Digital Whisper
Technologic Papers

כספת של סוכריות גומי

מאת ניר גלאון

הקדמה

מאמר זה הינו מאמר המשך ל"מנגנוני אבטחה באנדרואיד" שפורסם בגיליון ה-31. במאמר הקודם נגענו בנושאים כגון החשיבות של אבטחת המידע ועל יסודות האבטחה בעולם האנדרואיד, על מרכז הגישה של המכשיר, על הסכנות ועל דרכים להתמודד איתם. מומלץ לקרוא את המאמר הקודם. למידה על הנושא היא חשובה ביותר, אך אם לא נשאר מעודכנים - פספסנו את הרעיון. במאמר זה אציג את העדכונים והחידושים שגרסאות 4.1 Jelly Bean ו-4.2 מביאות עמן בתחום האבטחה (כגון ASLR, סריקת רוגלות ב-Play ועוד), ובנוסף נגע במספר נקודות למחשבה על חורים הקיימים במערכת.

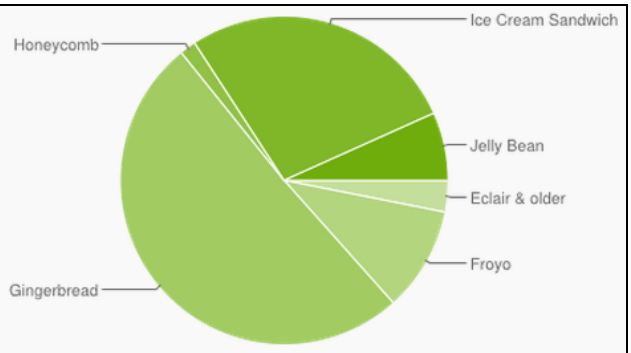
בסיס הבעיה של אנדרואיד

לפני שנתחיל, הייתי רוצה לציין שעולם האנדרואיד מתקדם מהר, אפילו מהר מאוד. גוגל מתקנים כל חור אבטחה שמתגלה ומוסיפים שכבות אבטחה במהירות יוצאת מן הכלל, אך דבר אחד הם מפספסים והוא עדכוני אבטחה.

בעוד שהאפליקציות לאנדרואיד מתעדכנות ברקע, ואף נהוג לקבל היום עדכוני OTA הכוללים שיפורים מסיביים למדי, אנדרואיד עדיין נמצאת מאחור מאוד בתחום הזה. בסיס הטעות הוא שהמערכת ניתנת ליצרניות והן משנות אותה, ובהמשך לכך לא מוציאות את העדכונים במהירות הנדרשת (שלא נדבר על זה שהן רוב הפעמים לא מוציאות עדכונים כלל).

כדי להמחיש את הבעיה מצורפת בעמוד הבא טבלת החלוקה על פי גרסאות (נכון לדצמבר 2012).

Version	Codename	API	Distribution
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.3%
2.1	Eclair	7	2.7%
2.2	Froyo	8	10.3%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	50.6%
3.1	Honeycomb	12	0.4%
3.2		13	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1	Jelly Bean	16	5.9%
4.2		17	0.8%



Data collected during a 14-day period ending on December 3, 2012

גוגל הכריזה בכנס ההכרזה של מוטורולה (הכנס האחרון בו קיבלנו מידע רשמי, והתקיים לפני כ-4 חודשים, בספטמבר האחרון) על 480 מיליון מכשירים שאוקטבו עד היום, בכל העולם (נכון לספטמבר כמובן), בקצב (שהולך וגדל) של 1.3 מיליון מכשירים שמאוקטבים כל יום (תארו לעצמכם כמה מכשירי אנדרואיד יהיו שנה הבאה בקצב גידול שכזה).

עכשיו, בואו נחשב שניה במספרים, כדי להבין את סדרי הגודל: גרסה 1.6: 1.44 מיליון מכשירים (0.3% מ-480 מיליון). גרסה 2.1: 12.96 מיליון מכשירים. גרסה 2.2: 49.44 מיליון מכשירים. גרסאות 2.3עד 2.3.7: כמעט 244 מיליון מכשירים. ובקיצור קצת יותר מ-65% מאוכלוסיית האנדרואיד (שהם יותר מ-310 מיליון מכשירים ברחבי העולם), בעלי גרסאות ישנות.

מדובר על גרסאות שיצאו לפני ICS (שיצאה לפני שנה וחצי), ובעולם הסלולר "שנה וחצי" זה נצח! שלא לדבר על כך שלא מדובר על מערכת מוכנה ו-"בוגרת", אלא על מערכת שעדיין בבניה, ורוב הפיצ'רים שמתווספים לה הינם פיצ'רים הכרחיים וחשובים ולא מותרות אבטחה (לדוגמא זיהוי הפנים).

עדכוני אבטחה סדירים צריכים להיות בראש סדר העדיפויות של הצרכן כשהוא רוכש מכשיר חדש. מניעת העדכונים האלו על ידי היצרניות השונות (כשלרוב מדובר על טענות מוזרות וחסרות כל בסיס בגינן הן לא מעדכנת את המכשירים) הינה בעיה חמורה והצרכנים חייבים לגנותה.

אוקיי, מה זה אומר לנו?

ישנה בעיה שלא בשליטת גוגל. גוגל מעדכנת ומתקנת פרצות אבטחה, אך העדכונים לא מגיעים לצרכן הסופי. זה אומר שפרצות אבטחה שמתגלות עדיין קיימות אצל אחוז נכבד מאוד מהאוכלוסייה (וסביר להניח שיהיו קיימות עוד הרבה זמן), וזה אומר שלא חייבים לנצל דווקא פרצות שעובדות על הגרסה האחרונה, שמותקנת רק אצל 6.7% (4.1+4.2 - JB) מהאוכלוסיות האנדרואיד, גם אם נשתמש בפרצה ישנה שכבר תוקנה בגרסה החדשה, עדיין נוכל להשתמש בה על הרוב הגדול של האוכלוסייה.

אז נעלת את המכשיר ו...?

נעילת המכשיר, רובנו מבצעים את זה, האם זה בטוח? לא ממש, הרי המערכת "מקליטה" את הצורה או את הסיסמה, היא צריכה להשוות עם משהו. איפה ההקלטה הזאת יושבת? בנתיבים הבאים:

```
/data/system/gesture.key
/data/system/password.key
```

למרות שהקובץ מאובטח, לא ניתן להיכנס אליו ופשוט למשוך את הצורה או הסיסמה. אבל מה שניתן לעשות הוא פשוט למחוק את הקובץ, במידה ואין עם מה להשוות, המערכת מקבלת כל סיסמה או צורה שתכניסו.

אז מה צריך בשביל זה:

1. USB Debugging פועל, במידה והוא לא פועל ניתן להריץ את הפקודות דרך ה-Recovery (במידה ויש. במידה ואין, לא יהיה ניתן למחוק את הקובץ כשהמכשיר כבר נעול, למרות שלרוב ניתן "לדחוף" את הריקברי המפותח "בכוח", במצב של Fastboot).

2. יש לציין כי לא חשוב אם קיימות הרשאות root או לא.

הפקודה:

```
adb shell rm /data/system/gesture.key
```

או:

```
adb shell rm /data/system/password.key
```

הפקודה רצה כמובן דרך הADB, לאלה שלא מוכר להם העניין חיפוש קצר בגוגל של הערך "Android ADB" יעזור מאוד.

מה אנחנו בעצם עושים בפקודה הזאת?

rm זהו קיצור של remove. ובכך אנחנו מסירים את הקובץ gesture.key (או password.key) שנמצא בנתיב /data/system (לפעמים לאחר הרצת הפקודה יש לכבות-ולהדליק את המכשיר).

בנוסף יש לציין כי הפרצה נבדקה ועובדת על גרסאות 4.2 - 4.1.2 - 4.0.3 - 2.3.3 - 2.2 - 2.1 המהווים קצת יותר מ-95% מסך המכשירים המאוקטבים היום (וניתן להניח כי הפרצה עובדת גם על גרסאות קודמות יותר).

יש לציין כי הפרצה נוסתה על גרסת AOSP - גרסה נקייה של אנדרואיד, ואשמח לשמוע אם היא עובדת על המכשירים עם הממשקים השונים (sense של HTC, touchwiz של Samsung וכד').

גישה לכרטיס ה-SD ובעית המפתחים

לא אחת כבר כתבו על זה, כרטיס ה-SD מכיל עלינו כל כך הרבה מידע, אנחנו פשוט לא מתארים עד כמה. כרטיס ה-SD מהווה מין 'פח אשפה' של המכשיר, כל אפליקציה שאנו מתקינים יוצרת עליו תיקייה משלה ומאחסנת שם מידע (ולרוב גם לאחר הסרתה היא משאירה שם את המידע, ולא מוחקת אותו כמו שנהוג לחשוב), ולכרטיס ה-SD כל אחד יכול לגשת (כמובן, יש צורך בהרשאה לכך, אך כמעט כל אפליקציה מבקשת הרשאה לכתיבה וקריאה מכרטיס ה-SD ובלעדיה אין היא יכולה לפעול כראוי).

הבעיה היא שההרשאה ניתנת באופן מלא, או שיש גישה לכל התכנים ב-SD או שאין גישה בכלל (וכמו שאמרנו, האפליקציה לא תוכל לפעול כראוי). אז ברגע שיש לאפליקציה גישה ל-SD יש לה גישה לכל המידע עליו, אם זה תמונות, סרטונים, ושאר הדברים שנמצאים שם בצורה גלויה ולא מוצפנת, ניחא. הבעיה גדלה כשאפליקציות צד שלישי לא שומרת את המידע בצורה מוצפנת.

להלן כמה דוגמאות של המידע שאנו מאכסנים (בכוונה או ש"מאסחנים אותו בשבילנו"):

1. כל אפליקציה שמותקנת אצלנו במכשיר יוצרת תיקייה בנתיב Android/data/, וכך ניתן לקבל רשימה של כל האפליקציות המותקנות אצלנו במכשיר.
2. ניתן לראות / להעביר את כל הסרטונים, התמונות והשירים שיש אצלנו ב-SD.
3. במידה וביצענו גיבוי של אנשי הקשר ב-SD, קובץ ה-VCF חשוף לגמרי וכולם יכולים לראות את אנשי הקשר שלנו (עם כל המידע שרשום אצלנו כמו קישור לפייסבוק, כתובות מייל וכד').
4. גישה למידע של אפליקציות. דוגמאות:
   א. Dropbox שם כל אחד יכול לראות את התמונות/קבצים/מסמכים שהורדתי/העליתי ל/מהמכשיר.
   ב. SMS Backup & Restore שמגבה את כל ה-SMS-ים שלי, וכל אחד יכול לפתוח את הקובץ ב-Notepad ולקרוא את כל ה-SMS-ים שכתבתי אי פעם.

ג. Waze שומרת קובץ בשם waze_log בו קיים המידע של האפליקציה בעת ההתקנה כמו על איזה מכשיר היא הותקנה (דגם מדויק), איזה גרסת אנדרואיד (לפי API), מתי היא הותקנה, גרסת רום, איזה גרסה של האפליקציה, על איזה רשת התחברתי וכד'.

להלן תמונה של הקובץ מהמכשיר שלי, סימנתי את חלק מהדברים:

ד. Readitlater שומרת אתרים וכתבות כדי שנוכל לקרוא אותם אחר כך (ועל הדרך בצורה נוחה), איפה היא שומרת את הכתבות, בנתיב:

\Android\data\com.ideashower.readitlater.pro\files\RIL_offline\RIL_pages

שם כל הכתבות, התמונות מהכתבות והאתרים עצמם חשופים לכל, דוגמה:

ה. WhatsApp כל המידע שהעברנו באפליקציה (תמונות, סרטונים, קבצי קול) פתוח לכל. ובנוסף הגיבויים יושבים על המכשיר עצמו, בנתיב: \WhatsApp\Databases, זאת אומרת שבקלות אפשר לקחת את קבצי הגיבוי.
כמובן, שגם ניתן לפתוח את קבצי הגיבוי, הנה הדרך הקלה ביותר (שאני מכיר):
http://forum.xda-developers.com/showthread.php?t=1583021

ו. Weather1 ששומרת בקובץ שכל אחד יכול לקרוא את המיקום שבו הייתי לפי תאריך, שעה, מזג האוויר באותו מיקום וכו'.

בכוונה צורפו קישורים לעמודי האפליקציות ב-Play, כנסו ותראו שלא מדובר על אפליקציות נידחות, אלא על כאלו פופולריות מאוד עם מיליוני (ולפעמים עשרות מיליוני) הורדות. ואלה דוגמאות פשוטות מבדיקה בזיכרון החיצוני במכשיר שלי.

הנקודה היא שרוב האפליקציות צד שלישי לא מצפינות את המידע שהן מאחסנות. הפלטפורמה (אנדרואיד) מתבססת על מפתחים עם אפליקציות צד שלישי, ואנחנו אוהבים להתקין אפליקציות. הבעיה היא שהם שומרים את המידע לא מוצפן ובמקרה של כרטיס ה-SD כל אחד יכול לגשת לקבצים האלו (אם הם מוצפנים ואם הם לא, ובמידה וכן כמו ב-whatsapp, זאת לא ממש בעיה לפתוח אותם בכל זאת).

הצפנה

מגרסה 3.0 (honeycomb) ניתן לבצע הצפנה מלאה של המידע במכשיר. וההצפנה הינה טובה ואיתה אין לי שום בעיה, הבעיה היא שלסיסמה שההצפנה יש חוקים משלה.

"החוק" הבעייתי הינו שהסיסמה שההצפנה דורשת תהיה אותה אחת שפותחת את המכשיר (נעילת המסך). הנ"ל יוצר בעיה, כל פעם שאנו נצטרך לפתוח את המכשיר נצטרך לרשום את אותה הסיסמה כדי לפענח את ההצפנה?! אנחנו פותחים את המכשיר המון פעמים ביום, אף אחד לא היה רוצה לכתוב כל פעם סדרה קשה של תווים. אז מה שאנו עושים זה בוחרים בסיסמה הלא נכונה (לא אחת שתהיה מורכבת, בעלת סימנים, אותיות ומספרים) אלא בסיסמה פשוטה הקלה לזיכרון והקלדה, כדי שנוכל להקליד אותה מהר ולעבור את מסך הנעילה בזריזות.

מי שמצפין את כל המידע במכשיר שלו מבצע את זה במיוחד כדי שאם המכשיר ייפול לידים הלא נכונות, המידע שנמצא עליו לא יוסגר. בכך שאנחנו בוחרים סיסמה קלה לזיכרון ובעלת מספר מועט של תווים, אנחנו בוחרים סיסמה שקל יהיה לפצח בעזרת Brute-force attack, ואז איבדנו את הסיבה העיקרית שבשבילה הצפנו את המכשיר. במקרה הגרוע יותר, רוב האנשים משתמשים בסיסמת ה-PIN - סיסמה בעלת 4 תווים שהינם רק ספרות. וזה כבר יהיה די קל לפצח סיסמה כזאת בעזרת Brute-force attack, אך ורק עניין של קצת זמן.

הפתרון של ה"תקלה" הזאת יכול להיות מבוצע בדרך קלה ביותר, 2 סיסמאות! הסיסמה הראשונה הינה המורכבת ואחראית על פענוח המידע המוצפן. את הסיסמה הזאת נצטרך להכניס בעת הדלקת המכשיר לשם פענוח ראשוני של המידע. הסיסמה השנייה קלה יותר לכתיבה וזיכרון ותשמש לפתיחת מסך הנעילה.
הרי בסופו של דבר אנחנו מאתחלים את המכשיר פעם ב... לא בדיוק באותו יחס שבו אנו פותחים את מסך הנעילה.

> יש לציין שההצעה לפתרון ה"תקלה" הוא רעיוני בלבד ולא קיים באנדרואיד (לפחות עד שאחד עובדי אנדרואיד יעלה רעיון כזה או אחר).

החידושים של Jelly Bean בתחום האבטחה (4.1-4.2)

לבסוף, צריך גם לציין את החידושים והשינויים בגרסה החדשה (הלא היא Jelly Bean - JB 4.1 ו-4.2) שגוגל מבצעת על מנת לאבטח את המערכת.

נעילת הפנים

נתחיל מפיצ'ר האבטחה הכי מדובר שהגיע אלינו בגרסה ICS - 4.0. גוגל רצתה לאבטח את המכשיר בצורה הכי בטוחה, וזאת לדעתה על ידי אלגוריתם ניתוח פנים. הבעיה שזה לא כל כך הצליח לה, כבר באותו יום של ההכרזה עלו ל-Youtube המון סרטונים המדגימים כיצד המכשיר נפתח על ידי הצגת תמונה של הבעלים של המכשיר (והיום, בעידן הפייסבוק, לא בעיה למצוא תמונה של כל אחד).

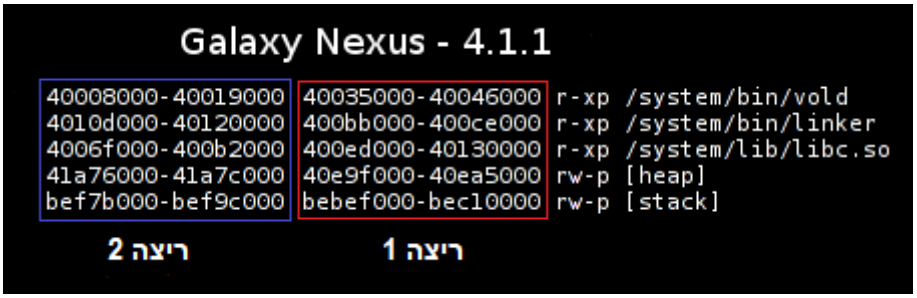
מה שגוגל עשתה כדי לתקן את הבעיה הזאת (שהופכת את הפיצ'ר ללא רלוונטי) זה להכניס מנגנון בדיקת חיים (בתרגום חופשי לעברית), מה שהם בעצם מבקשים זה פשוט לקרוץ, להראות שאתה חיי ולא תמונה, ברוב הפעמים זה תיקן את הבעיה, אך בבדיקה אישית שלי, המכשיר נפתח כמה פעמים רק על ידי הזזת התמונה (וגם במציאות על ידי הזזת הראש וללא פעולת הקריצה).
גוגל בדרך הנכונה עם הפיצ'ר הזה, אך יש להם עוד הרבה עבודה על מנת להוכיח את האמינות שלו.

ASLR

הרבה התקפות מסתמכות על יכולתו של התוקף (המתכנת) לזהות במדויק היכן תהליכים ספציפיים או פונקציות מערכת ממוקמות בזיכרון. על מנת שתוקף ינצל חור ספציפי בפונקציה או ימנף אותה לטובתו הוא הרי חייב קודם כל "להגיד" לקוד שלו היכן למצוא את הפונקציה/תהליך שיותקף/ינוצל.

ASLR (קיצור של Address Space Layout Randomization) היא שיטת אבטחה שבסיסה הוא סידור אקראי (ועצמאי) של אזור נתונים בזיכרון בכל טעינה שלו מחדש. השיטה הזאת הוטמעה בגרסה JB 4.1 באופן מלא, כך שהמיקום של הספריות, המחסנית, הערמה וגם הלינקר, כולם מסודרים בזיכרון באופן אקראי.

כמו שניתן לראות בתמונה:



בשתי הריצות של המערכת (בוצעה העלה של המערכת מחדש על ידי כיבוי והדלקה) ניתן לראות כי הקבצים לא עלו באותו מקום בזיכרון.

יש לציין כי תמיכה ב-ASLR הייתה קיימת כבר מגרסה ICS (4.0) אך בשל העדר אקראיות הלינקר (אזורי זיכרון מקשרים) הפיצ'ר הפך לחסר תועלת. ב-4.1 גוגל הוסיפו תמיכה גם בסידור אקראי של הלינקר.

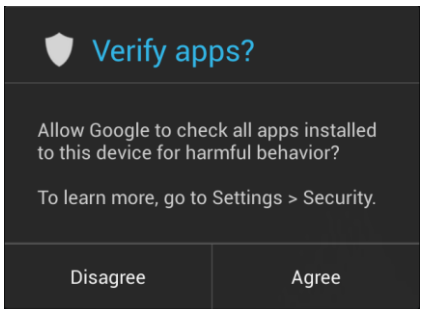
סריקת אפליקציות בזמן אמת ב-Play

בגרסה 4.2 התגלה סורק רוגלות חדש, כזה שיהפוך את החיים שלנו לקצת יותר בטוחים (יש לציין כי סורק הרוגלות החדש אינו קשור לרכישה של גוגל את חברת VirusTotal).

ההבדל בין סורק הרוגלות הנ"ל ל-Bouncer עליו כתבתי במאמר הקודם (מעבר לכך ש-Bouncer כבר קיים ופועל זמן מה), הוא שה-Bouncer הינו סורק רוגלות בצד השרת של Play, ז"א כל אפליקציה שעולה ל-Play נסרקת על ידו (הוא ממש מדמה פעולה מלאה של האפליקציה על מכשיר אנדרואיד בשרת נפרד), והסורק רוגלות הנוכחי עליו מדובר, סורק את האפליקציות שיותקנו במכשיר.

ה-Bouncer מבצע עוד פעולות, הנ"ל הוא הסבר במשפט, רק כדי להזכיר. למידע מלא יש לקרוא את המאמר הקודם.

<u>שורש הבעיה</u>



אנדרואיד ידועה בפתיחות שלה, וכתוצאה מכך אנחנו יכולים להתקין איזה אפליקציה שאנו רוצים (גם אם היא לא מה-Play). במצב כזה, שירות ה-Bouncer לא עוזר לנו, הרי האפליקציה לא נסרקה על ידיו (כי היא לא עלתה ל-Play). בדיוק במצב הזה, יקפוץ לנו חלון קטן שישאל אם ברצוננו לאפשר לגוגל לסרוק את כל האפליקציות שיותקנו במכשיר.

Hiroshi Lockheimer - סמנכ"ל ההנדסה בצוות אנדרואיד מסביר על סריקת האפליקציות במכשיר:

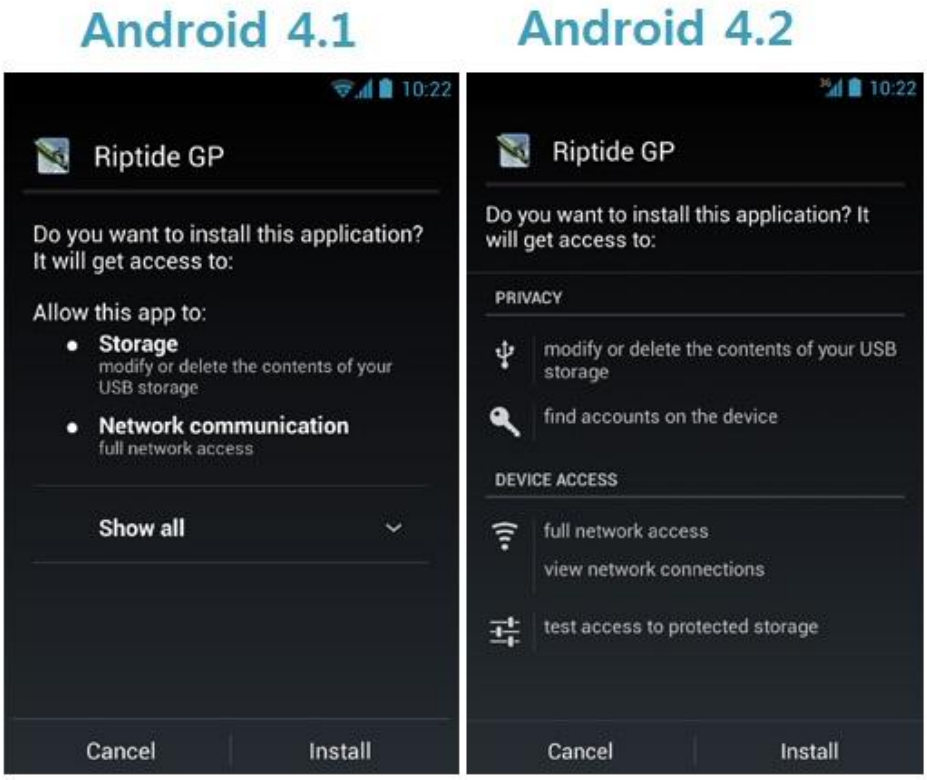
"יש לנו קטלוג של 700,000 יישומים ב-Play, ומעבר לכך, אנחנו תמיד סורקים חומר באינטרנט במונחים של רכיבי APK שמופעים. כעת, יש לנו הבנה די טובה של המערכת של האפליקציה, לא משנה אם ה-APK ב-Play או לא."

הוא ממשיך ומציין כי:

"רוב הזמן - אם האפליקציות שנתקין יזוהו כבטוחות - אף אחד לא ירגיש שהשירות קיים. השרת עושה את כל העבודה הקשה. המכשיר שולח רק חתימה של ה-APK כך שהשרת יוכל לזהות אותו במהירות. רק אם אתם נתקלים באפליקציה לא בטוחה, תהליך ההתקנה יופרע".

מסך ההרשאות

בנוסף, בגרסה 4.2 אנו מקבלים כמה שינויים וויזואליים קטנים אך חשובים מאין כמותם. מסך ההרשאות בעת התקנת אפליקציה עבר שינויים וויזואליים ולא יסתיר יותר מידע, בנוסף לכך יהיה ליד כל הרשאה אייקון קטן המתקשר לסוג ההרשאה, ובכך הופך את המעבר עליו לקליל יותר. תמונת מצב (מימין, המצב החדש (גרסה 4.2). משמאל, המצב הישן (גרסה 4.1 ומטה)):



SMS-ים בעלות

בגרסה 4.2 התווספה לה תוספת נחמדה, כעת ישנו פיצ'ר חדש הפועל מאחורי הקלעים ומתריעה כל פעם שאפליקציה מנסה לשלוח SMS שעלול לעלות לנו כסף. אם אפליקציה מנסה לשלוח SMS לקוד ידוע - כזה שבאופן אוטומטי יחייב את הספק של שולח ההודעה - המערכת מקפיצה הודעה ומתריעה על הפעולה. אנו יכולים לאשר את הפעולה ולאפשר לאפליקציה להמשיך את התהליך, או למנוע אותה. זאת כמובן גם אם האפליקציה פועלת ברקע, במצב כזה חלון ההתראה יקפוץ לנו "סתם ככה" בלי שביצענו כל פעולה.

SELinux

SELinux (ראשי תיבות: Security-Enhanced Linux) זהו סדרה של תוספים לקרנל וכלים למשתמש שניתן להוסיף להפצות השונות של לינוקס. הפרויקט עצמו התחיל לראשונה בידי ה-NSA. ב-SELinux, הקונספט של root לא קיים. מדיניות האבטחה נקבעת בידי האדמין ותקפה על כל תהליך ואובייקט הקיים במערכת, ואף אחד לא יכול לעקוף זאת.

משתמשים יכולים להרוויח גישה ברמה גבוהה בעזרת הרשאות root, אם ניתן לאפליקציות מזיקות את אותן הרשאות (root), תהיה לאפליקציה גישה להכל והנזק הפוטנציאלי שתוכל לבצע גדל בעשרות מונים. ברגע שנכיל עליה את מדיניות האבטחה (ובעצם לא יהיו לה הרשאות ה-root) היא לא תוכל לבצע כל דבר ובכך אנו מקטינים את הסיכויים לחדירה למכשיר ונזרק גדול (כי בעצם אין אפשרות לאף תהליך להימלט ממדיניות האבטחה שנקבעה).

- כמובן שגוגל לא סוגרת את אפשרות ה-root. האופציה הזאת תהיה אופציית בחירה במכשיר, וכנראה שלרוב רק ארגונים גדולים יחייבו את העובדים שלהם לבצע שימוש באותה אופציה.

- כפרויקט קוד פתוח, צוות הפיתוח של אנדרואיד מקבל כל הזמן קוד חדש ומשלב אותו במערכת, פרויקט שילוב ה-SELinux באנדרואיד (נקרא גם SE Android) החל בינואר האחרון על ידי ה-NSA (בעוד אנדרואיד מתבססת על לינוקס היא לא לינוקס, וה-NSA ביצע שינויים כדי להתאים את התוספים והכלים למערכת האנדרואיד), וניתן לראות כי גוגל אכן משלבת כל הזמן עוד חלקי קוד עם פרויקט האנדרואיד.

סיכום

מערכת האנדרואיד מתבגרת משנה לשנה ומתווספים לה המון פיצרים גם בתחום האבטחה. אך עקב האכילס שלה הינו בעיקר הפתיחות של המערכת, בכך שניתנת ליצרניות שמצידן משנות אותה ולא מספקות עדכוני אבטחה בזמן סביר ובצורה סדירה, והמפתחים שיכולים להעלות אפליקציות בלי בדיקה אחת אפילו שנעשתה על ידי גורם חיצוני ואינם מיישמים את מדיניות האבטחה (שאם נאמר את האמת גם לא כל כך קיימת).

שלא תבינו לא נכון, אני לא קורא "לסגור" את המערכת ואפילו לא קצת, אבל הנושא צריך לבוא לידי ביטוי בצורה גדולה יותר על ידי יישום מדיניות אבטחה ברורה ובאמצעות הסברים/מדריכים/סרטונים (ומה לא), כדי להפנות את צומת לב המפתחים לנושא וליישום שלו באפליקציות שלהם.

בכנס האחרון גוגל הכריזה על כלי ה-PDK שדרכו יהיו ליצרניות השונות גישה לגרסה החדשה של אנדרואיד כמה חודשים מראש, ובכך זמן העדכון אמור להתקצר (תאורטית), בינתיים זמן העדכון מתקצר רק למכשירי הדגל, בעוד שאר המכשירים מוזנחים באופן גורף. פה לדעתי האחריות היא לא על גוגל, אלא על הצרכנים להקים צעקה.

ולבסוף, מקווה שנהנתם לקרוא את המאמר לפחות כמו שנהנתי לכתוב אותו, ונתראה במאמר הבא.

על המחבר

ניר גלאון הינו סטודנט שנה שלישית לתואר בניהול ומדעי המחשב בפתוחה. מכור לספורט מוטורי, מחשבים, גאדג'טים וטכנולוגיה (בקיצור גיק מגיל קטן). חי ונושם אנדרואיד מאז גרסה 1.6 ומתעסק במערכת במסגרת עבודתו בסלקום. אוהב כל מה שקשור לקוד, אבטחת מידע ופיזיקה ותמיד שמח ללמוד עוד. ניתן ליצור קשר באמצעות כתובת האימייל nirgn975@gmail.com.