WarDialer בכפות ידיך

מאת עדן ברגר

הקדמה

לפני מספר שבועות התחלתי פרויקט חדש בשם Android WarDialer, הטמעת יכולות WarDialing לאנדרואיד.

Wardialing הינה שיטה למיפוי קווי טלפון שהייתה נפוצה בשנות ה-80 וה-90, בתקופה שטלפוניה הייתה טכנולוגיה עיקרית בתקשורת בין אישית, בין המכונות לעצמן ובין האנשים למכונות.

המטרה שלה, בדומה ל-"ציידי IoT/IoE" היא למצוא מכונות שמאזינות לקווי טלפון והיא עושה זאת ע"י חיוג למספר לא מוכר וניתוק לאחר שני צלצולים (בהנחה שלמכונה לוקח לענות עד שני צלצולים). השימושים הם ע"י חובבנים וחוקרי אבטחה או אנשי תחזוקה של חברות גדולות שרוצים לבדוק/למפות את הטלפוניה בהן. למשך תקופה ארוכה המכשירים היחידים שהיו יכולים לבצע את המשימה היו מרכזיות טלפוניה (מוגדרות מראש למשימה או מתוכנתות מחדש) או מחשבים עם תוכנה וחומרה מתאימה.

בשנת 2009 יצא הפרויקט WarVOX שאוחד אל תוך Metasploit ומשתמש ב-VoIP במקום קו הטלפון.

דרך פעולה:

1. להתקשר למספר מרשימה.
2. לחכות שני צלצולים.
3. לנתק.
4. אם מישהו ענה, לשמור את המספר ולעבור הלאה.
5. אחרת, לעבור הלאה אל המספר הבא.

אם נרצה להגדיל את הסיכויים שזו באמת מכונה נתקשר בשעות נכונות.

תוצאות:

אחרי שמיפינו טווח מספרי טלפון מסוים, אנחנו יכולים להתחיל לבדוק מה עלה בחכתנו, האם זה פקס, שער חשמלי (של מושב או קיבוץ), מכונת קולה, בקרת רמזורים וכן הלאה. בהגדרות יותר מורכבות

התוכנה גם תקליט את השיחה שנענתה ואפילו תזהה בעצמה האם מדובר בבן אדם או מכונה וכך לחוקר נשאר רק להאזין להקלטות ולבחון את דרכיו.

סיכונים:

- ספק הטלפון בקלות יוכל להבין שזו תוכנה מחייגת ולא אדם.
- האנשים שיראו שיחה שלא נענתה עשויים לחזור למספר.
- חברות בעלות מספר רב של טלפונים עשויות להתעצבן מסריקה כזו.

על הקוד

הקוד המלא משוחרר בגיטהאב וכתוב בג'אווה, אתן כאן כמה דוגמאות לחלקים ממנו כמו איך להוציא שיחה, איך לנתק אחרי זמן מסוים ואיך להבין האם מישהו ענה. על מנת להבין איך להוציא שיחה צריך להכיר מונח בשם **Intent**.

Intent זו בעצם הדרך של אפליקציות לתקשר אחת עם השנייה באנדרואיד, Intent זה אובייקט וקוראים לו כך בגלל שהוא מסמל "כוונה" למשהו. מערכת ההפעלה אנדרואיד שולחת Intent-ים המעדכנים את האפליקציות (שמבקשות) בדברים הקשורים לפעילות, כגון - שיחות, הודעות, כיבוי המסך או המכשיר ועוד.

כך שבשביל לחייג אנו צריכים ליצור Intent מסוג מסוים, למלא בו את מספר הטלפוןואז לשלוח אותו החוצה. כך זה נראה:

MainActivity.java

```
Intent call = newIntent(Intent.ACTION_CALL);
call.setData(Uri.parse("tel:050-000000"));
startActivity(call);
```

- שימו לב שאנו משתמשים ב-String למרות שזה מספר טלפון. ואז נוסיף במניפסט הרשאה לחיוג:

AndroidManifest.xml

```
<uses-permissionandroid:name="android.permission.CALL_PHONE" />
```

מיד אחרי תחילת השיחה נרצה להתחיל ספירה לאחור של כמה שניות שהמשתמש בחר עד לניתוק השיחה.

השארתי את השניות לבחירת המשתמש בגלל ששמתי לב שכאשר אני מתקשר מגולן טלקום השיחה מתחילה עם צלצול אחד אקסטרה בטון שונה ואני מניח שזה המעבר בין המרכזיות של גולן טלקום למרכזיות של פלאפון ורק משם ליעד.

נתחיל ספירה לאחור ונכניס את זה לתוך try בשביל לנהל את ה-Error במקרה שקרה אחד:

MainActivity.java

```
try {
Thread.sleep(secondsPicker.getValue(); *1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
```

הבעיה העיקרית בלכתוב תוכנות עם sleep זה שהוא תוקע את הריצה ואת ה-View, במקרה הזה הטלפון נהיה משותק. הדרך הנכונה לפתור את זה היא להשתמש ב-Callback שחוזר כאשר פונקציה מסתיימת וכך אפשר להתחיל פונקציה אחרת מיד אחריה.

הדרך המהירה לפתרון היא לשים את מנגנון החיוג והניתוק, כולל ה-sleep בתוך Thread נפרד, באנדרואיד Thread חדש ממומש כך:

MainActivity.java

```
new Thread(new Runnable() {
public void run() {
// Code that runs on a different thread
  }
}).start();
```

בשביל לנתק את השיחה השתמשתי בפונקציה שמצאתי ברשת, היא לא Straightforward ולכן אשאיר אותה בגיטהאב.

על מנת לדעת האם השיחה נענתה או לא היה חלק Tricky ובסופו של חיפוש מצאתי את התשובה בתגובה לתגובה ב-Stackoverflow (רמת יאוש = קריאת תגובות של תגובות).

הדרך למימוש היא על ידי בדיקה אחרי שהשיחה הסתיימה, כמה זמן היא נמשכה (Duration). לאחר השניות שחיכינו והניתוק, נחכה עוד שתיים/שלוש שניות בשביל לוודא שהמידע שאנחנו מקבלים הוא המעודכן ביותר ואז נבדוק מה היה משך השיחה:

MainActivity.java

```
נצהיר על משתנה ונאתחל אותו באפס:
intcallDuration=0;
נביא את רשימת שיחות האחרונות:
Uri contacts =CallLog.Calls.CONTENT_URI;
CursormanagedCursor=this.getContentResolver().query(contacts, null,
null, null, null);
assertmanagedCursor!=null;
נשמור את מספר הטור של משך השיחות:
int duration1 =managedCursor.getColumnIndex(CallLog.Calls.DURATION);
נבחר בשיחה האחרונה שנעשתה (moveToLast) ואז נכניס את המידע מהטור של משך השיחות:
if( managedCursor.moveToLast() ) {
callDuration=managedCursor.getInt(duration1);
}
managedCursor.close();
כעת אפשר להשתמש ב-callDuration
```

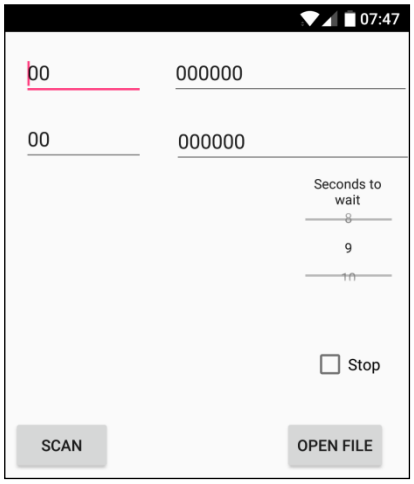
בשביל זה אנחנו נוסיף במניפסט הרשאה לקרוא את השיחות האחרונות:

AndroidManifest.xml

```
<uses-permissionandroid:name="android.permission.READ_CALL_LOG" />
```

על האפליקציה

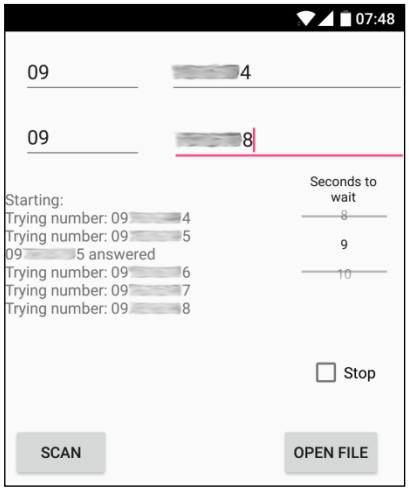
כך נראית האפליקציה כאשר מפעילים אותה:



- אין הגבלות בשדות והם Long כדי לאפשר מספרים ארוכים כמו 1800 או מספרים מחו"ל.

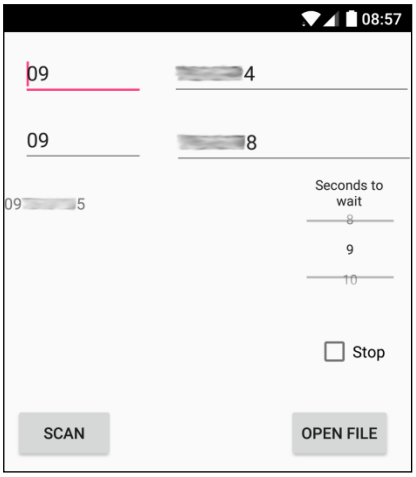
לאחר שנקבע טווח ונלחץ על SCAN הטלפון יתחיל לחייג, נוכל לעצור בכל רגע ע"י ניתוק השיחה וסימון Stop בצד.

בגמר הסריקה המסך יראה כך:



כשאחד מהמספרים עונה, נוצר קובץ בשם של המספר הראשון בטווח בתוך תיקיית Documents, בפעם הבאה שנפתח את התוכנה היא תזכור את הטווח האחרון שסרקנו ותפתח את הקובץ במקרה שהוא קיים.

היא תראה כך:



רשימת TODO

- להקליט שיחות שנענו.
- לזהות האם מדובר בבן אדם או מכונה.
- לחסום שיחות נכנסות מהטווח שנסרק או שנמצאות בשיחות שחויגו (והם לא חלק מרשימת הטלפונים).
- לייצא את הרשימה ל-CSV, JSON ו-Share לאימייל.
- תמיכה ברשימת מספרים מקובץ ולא טווח.

פיצ'רים עתידיים

- לצוד מכשירים ספציפיים, ע"י הוספת קובץ סאונד של הצליל שאנו מצפים לשמוע מאותו מכשיר (תודה לאפיק קסטיאל על הרעיון)
- להתקשר מכמה טלפונים במקביל ובתאום אחד עם השני (פטנט של Sandstorm)
- להשתמש במספר מהרשימה, מספר צלצולים ושניות בין לבין כל שיחה רנדומליים בשביל להקשות על הזיהוי של ספק הטלפון

פרויקטים דומים

קיימות בשוק אפליקציות עם מטרות דומות, כגון DemonDialing אשר מטרתה לתפוס קו במרכזיות עמוסות (כמו מתקשר המאה בתוכנית רדיו), התוכנות נקראות AutoDial וAutoRedial ב-GooglePlay.

SMS Bomber - שנועדה להפציץ מספר טלפון בהודעות SMS.

אפליקציות שמאזינות להודעות SMS ובמקרה שהגיעה המילה הנכונה הן מתקשרות לשער של המושב בשביל לפתוח אותו וכך עוקפות את האבטחה של "פתיחת השער על ידי טלפונים מורשים בלבד".

פרויקטים עתידיים

סך הכל מומש מגוון של כלי הטלפוניה מחדש על אנדרואיד, אך עדיין לא מצאתי AutoDialer שמתקשרת למספר מרשימה ומנגנת הודעה קולית בעת המענה (נראה שאפשרי רק בעזרת מחשב).

או Call Bomber שמתנהגת כמו DemonDialer רק שנועדה להוציא קו משימוש ע"י חיוג מ-Dual-SIM, חיוג מבוזר מכמה מכשירים או ניצול שירותי "אימות על ידי שיחת טלפון".

סיכום

WarDialing ו-DemonDialing אפשריות כיום בכמה לחיצות בזכות התפתחות הטלפונים החכמים והוזלת מחירי השיחות.

עם הנגישות אל תכונות הטלפון שהטלפון החכם הביא, יש לנו את האפשרות להערים או להתחכם על מנגנוני אבטחה מבוססי טלפוניה, מבלי הצורך לחבר את המחשב לקו טלפון ולהגדיר מחדש מרכזייה.

דוגמאות

מה יקרה אם נעשה Call Bomb / DemonDialing על שער של מושב, או על רשת הפקסים בבית משפט? או נניח אפליקציה שמשנה את המספר Follow me של הטלפון בשביל ליצור "פרוקסי טלפון" לטלפונים אחרים (Reverse follow me).

דוגמא ל-PT מוצלח בעזרת WarDialing, ניצול מודמים ועד כמה מסובך להגדיר אותו ניתן לראות בקישור הבא:

http://blog.aujas.com/hack-phone-ring-test

(נראה כי נכון לכתיבת שורות אלו המאמר ירד, אך עדיין ניתן לראותו בעזרת Google Cache או באמצעות Wayback Machine).

על עצמי

שמי עדן ברגר, איש סיסטם לינוקס, מתכנת וחובב אבטחת מידע. אנא צרו איתי קשר בכל הנוגע לפרויקט או פרויקטים דומים, אני פתוח להצעות בכל הנוגע ל:

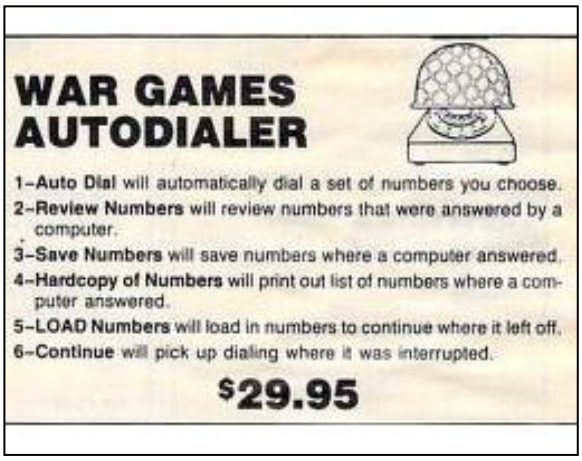
- פרויקטי OpenSource.
- משרות בתחום.
- פרויקטים ל-Commercial Use.

כתובת האימייל שלי היא:

Eden2036@gmail.com

מקורות והשראה

- archive.org - מגזין משנת 1985:



- חומר על phreaking:

tucops.info/tucops3

- מאמר על טלפוניה ועל הפרוטוקול SIP:

digitalwhisper.co.il/DW5-1-SIP-ASTERISK

- האפליקציה ב-Google Play:

play.google.com/com.bergereden.wardialerfree

- האפליקציה ב-GitHub:

github.com/android-wardialer