest, אבל לא נכנס אליהם כאן, לתיאור מלא של כל הגדרות ניתן לפנות לאתר המפתחים של אנדרואיד:

http://developer.android.com/guide/topics/manifest/manifest-intro.html

אחד הדברים היותר חשובים בנושא בו נוגע המאמר ו- **שמפתחים פחות נוטים להתסכל עליו ולקחת אותו בחשבון**, הוא המקטעים העוסקים בסוגי ההרשאות השונות . ועליהם נדבר בקטע הבא: מודל האבטחה באנדרואיד.

מודל האבטחה באנדרואיד:

"האם אנדרואיד בטוחה?" שאלה מעניית והתשובה העקרונית היא כן, לא פשוט לכתוב Exploit למערכת כזו, מפני שמדובר בקוד שהוא פתוח אין פה "Security by obscurity". הקוד מבוסס על קרנל של לינוקס יציב ואף מקבל עדכונים בעץ משלו, **אין זה אומר שאין חורי אבטחה בקרנל**, אבל יהיה הרבה יותר קשה למצוא אותם. בנוסף סט הכלים המגיעים עם אנדרואיד הוא מצומצם , מה שמקל על סגירת חורי אבטחה ברובם (לעומת מערכת לינוקס מלאה שבה גם תוכנות צד שלישי ניתנות להתקנה).

שוב, אין הבטחה שאין כלל באגים וחורי אבטחה , אבל יש מייל של גוגל שמיועד לדיווח על חורי אבטחה שנמצאו ועדכונים שוטפים לעץ ה-GIT.

לא חקרתי, ובמאמר זה אני גם לא אדבר על 2 בעיות פוטנציאליות שיכולות להיות באנדרואיד:

- שימוש ב-NDK שבו ייתכנו חורי אבטחה.
- יצירת Image ניפרד משלך והפצתו ברשת.

במאמר זה נתרכז ב-SDK עצמו, וכיצד צריך לשמור על מספר חוקים בסיסיים על-ידי כלים שכבר ניתנו ע"י גוגל כדי לשמור על פרטיות המשתמש ואיכות התוכנה שלנו.

נתחיל עם הבסיס לאבטחה עבור אפליקציות שכתובות ב-SDK של אנדרואיד:

כל אפליקציה מקבלת מזהה UID משלה, לא כמו מערכת יוניקס רגילה בה כל תוכנית כמעט רצה על ה-UID של המשתמש שהריץ אותה, כאן כל תוכנית מקבלת UID משלה ולכן הגישה לקבצים שלה, ומסדי ה-SQLIGHT שלה מותנה ביכולת לקבל גישה לאותו משתמש, זאת אומרת: בדרך כלל- לא אפשרי.

בנוסף מאחר ש **כל** אפליקציה מורכבת מקומפוננטות (שמעתם נכון, גם תוכניות מערכת בנויות באותה צורה) כל קומפוננט **יכולה** להיות פרטית או ציבורית בנוסף אפשר להגדיר שכל קומפוננטה תדרוש גישה עם Permission מסוים.

כל ה-Permissions שהאפליקציה מבקשת או שהאפליקציה מגדירה לטובת ניצול עתידי לצד שלישי (עבור הקומפוננטות שלה) מוגדרות בתוך AndroidManifest.xml ונרשמות בזמן ההתקנה שלה, וכן גם ווידוא מול המשתמש שהוא מוכן לתת לאפליקציה גישה לכל מני API's 'מסוכנים' מתרחש בזמן ההתקנה.

את הדוגמאות שאני מציג כאן אני לוקח מתוך מצגת על רכיבי אבטחה באנדרואיד שלמיטב הבנתי הוצגה ב: Black-Hat Summit. את המצגת ניתן למצוא כאן:

http://siis.cse.psu.edu/slides/android-sec-tutorial.pdf

נשאלתי בעבר איך אפשר, לאחר ההתקנה של האפליקציה לוודא שהאפליקציה שלי תרוץ בכל BOOT.
יש שתי תשובות לכך: הראשונה: לרוב אין בכך בכלל צורך בכך, כי תגובה לאירועים חיצוניים כמו קבלת SMS לדוגמה הרבה יותר יעילים. בכל מקרה אין ביכולתכם להאזין ב-"SERVER SOCKET" על פורט מסוים כאשר המכשיר ב- GSM MODE (כלומר לא מחובר ל-WIFI). השניה: במידה ואתם בכל זאת רוצים את העליה של האפליקציה שלכם בזמן ה-BOOT או יותר נכון בסיומו, שימו ב-MANIFEST בקשה למימוש ב-API ו-BraodcastReciever:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.apps.example"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".SecActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    <receiver android:name="BootFinishedReciever"
android:enabled="true">
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED"/>
          </intent-filter>
    </receiver>
</application>
```

```
    <uses-sdk android:minSdkVersion="3" />

<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED">
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>
</manifest>
```

כמו שאתם רואים יש בקשה אחת ל-permission ובקשה אחת (שונה) להאזין לאירועים, האירוע הספציפי ניתן בתוך ה-TAG: intent-filter ומתייחס לאירוע ה-Boot המיוחל. באופן דומה ניתן להאזין ל-SMS. TAG השם בכל אחד מה- BroadcastRecievers מתייחס ל-Class שיורץ ברגע שהאירוע יקרה, Class כזה חייב לרשת מ- BroadcasrReciever. כמו בכל דבר באבטחה עיקר הבעיה היא המשתמשים והיכולת לבצע עליהם Social Engineering, כלומר- לגרום להם להתקין את האפליקציה ולהשתמש בה בעוד היא (התוכנית) שולחת את המידע הפרטי שלהם מהמכשיר לשרת מרוחק. אתן דוגמא לרעיון שעשוי לעבוד:

דמיינו משחק תפקידים מגניב דרך הרשת הסלולארית

ישנו שליט מבוך שמגדיר אזורים בארץ (או בעולם) כיער, הרי געש וכו', אתם והחברים שלכם משחקים בעולם הזה, מקבלים משימות דרך הטלפון ואתם כמובן צריכים לבצעם (בואו נעזוב כרגע כל מני דברים מוסריים\חוקיים שקשורים לעניין הזה, הרי זה יכול לעבוד... ☺), אתם יכולים להציע לכל החברים שלכם בטלפון לשחק! המשחק כמובן צורך פרטים מהרשת ולכן זקוק לגישה, בנוסף עם ייתכנו SMS-ים משליט המבוך ולכן נדרשת הרשאה לקבלה של SMS-ים כאלה ייתכנו אפילו אינטגרציות מלהיבות נוספות עם לוח השנה וכדומה.

מה שבעצם קרה פה, זה שעבור משחק כזה פתחתם את המכשיר שלכם עבור אותו ספק משחקים ואתם סומכים עליו לחלוטין. הבעיה היא שכל אחד יכול להעלות אפליקציה כזאת לרשת!

שלא כמו ב- Iphone, ה-Store עבור אנדרואיד אינו מחייב, הוא אף אינו מחייב חתימה רשמית שאומרת שאתם גוף בר-סמכא! ומכאן, הדרך להוצאת כלל רשימת הקשר שלכם והעלאה שלה לרשת האינטרנט, כולל כל הפרטים, כולל אירועים מלוח השנה שלכם, כולל היכולת לשלוח SMS-ים דרך הטלפון שלכם ללא ידיעתכם דיי קצרה אני חושש. אך זה לא ניגמר פה.

גם אם יש אפליקציה שמכילה מידע רגיש כמו אפליקציות לשמירת סיסמאות אתרי אינטרנט או ששומרת מיקומי חברים (פיזית) או שניגשת לחשבון הבנק שלכם- ואותה אפליקציה לא כתובה עם ההגנות הנכונות אפליקציה צד שלישי תוכל לנצל זאת.

למה הכוונה?
לכל Activity ו-Service במניפסט יש TAG בשם Exported מסוג בוליאני, הוא קובע בעצם האם אותו רכיב יהיה זמין (או נגיש) לאפליקציות אחרות, פרט לשלכם כמובן. בברירת המחדל הוא זמין לכולם, ושוב אם יש Serivce שמצפין, לדוגמה, פרטים אישיים, ומסוגל גם לעשות את הפעולה ההפו כה, ומשתמש חיצוני יכול לגשת אליו, הרי שכאילו ולא הצפנתם דבר! אם נסתכל על המניפסט הקודם שהצגתי נוכל לראות בו שניתן לדאוג שה-Activity המרכזי לא יהיה חשוף:

```
<activity android:name=".SecActivity" android:label="@string/app_name"
android:exported="false" >
```

זה רק דבר אחד שניתן לעשות, (ודרך אגב, מניסיוני, בד"כ לא עושים אותו) הדבר הבא הוא לגבי שליחת Broadcast באפליקציה שלכם, כמו שהמערכת יכולה לשלוח Broadcast של הודעת SMS שהגיע, כך גם כל אפליקציה ואפליקציה מסוגלת לשלוח Broadcasts משלה. שימו לב שניתנת לכם האפשרות להגדיר Permission שמחייבים אפליקציות אחרות להצהיר על אותן Permissions כדי שיוכלו להשתמש בהם.

דוגמא לשליחה כזו:

```
Intent i = new Intent("org.apps.example.myAction");
        i.putExtra(MISSION_TYPE, MISSION_SEARCH);
                    sendBroadcast(i, "org.apps.example.RECIEVE_MISSIONS");
```

בדוגמא, זו אם אפליקציה אחרת לא תגדיר במניפסט שלה:

```
<uses-permission
android:name="org.apps.example.RECIEVE_MISSIONS"></uses-permission>
```

אבל תנסה בכל זאת להאזין למשימות, היא תקבל SecurityException! מאוד רצוי להגן על הרכיבים הפנימיים של אפליקציה, בייחוד על שידורים של אירוע כדי למנוע sniffing לא רצוי.

Last But not Least ☺ ישנו אובייקט באנדרואיד שנקרא PendingIntent שמיועד, כשמו, לשלוח Intent מאוחר יותר. העניין הוא איך שעובד האובייקט הזה. הוא מכיל שדה reference למזהה ייחודי אשר מזהה במערכת את ה- Activity\Serivce\BroadcastReciever שאליו שייך ה- PengingIntent וכאשר משדרים את אותו Intent לבסוף הוא מורץ כאילו רץ מאותו:

Activity\service\BraodcastReciever

המקורי. היכולת הזאת מיועדת בעיקר לדלגציה של קומפוננטות פנימיות ושימוש ע"י שרותי מערכת כגון Alarm שמקבל PendingIntent כקלט אותו יריץ לאחר פקיעת הטיימר. הבעיה היא, במידה והשתמשתם באובייקט כזה, כיצד אתם מגינים עליו? בעיות שיכולות להיגרם אם אובייקט כזה מגיע למקום שהוא לא אמור להגיע אליו הן:

1. סיום Service שלכם ללא הרשאתכם.

2. הרצה של Activities פנמיים ללא הרשאה ע"י שינוי ה-Class שמוגדר לריצה בתוך ה-PengingIntent.

3. הקרסת התוכנית עקב שינוי פרמטרים ב-PengingIntnet ושליחתה.

כדי למנוע מצבים כגון אלו, תדאגו להשתמש ב-PengingIntent רק על מנת לייצר Callbacks נידחים (כל מני טיימרים למיניהם, הרצה של מקטע תוכנית אחרי שינה וכו') ותוודאו שאתם מפרטים ב-Intent את ה-

Class שהוא יעד ה-Intent (כלומר אל תייצרו implicit intents) והכי חשוב, שיהיה משתנה פנימי שלא יהיה נגיש מחוץ ל-Scope של התוכנית שלכם.

לסיכום

אנדרואיד ללא ספק נחשבת למערכת **יחסית** בטוחה, למודת ניסיון מקודמיה ועם פלטפורמה בשלה היא בהחלט מרשימה. טעויות או חוסר שימוש במנגנוני הביטחון שניתנו עם ה-SDK עלולים ליצור מצבים בהם מידע של משתמש עלול להיות חשוף לכל.

במציאות שבה להרוס מכשירים, או לכתוב וירוס שפוגע בכל המידע במכשיר נהפך ל- "פחות מעניין", אך להשיג את כלל המידע שיש לך על המכשיר, או להשתמש במכשיר שלך בשמך ללא ידיעתך הפכו להיות המטרה, אנדרואיד עדיין חשופה לסיכוני אבטחה כמעט כמו כל פלטפורמה אחרת ובסופו של דבר ערנותו של המשתמש נחוצה כדי להשלים את התמונה, ואפילו אז, לעתים קשה לראות איפה ישנו עוקץ.

יש לי הרגשה שעוד נראה מאמרים דומים למאמר זה וכתבות על האבטחה תחת אנדרואיד, אם זה יהיה בנושא ה-SDK, סיפורים על אימג'ים שמכילים קוד זדוני ונשתלו באתרים לגיטימיים של גרסאות אנדרואיד להורדה ובין אם זאת פרצה חדשה ב-NDK.

בקיצור יהיה מעניין, שימו לב למה שאתם מתקינים ומאיפה זה בא!