Digital Whisper
Technologic Papers

Android Fragment Injection

מאת רועי חי

הקדמה

במאמר זה אציג פגיעות חדשה שקבוצת המחקר שלי גילתה באנדרואיד. ליתר דיוק הפגיעות היא ב-Android Framework, והיא משפיעה על כל אפליקציה אשר מכילה exported Activity שיורש מ-PreferenceActivity. מכיוון שהשימוש ב-Activity זה הוא שכיח למדי, לא הופתענו לגלות מספר רב של אפליקציות פגיעות (Android Settings, Dropbox, Gmail, Evernote וכד').

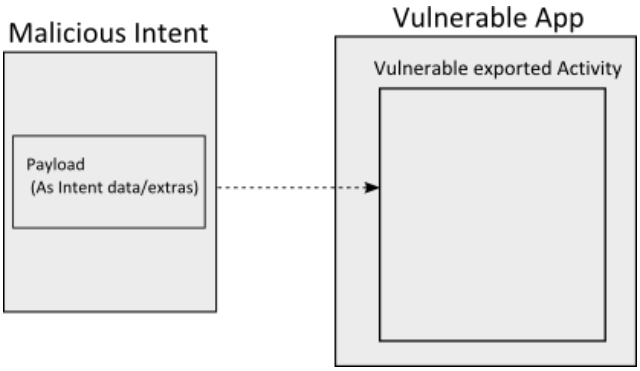
נתחיל מהקדמה קצרה על Android ועל מנגנוני ה-Security בו, נצלול לפגיעות עצמה, נתאר תקיפה אפשרית, ונקנח בתיקון.

קצת על אפליקציות Android

אפליקציות באנדרואיד בנויות ממספר סוגי רכיבים. העיקרי ואולי החשוב מהם הוא ה-Activity. כל Activity מספק מסך UI למשתמש, למשל מסך ה-Bookmarks של הדפדפן. אחת התכונות המרכזיות ב-Android היא שאפליקציה אחת יכולה להריץ (חלק) מה-Activities של אפליקציות אחרות. תכונה זו מאפשרת Feature reuse. למשל, הדפדפן מריץ את Google Play ברגע שהמשתמש גולש ל-Play Store. המימוש של מנגנון זה צריך לקחת בחשבון את העובדה שאפליקציות ב-Android רצות בסביבה מבוקרת, Sandbox. הסיבה לכך היא שהנחת היסוד, בשונה מ-PC, היא שקיים סיכוי גבוה שירוץ Malware על המכשיר, כך שמנגנון ה-Sandbox נועד למנוע מאפליקציה אחת לגשת למידע רגיש של המערכת או של אפליקציה אחרת. אפליקציות מוגבלות ע"י מספר מנגנונים. למשל, כל אפליקציה רצה עם User ID משלה, כך שקבצים אינם נגישים באופן דיפולטיבי ע"י אפליקציה אחרת. בנוסף לכך, Activities שאינם מוגדרים כ-exported (באופן מפורש או לאו) בקובץ ה-AndroidManifest שמסופק עם האפליקציה אינם יכולים להיות מורצים ע"י אפליקציה חיצונית.

Intents

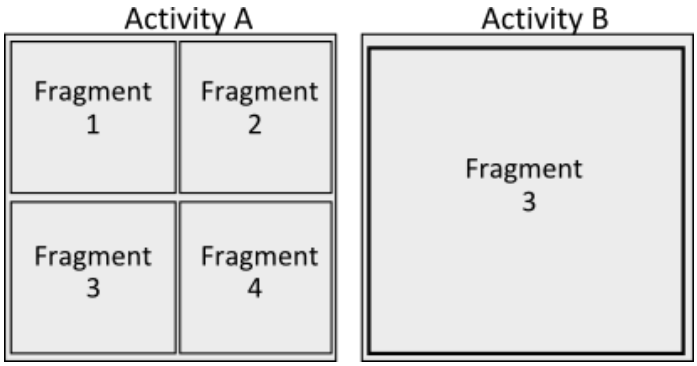
על-מנת להריץ Activity האפליקציה מייצרת אובייקט מסוג Intent ושולחת אותו ל-API המתאים (למשל Context.startActivity). אובייקט ה-Intent יכול לציין הן את Activity היעד והן את ה-Payload. האחרון נכלל במספר שדות באובייקט ה-Intent. בשדה ה-Data וכן ב-Extras שהוא שדה מסוג Bundle, מימוש אנדרואידי ל-Map. ברגע ש-Activity מוגדר כ-exported נוצר חור ב-Sandbox, כך שאפליקציה אחת יכולה לספק מידע זדוני לאפליקציה אחרת. אם המידע לא נצרך בזהירות (ע"י validation או sanitization) עלולה להיווצר בעיית Security. למשל, אם קיים פעפוע של המידע לשאילתת SQL, ללא בדיקת תקינות, האפליקציה תהיה פגיעה ל-SQL Injection. תרשים 1 מדגים את סכמת התקיפה.



[תרשים 1]

Fragments

אנדרואיד מספקת רמה נוספת של גרנולריות ב-UI: Fragments, אשר מהווים בעצם תתי Activities. הרעיון הוא שלהבדיל מ-Activity אשר מספק Feature reuse בכל המערכת, Fragments מספקים Feature reuse בתוך האפליקציה עצמה. כל מופע של Fragment משויך עם מופע יחיד של Activity מארח. הוא יכול לגשת אליו ולכן גם ל-Intent object שהריץ אותו. תרשים 2 מראה את היחס בין Activities ל-Fragments.



[תרשים 2]

מהו ה-PreferenceActivity?

Activity זה מסופק עם ה-Android Framework כך שכל אפליקציה יכולה להשתמש בו (לרשת ממנו). בעזרתו ניתן לייצר מסך הגדרות די בקלות, ולכן אפליקציות רבות עושות בו שימוש, כגון Settings (אפליקצית המערכת), Gmail, Dropbox ועוד. ההגדרות קשורות ל-PreferenceFragments. ה-PreferenceActivity קובע איזה Fragment להריץ ע"י Intent Extra שמסופק לו. הרצת ה-Fragment מבוצעת באופן דינמי ע"י Java Reflection, בתוך הפונקציה הסטטית Fragment.instantiate.

הקוד הבא מכיל את שרשרת הקריאות מתוך PreferenceActivity, מרגע יצירת ה-Activity (onCreate) עד לקריאה ל-Fragment.instantiate:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

   String initialFragment =
getIntent().getStringExtra(EXTRA_SHOW_FRAGMENT);
   Bundle initialArguments =
getIntent().getBundleExtra(EXTRA_SHOW_FRAGMENT_ARGUMENTS);

   if (savedInstanceState != null) {

   } else {
       if (initialFragment != null && mSinglePane) {
           // If we are just showing a fragment, we want to run in
           // new fragment mode, but don't need to compute and show
           // the headers.
           switchToHeader(initialFragment, initialArguments);

        } else {

           if (mHeaders.size() > 0) {
               if (!mSinglePane) {
                   if (initialFragment == null) {

                   } else {
                      switchToHeader(initialFragment, initialArguments);
                   }
               }
           }
       }

public void switchToHeader(String fragmentName, Bundle args) {
   setSelectedHeader(null);
   switchToHeaderInner(fragmentName, args, 0);
}

private void switchToHeaderInner(String fragmentName, Bundle args, int
direction) {
   getFragmentManager().popBackStack(BACK_STACK_PREFS,
```

```
FragmentManager.POP_BACK_STACK_INCLUSIVE);
   Fragment f = Fragment.instantiate(this, fragmentName, args);
   FragmentTransaction transaction =
getFragmentManager().beginTransaction();
   transaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
   transaction.replace(com.android.internal.R.id.prefs, f);
   transaction.commitAllowingStateLoss();
}
```

והמימוש של Fragment.instantiate באנדרואיד 4.3 הוא:

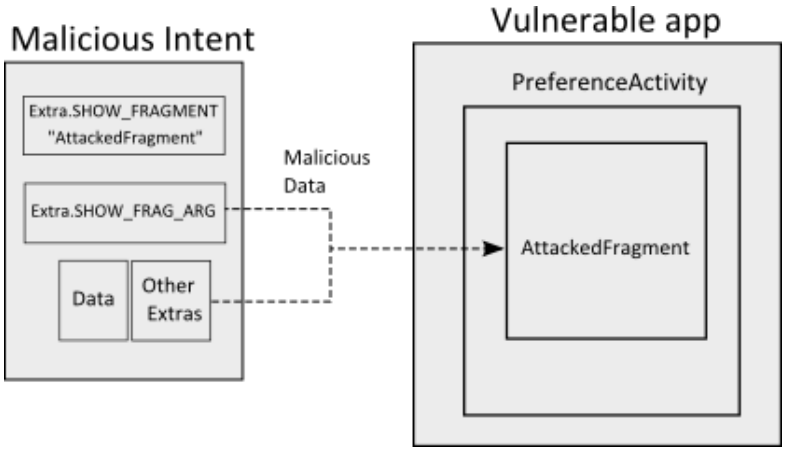
```
public static Fragment instantiate(Context context, String fname, Bundle
args) {
    try {
        Class<?> clazz = sClassMap.get(fname);
        if (clazz == null) {
  // Class not found in the cache, see if it's real, and try to add it
            clazz = context.getClassLoader().loadClass(fname);
            sClassMap.put(fname, clazz);
        }
        Fragment f = (Fragment)clazz.newInstance();
        if (args != null) {
            args.setClassLoader(f.getClass().getClassLoader());
            f.mArguments = args;
        }
        return f;
    }
...
}
```

הפגיעות

כל אפליקציה המכילה exported Activity שיורש מ-PreferneceActivity יכולה להיות מותקפת ע"י אפליקציה זדונית, עקב אי-בדיקת תקינות הקלט במנגנון הרצת ה-Fragments.

אפליקציה זדונית יכולה להריץ את ה-PreferenceActivtiy ולהחליט איזה Fragment הוא יריץ (ע"י שימוש ב-"android:show_fragment:" Intent extra). במאמר המלא תיארנו שתי דרכי תקיפה אפשריות, אחת מהן היא למצוא Fragment באפליקציה הנתקפת שמשויך ל-non-exported Activity. בדרך זו ה-Fragment בעצם נחשף ע"י האפליקציה הזדונית, שיכולה לספק לו גם מידע. מכיוון שה-Fragment רץ בדרך כלל בתוך Activity שהוא non-exported, קיים סיכוי סביר שהוא יסמוך על הקלט, דבר העלול לעורר בעיית Security.

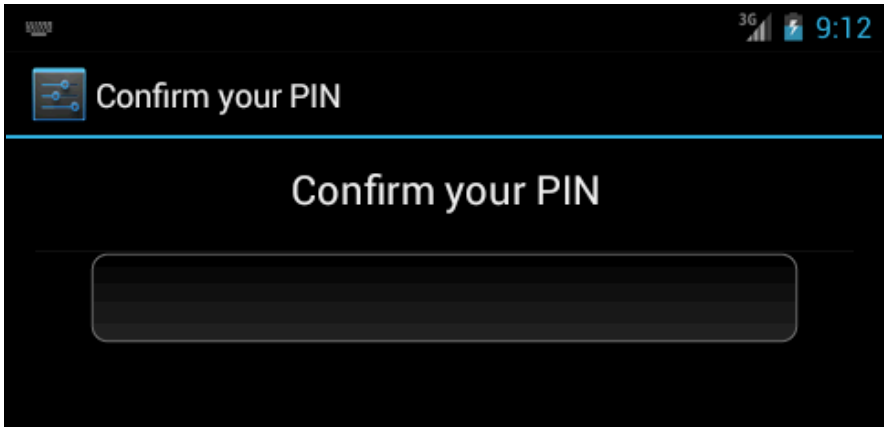
למעשה הפגיעות מעניקה יכולת לאפליקציה זדונית לשתול Fragment מסוים שחי בעולם אוטופי בו הקלט תמיד בטוח, בתוך עולם מסוכן בו לאפליקציה זדונית יש יכולת להשפיע על הקלט. איור 3 מדגים את התקיפה.



[תרשים 3]

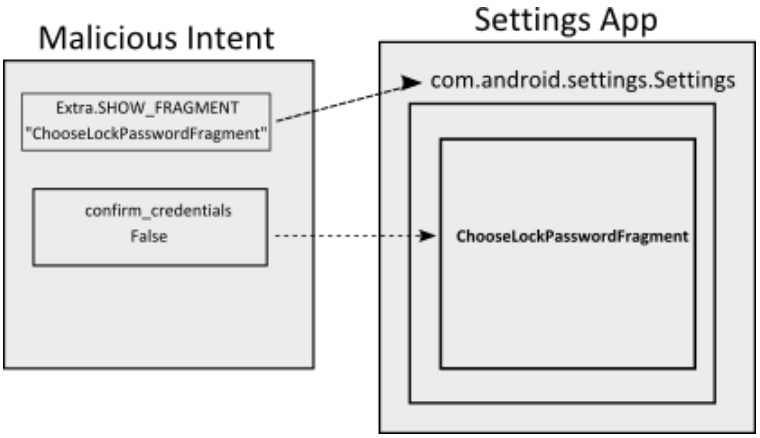
תקיפה לדוגמא: אפליקציית ה-Settings

בחרנו להתמקד באפליקציה זו (אשר פגיעה מכיוון שהיא מקיימת את התנאים המוזכרים בסעיף הקודם) כי היא מצויה בכל מכשיר אנדרואיד, ויש לה הרשאות רבות. למשל, היא יכולה לשנות את הסיסמא (או ה-PIN) של מסך הנעילה. פונקציה זו ממומשת תחת ChooseLockPassword$ChooseLockPasswordFragment. כאשר טוענים Fragment זה הוא מבקש מהמשתמש להכניס את הסיסמא הנוכחית, אלא אם מסופק ל-Activity המארח (ChooseLockPassword) פרמטר בשם "confirm_credentials" (תחת Intent extra). אם ערכו של פרמטר זה הוא false, אז ה-Fragment לא מבקש להכניס את הסיסמא (או ה-PIN). תרשים 4 מכיל צילום מסך של ה-Fragment כאשר לא מסופק הפרמטר.

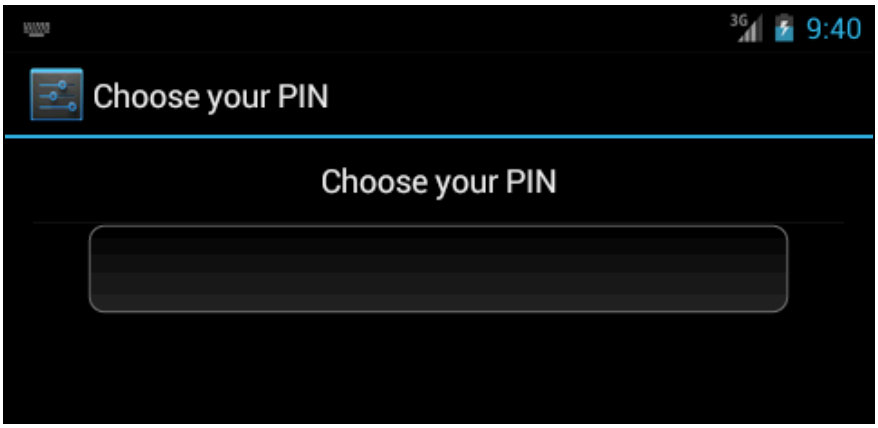


[תרשים 4]

בעזרת הפגיעות אפליקציה זדונית יכולה לטעון את ה-Fragment הנ"ל לתוך exported Activity (Settings), ולספק לו מידע ע"י Intent extras. כך היא יכולה לשלוט ב-"confirm_credentials" ולהגדיר אותו כ-false! המתקפה על ה-Settings מתוארת בתרשים 5, ותוצאתה בתרשים 6.



[תרשים 5]



[תרשים 6]

למעשה בעזרת הפגיעות אפליקציה זדונית יכולה לעקוף כל מגבלה שמוגדרת על הסיסמאות (למשל אורך מינימלי). משתמש פוטנציאלי של התקיפה על Settings הוא תוקף פיזי, למשל גנב (שמעוניין לשנות את הסיסמא) או עובד ארגון (שמעוניין לעקוף את מגבלות ה-Device Administration).

התיקון

גוגל תיקנה את הפגיעות ב-Kit Kat ולמעשה היא העבירה את האחריות למפתח. כעת, עליו לדרוס את הפונקציה PreferenceActivity.isValidFragment אשר מקבלת כפרמטר את שם ה-Fragment ומחזירה true או false האם הפרמטר בטוח לטעינה ולהיפך.

המימוש הדיפולטיבי של פונקציה זו בודק את ה-target SDK version של האפליקציה, אם הוא Kit Kat ומעלה, המימוש זורק exception (ולכן אפליקציות קורסות), אחרת הוא מחזיר true. הקוד הבא מראה כיצד נעשה שימוש ב-isValidFragment בגרסה החדשה של PreferenceActivity תחת Kit Kat.

```
private void switchToHeaderInner(String fragmentName, Bundle args, int
direction) {
        getFragmentManager().popBackStack(BACK_STACK_PREFS,
                FragmentManager.POP_BACK_STACK_INCLUSIVE);
        if (!isValidFragment(fragmentName)) {
            throw new IllegalArgumentException("Invalid fragment for
this activity: "
                    + fragmentName);
        }
        Fragment f = Fragment.instantiate(this, fragmentName, args);
        FragmentTransaction transaction =
getFragmentManager().beginTransaction();

transaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        transaction.replace(com.android.internal.R.id.prefs, f);
        transaction.commitAllowingStateLoss();
}
```

לסיכום

לסיכום, אנו ממליצים להתייחס בחשדנות לכל נתון שמגיע מהמשתמש, אשר מחלחל לפונקציות רגישות, כדוגמת Fragment.instantiate. המקרה שתיארנו במאמר זה הוא מעניין במיוחד מכיוון שהקוד הוא של ה-Framework עובדה ההופכת את הפגיעות לחמורה בכמה סדרי גודל, מכיוון שכל אפליקציה אשר משתמשת בקוד זה יורשת את החולשה.