מנגנוני אבטחה באנדרואיד

מאת ניר גלאון

הקדמה

"יש המון וירוסים באנדרואיד" אמר לי חבר לפני כמה ימים, וזה גרם לי לחשוב כי רוב הציבור (הלא מבין) נוהג לחשוב על מערכות פתוחות כמערכות קלות לחדירה וכיוצא בזה כך גם חושבים על אנדרואיד.

המאמר הבא יעסוק בשאלה "האם אנדרואיד היא פלטפורמה מאובטחת או פרוצה", וכדי לענות על השאלה נתקוף את הנושא מכמה זוויות (מודל האבטחה של המערכת, מפתחים והרשאות, ממה המשתמשים צריכים לדאוג וכדו').

למה שיהיה לנו אכפת מאבטחה?

זו שאלת השאלות. לרוב המשתמשים לא אכפת מאבטחת המידע של המכשיר שלהם לפחות עד שהוא נפרץ/נגנב/נאבד, ואז אנו כבר רוצים לדעת שהמכשיר מוגן ולא שכל מי שימצא אותו יוכל לגשת למידע הנמצא עליו.

- אנדרואיד הינה הפלטפורמה הנפוצה ביותר כרגע, היא גדלה מיום ליום וכמו מערכות אחרות בעלות משתמשים רבים, קהילת ההאקרים (הלא טובים) יתמקדו באנדרואיד. הרי כלכלי יותר למצוא פרצה / לכתוב סוס טרויאני למערכת בעלת הרבה משתמשים כדי שנוכל למקסם את הניצול של מה שכתבנו.

- תאגידים גדולים משתמשים היום באנדרואיד. העובדים מחזיקים מכשיר אנדרואידי ומסנכרנים את המייל, מעבירים מצגות וכדו' (בקיצור פותחים חור בהגנה של הארגון).

- כולנו שומעים מידי יום על הרוגלות החדשות שמוצאים ~~במרקט~~ ב-PlayStore ואני בטוח שאם נכתוב בגוגל Android security נמצא המון כתבות.

יסודות האבטחה של אנדרואיד (או על מה נעבור?):

1. בידוד אפליקציות ומערכת אישורים (בעת התקנת אפליקציות):
   - האם אנחנו יכולים לשלוט על מה האפליקציות שאנו מתקינים יוכלו לעשות?
   - האם אפליקציות שהתגלו כרוגלות יכולות להשפיע על שאר המערכת?
2. "מחבר" האפליקציה:
   - האם אנחנו יכולים לסמוך על כותב האפליקציה?
   - האם אנחנו יכולים לסמוך על האפליקציות שכבר התקנו?
3. קידוד והצפנת המידע:
   - האם המידע שלנו מוגן אם המכשיר שלנו נפרץ / נגנב / נאבד?
4. מרכז הגישה למכשיר

לפני שנתחיל לעבור על הדברים אני ממליץ להכיר באופן בסיסי את הקרנל של המערכת, ניתן לקרוא בקישור הבא:

http://nirgn-startup.blogspot.com/2012/01/android.html

בידוד אפליקציות

כדי להדגים את הנושא נחבר את המכשיר למחשב (בעזרת ה-USB), ניכנס ל-ADB. נרשום PS ונראה את התהליכים הרצים. קיימים הרבה תהליכים רצים (כ-ROOT, דרך אגב), לדוגמה:

- void
- neld
- installd
- service manger
- Zygole
- system_server
- com.foo.app1
- com.bar.app2
- ועוד

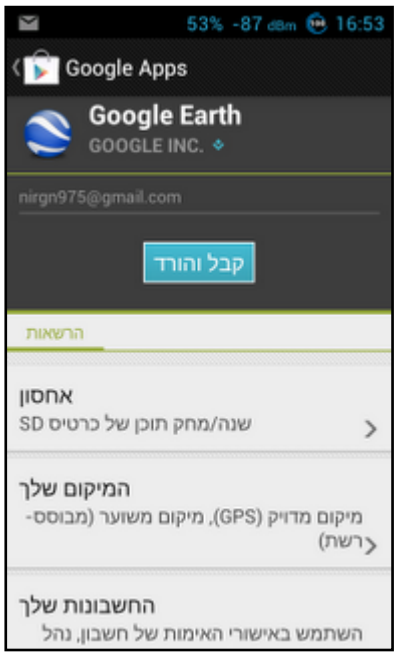
(ה-2 האחרונות הן כבר אפליקציות). ניתן לראות כי הם רצים כתהליכים נפרדים ואינם חולקים (ישירות) זיכרון / קבצי מערכת, אין להם הרשאה לקבצים אחד של השני / לרכיבים אחד של השני וגם אינם יכולים "לדבר" אחד עם השני.

הקרנל של מערכת ההפעלה (ה-"Linux Kernel") מספק את הבידוד (הריצה של כל אפליקציה כתהליך נפרד), פרט נוסף ומעניין (עליו לא נרחיב כעת), הוא שכל אפליקציה מריצה בעצמה Dalvik VM.

מה אנו לומדים מכאן?

- באופן מובנה כל אפליקציה רצה על תהליך נפרד ועם הרשאות מערכת נפרדות. הרעיון הזה אינו חדש ומבוסס על רעיון ישן ומובן שמתבצע במערכות UNIX ככה שאנדרואיד למעשה רוכבת על רעיון שנמצא כבר שנים (ועובד) עליו אנחנו יכולים לסמוך.
- שירותי מערכת (Framework) גם רצים כתהליכים נפרדים (system_server).
- הקרנל של מערכת ההפעלה הוא למעשה הבסיס של בידוד האפליקציות, או בניסוח קצת יותר מקצועי: App Sandbox.
- Dalvik לא מהווה תפקיד (בשונה מ-javaME בו האפליקציות רצו על אותה "מכונה וירטואלית", פה כל אפליקציה רצה על מכונה וירטואלית נפרדת וכולן צריכות לעמוד באותן הרשאות לא משנה אם הקוד נכתב ב-C++ או בג'אווה).
- כל הדברים האלו כוללים גם את אפליקציות המערכת המובנות.

אז איך מפתחים בורחים מ'ארגז החול'?



הרי בפועל אנחנו לא מרגישים שיש ארגז חול, לדוגמה באפליקציית פייסבוק אנו מושכים תמונה מהגלריה, או כשאנו עושים צק'-אין אנו משתמשים באפליקציית ה-GPS. אז רגע, יש ארגז חול או אין ארגז חול?

ארגז החול קיים, אך ניתן להתגבר עליו ע"י בקשת הרשאות. כל פעם שאנו לוחצים על מקש ה"הורד" כדי להוריד ולהתקין אפליקציה נפתח לנו מסך ההרשאות, המסך שעליו אנו רגילים לדלג ולהקיש ישר "קבל והורד" בלי לקרוא איזה הרשאות מיוחדות אנו נותנים לאפליקציה.

ההרשאות שאנו רואים שם אלה קבוצות הרשאות. לדוגמה, רוב האפליקציות מבקשות את ההרשאה "תקשורת רשת" ההרשאה הנ"ל היא קבוצת הרשאות שמכילה בתוכן את כל ההרשאות לאפליקציות

שקשורות לאינטרנט, (או יותר נכון לפתיחת שקעי אינטרנט) לדוגמה האפשרות ליצור שקעי אינטרנט דרך WIFI וגם האפשרות ליצור שקעי אינטרנט דרך האינטרנט הסלולרי.

דוגמה נוספת היא ההרשאה למצלמה (כקוד, זה נראה כך: android.permission.CAMERA), הרשאה נותנת לנו את הגישה למצלמת הסטילס, הוידאו, לאפקטים של המצלמה ולמצב פנורמה. (בקשות שנחשבות "נורמליות" נמצאות למטה תחת "הצג הכל"), דוגמאות נוספות להרשאות שאפליקציות יכולות לקבל:

ACCESS_FINE_LOCATION	לאפליקציה תנתן הגישה לנתוני המיקום של האנדרואיד
ACCESS_NETWORK_STATE	לאפליקציה תנתן הגישה לנתוני קישוריות הרשת
ACCESS_WIFI_STATE	לאפליקציה תנתן הגישה לנתוני קישוריות רשתות ה-WIFI הנמצאות בקרבת האנדרואיד.
CHANGE_WIFI_STATE	לאפליקציה תנתן הגישה לשנות את הקישוריות לרשתות ה-WIFI של האנדרואיד
BATTERY_STATS	לאפליקציה תנתן הגישה לנתוני מצב הסוללה של האנדרואיד
BRICK	לאפליקציה תנתן הגישה לנטרל את האנדרואיד לגמרי
READ_SMS	לאפליקציה תנתן הגישה לקרוא הודעות SMS שנשלחו / התקבלו באנדרואיד

לרשימה נרחבת של כל ההרשאות הקיימות ומה כל אחת מאפשרת:
http://developer.android.com/reference/android/Manifest.permission.html

השאלה המתבקשת היא "האם משתמש רגיל יכול להבחין האם אוסף הבקשות האלו בטוח או לא?"
ההיגיון של גוגל (או יותר נכון צוות האנדרואיד) הוא "בואו ניתן למשתמש הקצה את הכוח להחליט" (דבר שבא גם לידי ביטוי בכל הקשור לפתיחות המערכת), אבל כמו שאנחנו לומדים בשטח משתמשים רגילים פשוט לוחצים "הבא > הבא > הבא" בלי לקרוא.

אז מה החלופות?
החלופה נקראת "הרשאה דינמית", (שזה דרך אגב המצב ב-iOS), אבל לא הכל מושלם:

- זה יציק לנו אם כל פעם תיפתח תיבה קטנה שתבקש אישור.
- זה מקשה על מעבר בין אפליקציות (שזה אחד הדברים העיקריים ששמים עליו דגש בצוות האנדרואיד-שהמעבר יהיה כמה שיותר נקי וחלק) אם כל פעם שניכנס לאפליקציה נצטרך לתת לה הרשאות להכל מחדש.

- זה יוביל למצב של "הבא > הבא > הבא" גם אצל משתמשים שבדרך כלל כן שמים דגש על הנושא.
- אין היגיון בנושא, הרי כבר הורדנו את האפליקציה, "התחייבנו" אליה. אין שום סיבה שלא נרצה לתת לה את האישורים לפעול בצורה תקינה.

אז מה עושים?

צוות האנדרואיד יצר במרקט מערכת תגובות ודירוג. על מנת שנוכל לראות את תגובות המשתמשים ודירוג האפליקציה, דבר שיעזור למשתמשים רגילים לגבש דעה על האפליקציה טרום ההתקנה. אבל שוב, לא הכל מושלם והגענו למצב שבו המשתמשים פשוט מסתכלים על כמות ההורדות של האפליקציה ומניחים שאם הורידו אותה הרבה פעמים אז היא בטח בטוחה.

אז בסוף נגיע לקרן אור (אפשר גם להסתכל עליה אחרת) והיא שגוגל שמרה לעצמה את הזכות "להרוג" מרחוק אפליקציות שהיא מחליטה שהן מזיקות, ז"א שבמידה וכבר הורדנו אפליקציה שהתגלתה כמזיקה וירדה מ-Google Play, גוגל יכולים להרוג אותה מרחוק במכשיר שלנו וככה אנו לא צריכים להיות כל הזמן מעודכנים באפליקציות שהתגלו כמזיקות (אם אפליקציה תוסר מהמכשיר בצורה כזאת המשתמש יקבל מייל מגוגל).

אך בסופו של יום המשתמש אחראי למה שהולך אצלו במכשיר.

מפתח האפליקציה

וזה מוביל אותנו לכמה שאלות בהקשר למקור של היישום:

האם אנו יכולים לסמוך על המפתח של האפליקציה שאנו עומדים להתקין?

התשובה היא כנראה לא. אחת הדרכים שבהם אנדרואיד מתמודדת עם השאלה הזו היא שאפליקציות חייבות להיות חתומות וברגע שמתגלה אפליקציה מזיקה גוגל יכולה להסיר את כל האפליקציות מהמרקט בעלות אותה חותמת (ז"א כל האפליקציות שהגיעו מאותו המפתח, אפילו אם העלה כל אחת מחשבון מפתחים אחר).

החתימה מבוססת על אלגוריתם פרטי-ציבורי מוצפן בו בעיקרון יש לנו מפתח פרטי שמשמש אותנו לחתימה דיגיטלית של האפליקציה והמפתח הציבורי שאחראי על כתיבת התעודה מחוברים ביחד לקובץ ה-APK (החתימה נמצאת בתיקייה Meta). לא נרחיב על איך החותמת בנויה בדיוק אך לקריאה מעמיקה בנושא לחצו כאן. השורה התחתונה היא שכל אחד יכול לחתום את האפליקציה שלו גם אם היא רוגלה זדונית.

האם אנחנו יכולים לסמוך על כך שהאפליקציות שלנו עמידות לחבלה ברגע שהותקנו ואף אחד לא יוכל לשחק בקוד שלהן לאחר שהן הותקנו אצלנו במכשיר?

פה לעומת זאת התשובה היא כנראה כן, התפקיד של החתימה בנושא הזה הינו משמעותי. האפליקציות המתעדכנות (אם באופן ידני או אוטומטי) מסירות באופן אוטומטי את הגרסה הקודמת ומתקינות את הגרסה החדשה (ז"א APK חדש או אם תרצו אפליקציה חדשה לגמרי), אז מאיפה ניתן לדעת שלא החליפו את ה-APK ברוגלה כזאת או אחרת? הסיבה לכך היא החתימה, החתימה של ה-APK החדש חייבת להיות זהה לחתימה של ה-APK הישן ובכך נמנעת כמעט לחלוטין האפשרות להתקנת APK ממקור זר.

קידוד המידע

באופן דיפולטיבי כל הקבצים של האפליקציות מוגדרים כ-"פרטיים" (זאת אומרת שאפליקציות אחרות לא יכולות לתמרן או לערוך את הקבצים ישירות), למעט:

- היוצר של האפליקציה יכול ליצור קבצים שיהיו מוגדרים כ-"MODE_WORLD_READABLE" ו/או "MODE_WORLD_WRITABLE".
- אפליקציות שלהן אותה תעודה + חתימה - ז"א שנוצרו ע"י אותו היוצר (בגלל שהן רצות על אותו UID הן יכולות לחלוק קבצים משותפים).
- קבצים שאנו שמים ב-SD, שלשם כל אפליקציה יכולה לגשת (כמובן בעזרת הרשאה ספציפית).

אלו הצפנות קיימות לנו במערכת האנדרואיד (את חלקם אני מכיר יותר וחלקם פחות אך לא אסביר על כל אחת מחוסר זמן, מקום וכי לא זה הנושא שלנו. אם מישהו מתעניין בפרוטוקול / תקן ספציפי הוא יותר ממוזמן לגשת לגוגל / ויקיפדיה):

- VPN טלפוני ואינטרנטי (לאירגונים) (עם IPsec והצפנת 3DES ו-AES ומערכת אישורים-CA).
- wifi עם תקן 802.11 ופרוטוקלי אבטחה WPA/2.
- פרוטוקול OpenSSL.
- JCE (מבוסס על - Bouncy Castle) מספק לנו תמיכה בכל האלגוריתמים הנפוצים (הצפנות ציבוריות וסימטריות
- Apache HTTP (תומך ב-SSL).
- Keychain API - אפליקציות יכולות להתקין ולאחסן תעודות אבטחה של המשתמש.
- הצפנת כל הזיכרון (זמין מגרסה 3.0 ומעלה).

הצפנת כל הזיכרון:

בכדי לבצע פעולה זו ניכנס ל-"הגדרות" > "אבטחה" ב-ICS (או "מיקום ואבטחה" בגרסאות נמוכות יותר) > "הצפנה".

כמה חידודים:

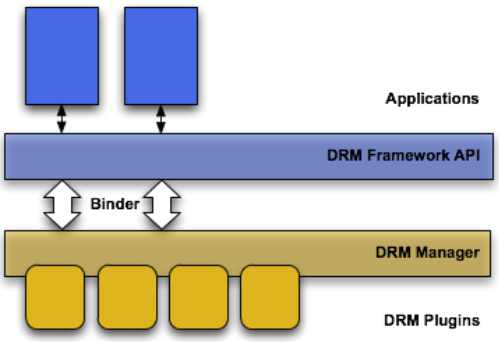
- הפעולה הנ"ל דורשת סיסמת מסך (לא סיסמת "צורה").
- הפעולה מצפינה את כל המידע שנמצא בזיכרון עם AES 128bi ו-CBC ועם SHA 256bit: ESSIV (הסיסמה משולבת בהצפנת SHA).
- ביטול ההצפנה דורשת איפוס מלא של המכשיר!
- ההצפנה מבטלת את אופציית ה-"האצת-חומרה" בקריאה וכתיבה של נתונים (ההערכות הן שפעולה זו מפחיתה בכ-54% את הביצועים של המכשיר בקריאה וכתיבה של קבצים גדולים).

החסרונות הם:

- פעולה זו אינה מונעת התקפות פיזיות, ז"א שאם גנבו לכם את המכשיר פיזית, התוקף יכול להתקין key logger על מנת לשחזר/לפענח את הסיסמה שלכם.
- פעולה זו אינה מונעת התקפות שנקראות Cold-Boot Attacks, אם מישהו יצליח להחזיק את המכשיר שלכם כשהוא דולק וכשהסיסמה כבר פועלת (אפילו אם יש נעילת מסך) הם יוכלו (בעיקרון) להוציא את הזיכרון RAM (לא אומר שזה קל אבל בהחלט אפשרי) לשים אותו במכשיר אחר, להדליק ולחלץ את המידע מה-RAM למחשב וכד'. בעיקרון כל עוד הכנסתם את הסיסמה והיא בזיכרון, ניתן לשחזר אותה.

DRM זכויות ניהול דיגיטליות.

ראשי תיבות: Digital Right Management. ב-API 11 אנדרואיד הוסיפה את האפשרות ל'זכויות ניהול דיגיטליות' - זהו ממשק המספק למכשירי OEM אישור לתוכן מסוים. הרעיון הוא להגן על התוכן של המשתמש באמצעות שימוש ברישיונות דיגיטליים לשימוש בתכנים. מה שמספק את ההגנה/החוצץ הזה הוא ה-Binder.



ה-Binder מספק תקשורת בין תהליכים, באמצעות מודל מוקטן של הקרנל. ה-Binder (כמו הפעילות הסטנדרטית של ה-IPC בלינוקס קרנל) הוא כמו שליח בין תהליכים,

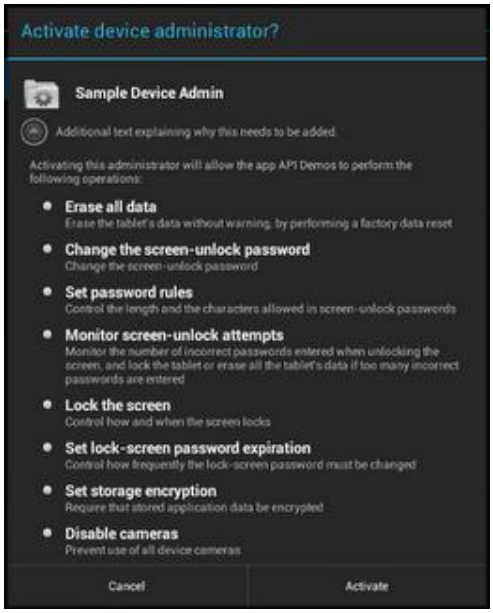
הוא יודע לזהות האם מדובר באיום ו'רץ' בין התהליכים כדי להעביר מידע (על מנת שלא תיהיה תקשורת ישירה). לקריאה מעמיקה יותר על ה-Binder ניתן לקרוא בקישור הבא:

http://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf

DRM framework API: ה-API חשוף לאפליקציות (באמצעות ה-Framework של אנדרואיד) ורץ באמצעות ה-Dalvik בשביל אפליקציות סטנדרטיות.

DRM manager: חושף את ממשק הניהול בשביל פלאגינים (נקראים גם: "סוכנים") ומבצע ניהול רישיונות ופענוח עבור התוכניות.

מרכז הגישה של המכשיר



מרכז הגישה של המכשיר כולל נעילת המסך על ידי סיסמה / קוד Pin / צורה תבנית. בנוסף בגרסה 2.2 (פרויו) התווספה לה אפשרות ה-"אדמין".

למה זה טוב? למשתמש הפשוט זה לא עוזר, לאירגונים זה הכרחי. האירגונים יכולים לשלוח לעובדים אפליקציה שמבקשת להיות מנהל המכשיר (Device Administrator) ונותנים לה שליטה לבצע הכל, החל מלהכריח את המשתמש להשתמש בסיסמה למסך הנעילה ועד נעילת המצלמה כדי שלא יוכלו להשתמש בה (אפילו נעילה של אפליקציית המצלמה בשעות מסויימת, לדוגמה בשעות העבודה, 17-8).

האפשרות הזאת נועדה להגן על המידע של הארגון הנגיש או הנמצא על המכשיר. עוד דוגמה: במידה והמכשיר נגנב הארגון יכול לשלוח הוראה למכשיר למחוק את עצמו.

סינון רוגלות

צוות אנדרואיד חשף בתחילת פברואר שירות חדש בשם Bouncer, שסורק בצורה אוטומטית את המרקט בחיפוש אחר רוגלות ותוכנות זדוניות. השירות מבצע את התהליך ברקע בלי שהמשתמשים או המפתחים מרגישים.

איך השירות עובד:

1. כשמעלים אפליקציה השירות מתחיל לנתח אותה בהשוואה לתוכנות זדוניות וסוסים טרויאנים מוכרים.
2. לאחר מכן השירות מנתח את "התנהגות" האפליקציה ומשווה אותה להתנהגות אפליקציות שנותחו בעבר והתגלו כזדוניות/סוסים טרויאנים.
3. השירות מבצע ריצה של האפליקציה בענן מיוחד של גוגל ומדמה כיצד האפליקציה תעבוד על מכשיר אנדרואיד.
4. בנוסף, השירות מבצע ניתוח של חשבונות מפתחים חדשים כדי למנוע ממפתחי אפליקציות זדוניות לחזור ולהעלות את אותם האפליקציות שוב שוב או אפליקציות חדשות.

יש לציין כי השירות קיים כבר משנת 2011 אך לא נחשף לציבור ובצוות של אנדרואיד מדווחים על ירידה של 40% בהופעת תוכנות זדוניות.

בנוסף, באותו הזמן חברות אבטחה דוגמת Lookout, AVG, Symantec וכד' דיווחו כי ישנה עליה בכמות הרוגלות המופיעות במרקט והמליצו להתקין את תוכנות האבטחה שלהם.

סיכום

צוות אנדרואיד מבצע עידכונים למנגנונים הנוכחיים כל הזמן ומכניס חדשים (דוגמת המסנן רוגלות שנכנס לאחרונה). אפשר לשים לב שנושא האבטחה נמצא על סדר היום אצל גוגל אך בסופו של יום אין מערכת החסינה לחלוטין בנושא אבטחה. כמובן שעם הייתרונות של המערכת מגיעים גם החסרונות, אנדרואיד היא מערכת הפעלה פתוחה (סוג של...) ובשל היותה כזאת, בסופו של דבר המשתמש אחראי על מכשירו ולא חשוב כמה מנגנונים יהיו, הוא עצמו החוליה החלשה במערכת.פרופורציה. שאלת הפרופורציה היא משמעותית; הפרופורציה דורשת שהפגיעה תהיה במידה שאינה עולה על הנדרש. השאלות שיעלו כאן הן האם מעבר לפריצה ומחיקה של החומר המקורי יבוצעו פעולות נוספות כמו "תג מחיר" לאותו פורץ, או האם יבוצע ניקוי ומחיקה רק של המידע של אליס; האם הפריצה תבוצע תוך פגיעה גם באחרים חפים מפשע (נניח, מחשבי זומבי בשימוש אותו פורץ) או רק במחשבים של הפורץ.