

OPCDE_2018 challenge write-up

خطوات حل التحدي

By @sasukeOurad

عرض أستاذنا المهندس محمد الدوب هذا التحدي في مجال الهندسة العكسية لتطبيقات الأندرويد. هذا الشرح يفترض استخدامك لنسخة لينكس (كالي أو غيرها) مع معرفة بسيطة بالمجال بشكل عام. بالإمكان حل هذا التحدي باستخدام نظم تشغيل مختلفة (مثل ماك ونظام التشغيل الآخر ^_^)



م. محمد الدوب
@Voulnet

Following

بسم الله نبدأ.

يلا يا جماعة الreversing.
رابط تحميل التحدي:

opcde.com/opcde2018-win- ...

كلمة السر لفتح الملف هي opcde2018

الملف عبارة عن برنامج أندرويد فيه مجموعة من الخدع لجعل التحليل أصعب، ستجد في مكان ما في الملف الكلمة المطلوبة flag، وستجد أيضا أين ترسلها

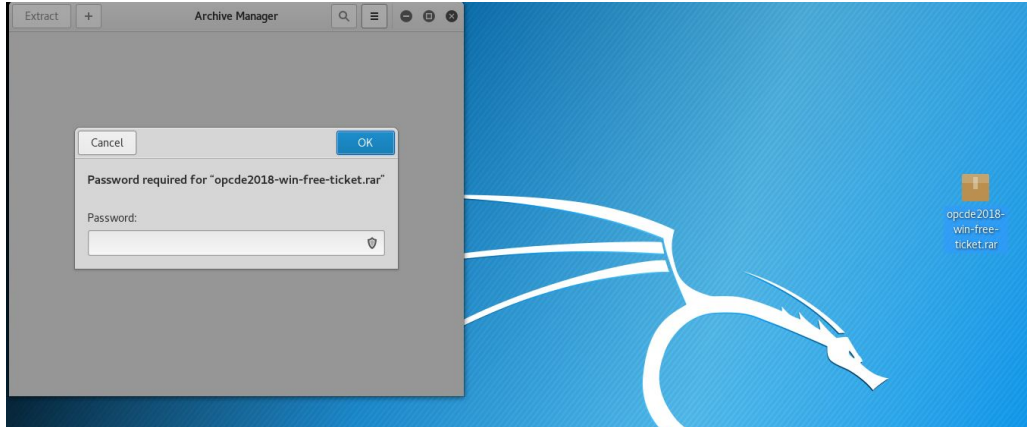
#OPCDE2018 @OPCDE

Translate from Arabic

8:56 AM - 12 Mar 2018

وجب تنبيهه أن "باسم الله" تكتب بالألف عندما لا تكون البسمة كاملة (:

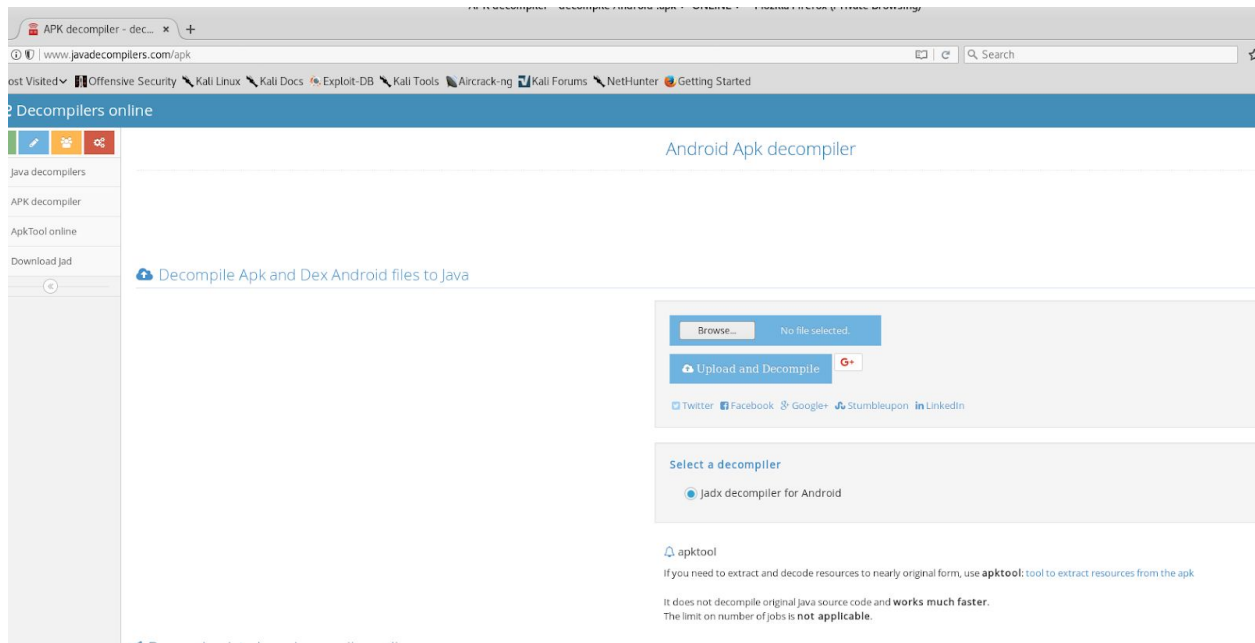
عند تحميل الملف المضغوط بصيغة rar ، بالإمكان فك الملف بالضغط عليه بالطريقة الاعتيادية. كلمة السر المطلوبة حينها هي opcde2018 كما قال أستاذنا محمد.



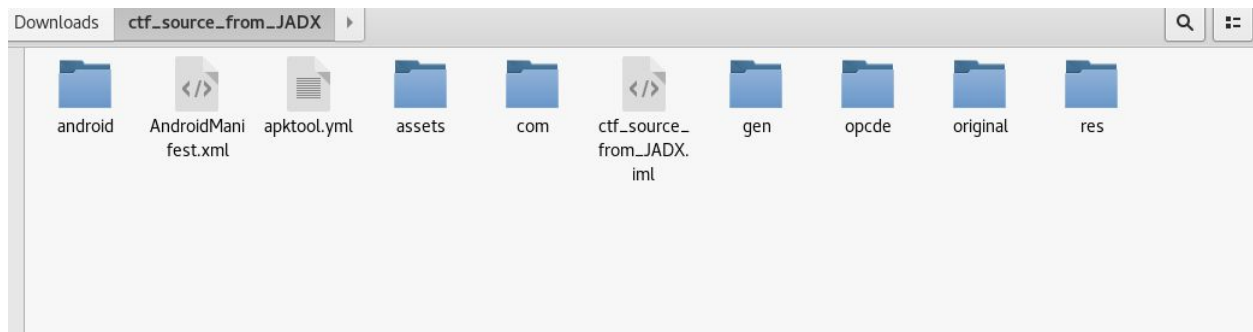
بفك الملف المضغوط، نحصل على تطبيق خاص بـ أندرويد باسم **ctf.apk**. فكرة التحدي الأساسي هي الهندسة العكسية لتطبيقات الأندرويد. إحدى ميزات تطبيقات الأندرويد لمحلل التطبيقات هي اعتمادها على لغة **java** بشكل أساسي. يسهل عكس وتحليل تطبيقات الـ **java** في العادة. لذلك تتوفر عدة طرق لاستخراج الكود الأساسي لتطبيق الأندرويد. الطريقة المستخدمة في هذا التقرير ليست الوحيدة وقد لا تكون الأفضل، ولكن تم استعمالها للسرعة.

<http://www.javadecompilers.com/apk>

هو أحد المواقع المتميزة في استخراج كود تطبيق الأندرويد. كل ما عليك فعله هو تسليم ملف الـ **apk** واستلام الكود مضغوطاً



بعد تنزيل الملف المضغوط بصيغة **zip** وفك الملف .. نتحصل على الملفات التالية الخاصة بتطبيق الأندرويد اللازم تحليله.



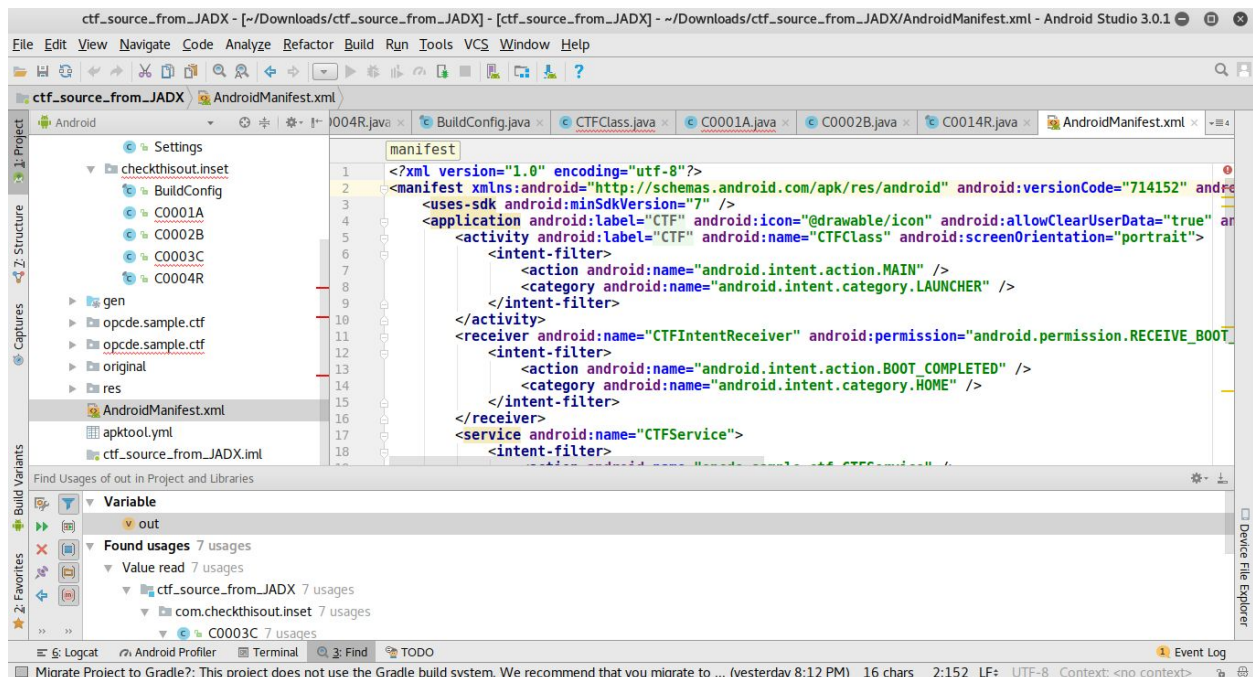
الخطوة القادمة تلتزم عمل شيين في نفس الوقت .. التحليل اليدوي والأتوماتيكي للملفات بالإضافة إلى تشغيل البرنامج في بيئة محكمة الإغلاق.

التحليل اليدوي (الساكن)

هذه الخطوة تستدعي قراءة شاملة وعامة للكود .. في رأيي المتواضع، عند التعامل مع ملفات الأندرويد يفضل فتح الملفات في برنامج يعطيك نظرة المصمم أو المبرمج لهذا البرنامج. لذلك أنصح باستخدام android studio أو أي برنامج تفضله لبرمجة تطبيقات الأندرويد. في android studio اتجه إلى

File > New > import project

واختر الملف الذي يحتوي على الكود الذي نزلناه في الخطوة السابقة
(ثم next next next) ^_^ لتحصل على شيء مشابه للصورة التالية



تبدأ رحلة البحث من ملف `AndroidManifest.xml`. يحتوي هذا الملف على تفاصيل تطبيق الأندرويد من حيث الاسم والصلاحيات ونقطة انطلاق التطبيق `Main activity`. في هذا التحدي، نقطة انطلاق التطبيق هي ملف `opcode.sample.ctf.CTFClass` ملف آخر مثير للاهتمام في هذا التحدي هو `res/values/strings.xml` بسبب احتوائه على النصوص المستخدمة في هذا التطبيق، يمكن الخروج منها ببعض المعلومات المفيدة لاحقاً. على سبيل المثال، `info_first_use` يشرح أن هذا التحدي تم اقتباسه من برنامج ضار حقيقي، ويحذر من تنزيل البرامج الضارة أو الغير موثوق بها (كلام سليم).

```

<string name="service_start">Service is starting...</string>
<string name="service_idle">It looks like i have nothing to do :-)</string>
<string name="db_no_boot_data">Oops, found no data on boot. Using default data.</string>
<string name="service_shutdown">Service finished</string>
<string name="init">First app init</string>
<string name="cache_size_error">Cache size is illegal. Using default value of 128kb.</string>
<string name="menu_info">Info</string>
<string name="menu_exit">Exit</string>
<string name="info_first_use">This CTF is based on a piece of malware, never install it on a real device! Try to find the flag!</string>
<string name="info_button">Try to find the flag!</string>
<string name="ok_label">Ok</string>
<string name="license">License</string>
<string name="cache_size_info">Sorry, this cache size is not allowed. Please use a value between 128 up to 8192 Kb.</string>
<string name="license_button">Ok</string>
<string name="no_devices">No SD-Cards found!</string>

```

نقطة أخرى ملاحظة أثناء قراءة `AndroidManifest.xml` هي وجود `android service` باسم:

`com.android.md5.Settings`

ال `android service` هو برنامج يعمل في خلفية النظام بحيث لا تتم ملاحظته من قبل المستخدم.

```

<service android:name="com.android.md5.Settings" android:enabled="true" />
</application>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>

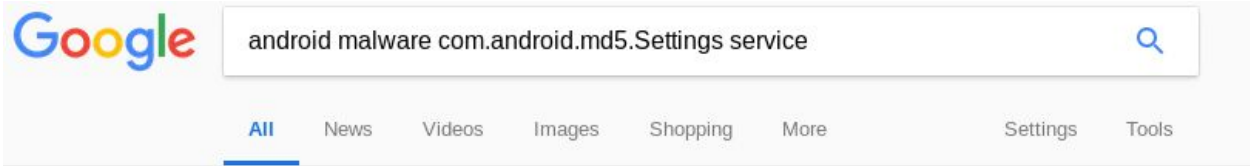
```

تشغيل البرنامج في بيئة محكمة الإغلاق.

أحد أهم وأخطر الخطوات (على المحلل أو الباحث) أثناء تحليل البرامج الضارة. كما ذكر سابقاً، لا ينصح باستخدام هذا الأسلوب إلا باتزان وحكمة لاجتناب تضرر أجهزة الباحث. من الممكن تجهيز أجهزة افتراضية باتصال مموه بالانترنت لدراسة سلوك البرنامج. في هذا التحدي، لم تكن الحاجة ملحة لاستخدام هذا الأسلوب. ولكنه المفضل لي فلم لا `^_^` أنصح باستخدام `genymotion` ك تطبيق للهاتف الافتراضي على الحاسوب، فصل الاتصال بالانترنت وربطه عن طريق `proxy` عند الحاجة ك `burpsuite`. توفر `genymotion` هواتف ب `root` مما يمنحك التحكم الكامل في الهاتف. بعد ذلك، من الضروري تنصيب `xposed` للتحليل الديناميكي للهاتف وكل ما يعمل عليه والتحكم في البرامج والنظام. يمكنك `xposed` من وضع `proxy` داخلي وإخفاء حصولك على `root` من البرامج المختلفة. أكرر، بالإمكان حل هذا التحدي بدون اللجوء لهذه التقنية. ولكن لم لا `^^` بتشغيل البرنامج، نحصل على هذا الشكل الجميل:



فور تشغيل البرنامج يقوم بطلب الحصول على أعلى الصلاحيات root. باستخدام المعلومات التي تم تجميعها إلى الآن، يمكن بيحث صغير على الموقع المفضل للجميع google العثور على البرنامج الضار الذي يحاكيه هذا التحدي:



Home / Encyclopedia / Mobile Virus / Android/Gamex.A!tr

At a glance:	
ID	5229603
Released	Jul 18, 2013
Description Updated	Jul 25, 2013
Detection Availability	
FortiGate	
Active	☐
Extended	☐
Extreme	☐
Mobile	☒
FortiClient	

Mobile Virus

Android/Gamex.A!tr

Analysis

Android/Gamex.A!tr is a piece of malware targetting Android mobile phones.

The malicious package may come disguised as applications such as SDCardBooster or Bluetooth File Transfer or GPS that require root permissions from the end user. If this permission is granted, a second package contained in the assets of the original package is installed and started. This internal package then sends the victim's IMEI and IMSI to the attacker's server. Subsequently, it downloads a list of packages to be installed on the victim's phone and finally downloads and installs each package on the victim's phone.

The malware may come in applications called "SDboost", "GPS", "Bluetooth File Transfer" etc. and may come in the packages "de.mehrmann.sdboost", "it.medieval.blueftp", "com.chartcrossd.gptest" etc. These applications start a service called **com.android.md5.Settings** which in turn uses classes in a package called **com.gamex.inset** that perform the malicious activity of the application described below.

البرنامج الضار يسمى Gamex.A!tr وطبقاً للوصف فإن صورة الشعار تحتوي على برنامج أندرويد آخر بصيغة مشفرة، يقوم البرنامج بفك الشفرة فور حصوله على صلاحيات root.

هممم لم لا نلقي نظرة على شعار التطبيق الجميل؟؟؟؟



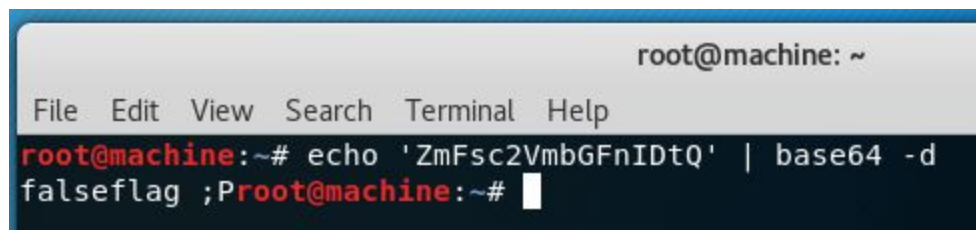
لم أجد أي شيء مختلف في الشعار 💔 ... ولكن، صورة بمسمى assets/ctf.png أثارت اهتمامي. عند محاولة فتح الصورة نفاجأ بأنها لا تعمل. بعد البحث عن وظيفتها في android studio عن طريق ضغطة اليمنى بالفأرة ثم find usages ، نجد أنه تم استخدامها في com/checkthisout/inset/C0003C.java

```
try {
    in = this.context.getAssets().open( fileName: "ctf.png");
    if (file.exists()) {
        file.delete();
    }
    file.createNewFile();
    FileOutputStream out = new FileOutputStream(file);
    try {
        byte[] buffer = new byte[1000];
        byte[] buf = new byte[1000];
        while (true) {
            int read = in.read(buffer);
            if (read == -1) {
                break;
            }
            for (int i = 0; i < read; i++) {
                buf[i] = (byte) (buffer[i] ^ 18);
            }
            out.write(buf, off: 0, read);
        }
    }
}
```

ما يقوم به هذا الجزء من البرنامج هو أخذ byte من الصورة وعمل xor مع العدد 18 وتكرار ذلك حتى تنتهي الصورة. لنقم بعمل مشابه، لنأمل الحصول على الصورة الأصلية ، عفواً سأستخدم python ^_^

ملحوظة: يحتوي com/checkthisout/inset/C0003C.java على رموز بصيغة base64 وعند تحويلها على أمل الحصول على السر نفاجأ بمزحة من مصمم التصميم <.

```
new File(this.flp).delete();
this.context.sendBroadcast(new Intent( action: "ZmFsc2VmbGFnIDtQ"));
this.context.stopService(new Intent(this.context, Settings.class));
```



```
root@machine: ~
File Edit View Search Terminal Help
root@machine:~# echo 'ZmFsc2VmbGFnIDtQ' | base64 -d
falseflag ;root@machine:~#
```

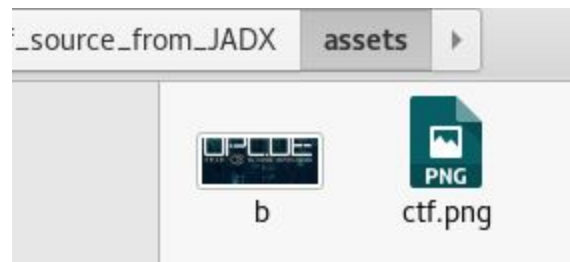
عدنا: البرنامج الصغير التالي يقوم بعمل الخطوة المذكورة مسبقاً: أخذ byte من الصورة وعمل xor مع العدد 18 وتكرار ذلك حتى تنتهي الصورة. لاحظ أن العدد 18 بنظام decimal هو نفسه 0x12 في نظام hexadecimal.

```

xor.py
~/Desktop
Open
b = bytearray(open('ctf.png', 'rb').read())
for i in range(len(b)):
    b[i] ^= 0x12
open('b', 'wb').write(b)

```

الملف الناتج سمي b إبداعاً وابتكاراً D: .. إذا كانت حساباتي صحيحة فستكون الصورة الأساسية خلف ctf.png



وهي كذلك ^_^ .. الآن وبناءً على قراءتنا السابقة للبرنامج الضار Gamex.Altr نظن بأن هذه الصورة ليست صورة عادية بل تحتوي على برنامج ضار آخر مخبأ بداخلها.

للبحث بداخل الملفات المخبأة، أنصح دائماً بالابتداء بbinwalk وهو برنامج ممتاز للبحث عن البرامج أو الملفات المدمجة داخل ملفات أخرى. نستخدم binwalk -e b لاستخراج أي ملفات أخرى متواجدة داخل الصورة. وبالفجأة ~_~

```

root@machine: ~/Downloads/ctf_source_from_JADX/assets
File Edit View Search Terminal Help
root@machine:~/Downloads/ctf_source_from_JADX/assets# binwalk -e b

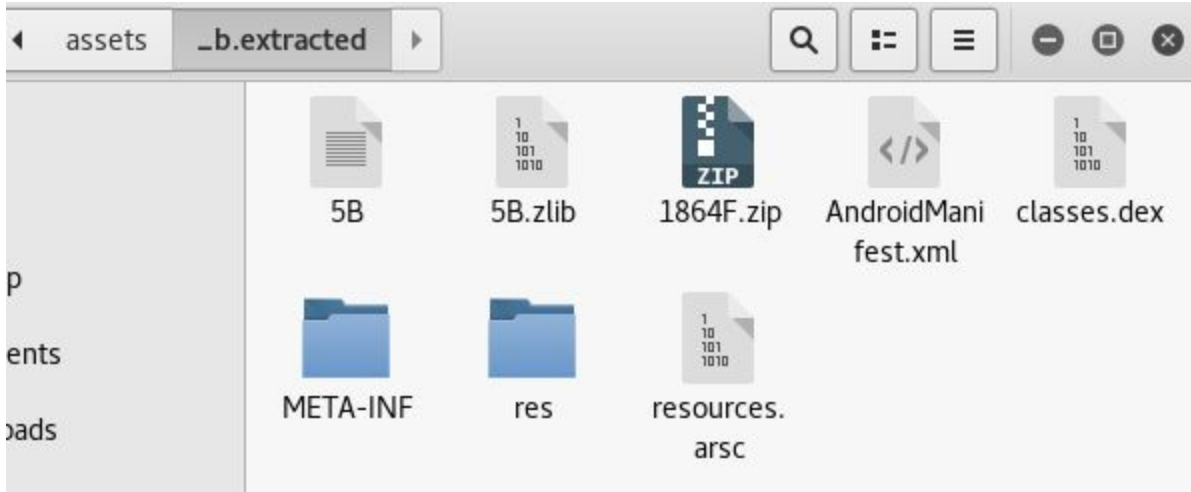
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 337 x 157, 8-bit/color RGB, non-interlaced
91	0x5B	Zlib compressed data, compressed
99919	0x1864F	Zip archive data, at least v2.0 to extract, name: META-INF/MANIFEST.MF
100294	0x187C6	Zip archive data, at least v2.0 to extract, name: META-INF/APKKEY.SF
100765	0x1899D	Zip archive data, at least v2.0 to extract, name: META-INF/APKKEY.DSA
101592	0x18CD8	Zip archive data, at least v2.0 to extract, name: AndroidManifest.xml
102771	0x19173	Zip archive data, at least v2.0 to extract, name: classes.dex
110115	0x1AE23	Zip archive data, at least v1.0 to extract, compressed size: 7301, uncompressed size: 7301, name: res/drawable/ic_launcher.png
117474	0x1CAE2	Zip archive data, at least v1.0 to extract, compressed size: 972, uncompressed size: 972, name: resources.arsc
118941	0x1D09D	End of Zip archive

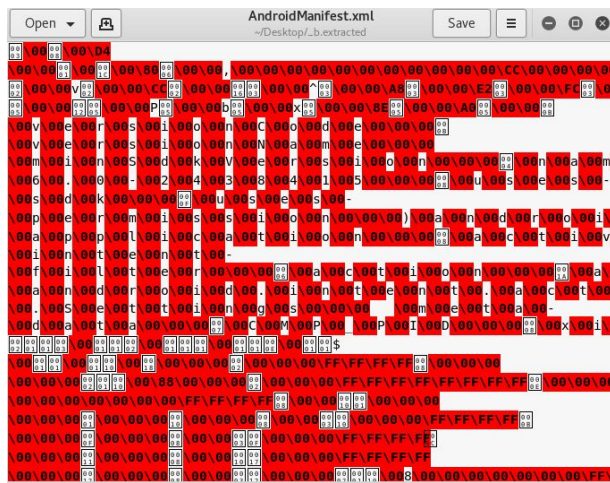
```

root@machine:~/Downloads/ctf_source_from_JADX/assets#

```



كل هذه الملفات كانت مخفية بداخل الصورة البريئة ^_^
 من النظرة المبدئية فإن أهم الملفات الموجودة هنا هي AndroidManifest.xml و classes.dex
 وجود هذين الملفين يدل على وجود تطبيق أندرويد. المشكلة هي وجودهما في صورة غير قابلة للقراءة.



لكل مشكلة حل .. لتتقية AndroidManifest.xml بالإمكان استعمال

<https://github.com/ytsutano/axmldec>

وللحصول على ملفات الأندرويد في classes.dex يمكن استعمال dex2jar لتحويل الملف إلى كودات جاهزة للقراءة ثم قراءة الملفات باستخدام android-studio كما فعلنا مسبقاً أو jd-gui. المشكلة هي أن dex2jar لم يكن موفقاً في تحويل جميع ملفات classes.dex إلى كودات قابلة للقراءة. كما يتضح في الشكل التالي، فإنه لم يوفق في التحويل وأصدر error في ملف E.class

```
SuppressLint.class x TargetApi.class x A.class x B.class x
BuildConfig.class x C.class x D.class x E.class x F.class x R.class x Settings.class x
IntentFilter localIntentFilter = new IntentFilter();
localIntentFilter.addAction("android.intent.action.PACKAGE_ADDED");
localIntentFilter.addAction("android.intent.action.PACKAGE_CHANGED");
localIntentFilter.addDataScheme("package");
paramContext.registerReceiver(this.a, localIntentFilter);
}

/* Error */
public void a(String paramString)
{
    // Byte code:
    // 0: new 78 java/lang/ProcessBuilder
    // 3: dup
    // 4: iconst_4
    // 5: anewarray 80 java/lang/String
    // 8: dup
    // 9: iconst_0
    // 10: ldc 82
    // 12: astore
    // 13: dup
    // 14: iconst_1
    // 15: ldc 84
    // 17: astore
    // 18: dup
    // 19: iconst_2
    // 20: ldc 86
    // 22: astore
    // 23: dup
    // 24: iconst_3
    // 25: aload_1
    // 26: astore
    // 27: invokespecial 89 java/lang/ProcessBuilder:<init> ([Ljava/lang/String;)V
    // 30: astore_2
    // 31: aconst_null
    // 32: astore_3
}
```

وبالتالي تمت الاستعانة بصديقنا القديم

<http://www.javadecompilers.com/apk>

الذي قام بالمهمة كما يجب.

Decompilation Results

File Name: classes.dex
Decompiler: jadx
Job status: Done.

Save

Twitter Facebook Google+ Stumbleupon LinkedIn



لنقم بفحص الملفين فحصاً أولياً، ولنبدأ بالملفات التي فشل dex2jar بقراءتها. لا شك أنها تحتوي سراً



ملحوظة أخرى: يتواجد سر مزيف آخر في هذه الملفات للقضاء على الآمال ولكن لا داعي لذكره، لنجاهله ^_^

الملف المسمى E.class عن طريق dex2jar تغيرت تسميته إلى C0005E.java وهو شيء طبيعي عند اختلاف الأدوات المستخدمة في الهندسة العكسية.

```

C0005E.java
~/Desktop/_b.extracted/crap/classes.dex_source_from_JADX/com/android/setting

L extends BroadcastReceiver {
} {

void onReceive(Context context, Intent intent) {
    if (intent.getAction().equals("android.intent.action.PACKAGE_ADDED") ||
        intent.getAction().equals("android.intent.action.PACKAGE_CHANGED")) && "com.android.update".equals(intent.getDataString().substring(0, 10).equals("com.android.update")) {
        context.sendBroadcast(new Intent("akjgikurhnfjghfkj"));
        new File(C0005E.this.f6p).delete();
        context.stopService(new Intent(context, Settings.class));
    }
}

@Override public void onReceive(Context context) {
    Context context = context;
    IntentFilter filter = new IntentFilter();
    filter.addAction("android.intent.action.PACKAGE_ADDED");
    filter.addAction("android.intent.action.PACKAGE_CHANGED");
    filter.addDataScheme("package");
    context.registerReceiver(this.f4a, filter);
}

@Override public void run() {
    try {
        Thread.sleep(1000L);
        File file = new File(this.f6p);
        if (file.exists()) {
            file.delete();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

حصلنا على هذا الرمز الجديد. فلنر إن كان السر المختبئ خلف base64

GRDDKMBUGM2DINBVGQZDIMRVGE2EMNCEGQ3TKRRUHE2UMNJXGRDDIRI=

```
root@machine: ~/Desktop/_b.extracted
```

File Edit View Search Terminal Help

```
root@machine:~/Desktop/_b.extracted# echo 'GRDDKMBUGM2DINBVGQZDIMRVGE2EMNCEGQ3TK  
RRUHE2UMNJXGRDDIRI=' | base64 -d  
0) [REDACTED] root@machine:~/Desktop/_b.extracted#  
root@machine:~/Desktop/_b.extracted#
```

بیس سو ی رموز بلا معنی < ما الحل .. هل وصلنا باباً مسدوداً ؟؟

هو كذلك .. إن ظننت أن العالم لا يحتوي على شيء سوى base64 .. ألم تسأل نفسك كيف وصلنا إلى base64 ؟ ألم نمر بتقنيات ذات رقم أصغر ؟ أليس من الأولى الخروج من الصندوق وتجربة base32 ؟؟



بتجربة base32 نكتشف الآتي:

```
root@machine:~/Desktop/_b.extracted#  
root@machine:~/Desktop/_b.extracted# echo 'GRDDKMBUGM2DINBVGQZDIMRVGE2EMNCEGQ3TK  
RRUHE2UMNJXGRDDIRI=' | base32 -d  
4F504344454242514F4D475F495F574F4Eroot@machine:~/Desktop/_b.extracted#  
root@machine:~/Desktop/_b.extracted#  
root@machine:~/Desktop/_b.extracted#
```

أهاااا

4F504344454242514F4D475F495F574F4E

ألا يبدو ذلك كنظام ترميز hexadecimal ؟
فلنر .. لنحوه إلى حروف ASCII

```
root@machine: ~/Desktop/_b.extracted
File Edit View Search Terminal Help
root@machine:~/Desktop/_b.extracted# python
Python 2.7.14+ (default, Feb 6 2018, 19:12:18)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> bytearray.fromhex("4F504344454242514F4D475F495F574F4E").decode()
u'OPCDEBBQOMG_I_WON'
>>> █
```



وهو كذلك **OPCDEBBQOMG_I_WON**
حصلنا على السر بحمد الله ^_^ شكراً أستاذنا محمد على هذا التحدي الجميل ^_^

طبعاً لم تنتهي الحكاية هنا كما ظننت .. أستاذنا محمد شرح ووضح أن الفائز سيجد رابطاً يدخل فيه اسمه وإيميله وحسابه وهذا الملف:



@Voulnet · Mar 12 م. محمد الدوب

للفوز تحتاج طبعاً لإرسال الكلمة المطلوبة (العلم flag) بالمكان الذي ستجده في التحدي،
و تحتاج إدخال اسمك وإيميلك وحسابك بتويتر وترفع ملف pdf أو txt تشرح بشكل وافى
فيه طريقة الحل بالعربية حتى ننشرها للجميع!

لكن بتسرعي المعهود نسيت وأضعت وقتاً ثميناً في الاحتفال. ثم عدت خائب الوفاض أبحث عن الرابط..
أولاً انجذبت للكود المسمى getUrl لأن "الكتاب يبين من العنوان" ولكن الصورة "logo.png" ليست موجودة من الأساس

```
private String getUrl() {
    String url = "";
    try {
        InputStream is = getAssets().open("logo.png");
        int size = is.available();
        byte[] buffer = new byte[size];
        byte[] outBuffer = new byte[size];
        is.read(buffer);
        for (int i = 0; i < size; i++) {
            outBuffer[i] = (byte) (buffer[i] ^ PASS[i % PASS.length]);
        }
        url = new String(outBuffer);
    } catch (Exception e) {
    }
    return mllc(url);
}

public String mllc(String d) {
    int i;
    byte[] a = d.getBytes();
    for (i = 0; i < d.length() / 2; i += 2) {
        byte t = a[i];
        a[i] = a[(d.length() - 1) - i];
        a[(d.length() - 1) - i] = t;
    }
    String[] s = new String(a).split(" ");
    String dx = "";
```

ثم حاولت إعادة تصميم الكود بـ java ... ولكن منذ متى طاوعتنا هذه اللغة 💔 .. بل أثرتنا بالاعتراض والexceptions.

