SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ApkAnalyser

Valerio Franković

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

Zagreb, 12, 2014

Sadržaj

[1. Uvod 1](#_Toc406953238)

[2. Grafičko sučelje 2](#_Toc406953239)

[2.1. Meni 3](#_Toc406953240)

[2.2. Glavna alatna traka 4](#_Toc406953241)

[2.3. Prozor knjižnica 5](#_Toc406953242)

[2.4. Prozor razreda i resursa 6](#_Toc406953243)

[2.5. Prozor pregleda 7](#_Toc406953244)

[3. Primjer analize zloćudnog programa 9](#_Toc406953245)

[3.1. Zadatak 9](#_Toc406953246)

[3.2. Instalacija programa 9](#_Toc406953247)

[3.3. Analiza programa 13](#_Toc406953248)

[3.4. Zaključak analize 23](#_Toc406953249)

[4. Zaključak 24](#_Toc406953250)

[5. Literatura 25](#_Toc406953251)

# Uvod

ApkAnalyser je statični, virtualni alat za analizu koji se koristi za analizu API referenca, uvid u arhitekturu i ovisnosti programa te za disasembliranje byte kodova u Android aplikacijama. ApkAnalyser je potpuni lanac alata koji podržava modifikacije binarnog koda programa kako bi se dobilo ispise. U mogućnosti je ponovno napraviti *.apk* paket te instalirati i pokrenuti program te utvrditi rezultate logcatom – Androidovim sustavom za praćenje bilješki.

Neke od mogućnosti koje ApkAnalyser omogućava su sljedeće:

* Istraživanje paketa, razreda, metoda i polja
* Pretraga paketa, razreda, metoda, referenca i ulazno-izlaznog pristupa
* Disasembliranje Dalvik byte kod metoda s osvijetljenom sintaksom
* Dekodiranje Android XML datoteka s osvijetljenom sintaksom
* Prikaz UML dijagrama paketa i razreda te osvijetliti ovisnosti paketa i razreda
* Modificirati APK datoteku s predefiniranim Dalvik byte kod injekcijama
* Vidjeti logcat filterima nivoa debugiranja
* Podržati odex aplikacije i knjižnice
* Izlistati identifikacijske brojeve resursa te njihov sadržaj
* Naći nekorištene resurse po identifikacijskom broju ili datotekama
* Naći resurse referenca sistema

ApkAnalyser je razvila tvrtka Sony, a može se dohvatiti na sljedećoj stranici:

<https://github.com/sonyxperiadev/ApkAnalyser/downloads>

Dohvaćena datoteka je jedna datoteka *.jar* ekstenzije te se pokreće preko konzole izdavanjem sljedeće naredbe:

*java –jar apkanalyser-verzija-exec.jar*

, odnosno, prema posljednjoj dostupnoj verziji:

*java –jar apkanalyser-5.2-exec.jar*

# Grafičko sučelje

Grafičko sučelje se može podijeliti na 5 cjelina:

1. Meni
2. Glavna alatna traka
3. Prozor knjižnica
4. Prozor razreda i resursa
5. Prozor pregleda



*Slika 2.1. Grafičko sučelje*

## Meni

Meni omogućuje sljedeće opcije:

1. **File**

Pokretanje analize, postavljanje puteva, postavljanje postavki i izlaz iz programa

1. **View**

Pregled razreda, byte koda i prikaz grafa za lokalne pozive i pozivatelje

1. **Lookup**

Traženje labela u *.apk* paketima, pregled metoda i paketa koji pozivaju određenu metodu, pregled metoda i paketa koje određena metoda poziva, pregled lokalnih poziva, pregled lokalnih pozivatelja, traženje kontrolnih poziva, traženje prirodnih poziva i traženje pristupa poljima

1. **Modifications**

Odjava modifikacija byte koda, izvedba modifikacija byte koda, ispis ulaza i izlaza metoda s ili bez parametara, ispis dretve u trenutku ulaska ili na offsetu, priziv metode *System.gc()* pri ulazu u određenu metodu ili na offsetu, ispis stoga u trenutku ulaska u metodu ili na offsetu, ispis nove instance, ispis konstruktora i destruktora lokalnih ili svih razreda, ispis čitanja ili pisanja jednog ili svih polja, ispis ili čitanje lokalnih varijabli, ispis iznimaka *throw* i *catch*, ispis sinkronizacije ulaza ili izlaza, ispis dnevnika bilješki na offsetu te ispis poziva referenci bez ili s parametrima ili s povratnom vrijednošću

1. **Resource**

Pogled *.xml* resursa, lista resursnih identifikacijskih brojeva, pretraga referenci resursa, pretraga nekorištenih resursnih datoteka, pretraga referenci Androidovog paketa i utvrđivanje *.xml* resursa

1. **Device**

Instalacija i pokretanje *.apk* paketa, ponovno potpisivanje, instalacija i pokretanje *.apk* paketa, ponovno potpisivanje *.apk* paketa, brisanje *.apk* paketa, pregled Odex ovisnosti i otvaranje logcata

1. **Help**

Informacije o alatu

Slika 2

*Slika 2.2. Meni*

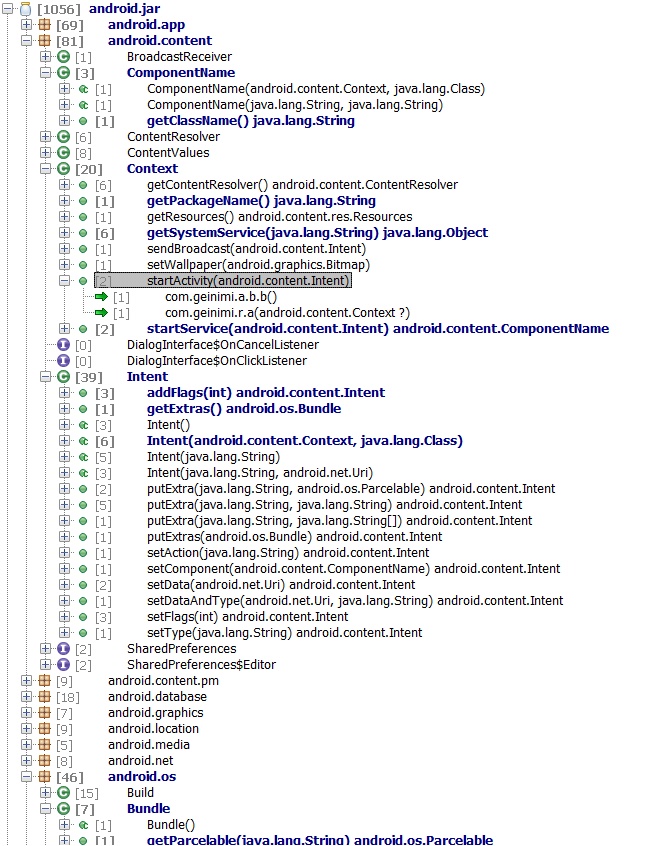
## Glavna alatna traka

|  |  |  |  |
| --- | --- | --- | --- |
| **Ikona u alatnoj traci** | **Ime stavke alatne trake** | **Pristup stavci preko menija** | **Opis** |
| Analyze | Load midlets and resolve against classpaths | File -> Analyse | Pokretanje analize *.apk* paketa |
| Tipo switch | Find references to select element in opposite tree | ---- | Pretraga knjižnica za odabranu metodu |
| Find | Show detailed information, class definition or bytecodes | View -> View Bytecodes | Prikazu metoda i varijabli u iskočnom prozoru |
| Modifikacija1 | Bytecode modify, add printout at method entry | Modifications -> Print method entry | Modificiranje byte codea da ispisuje ulaz metode |
| Modifikacija2 | Bytecode modify, add printout at method exit | Modifications -> Print method exit | Modificiranje byte codea da ispisuje izlaz metode |
| Mobitel | Unregister bytecode modifications | Modifications -> Unregisters bytecode modifications | Odjava modifikacija bytecodea |
| Pisi | Performs all registered bytecode modifications | Modifications -> Perform bytecode modifications | Izvršavanje modifikacija bytecodea |

*Tablica 2.1. Opis glavne alatne trake*

## Prozor knjižnica

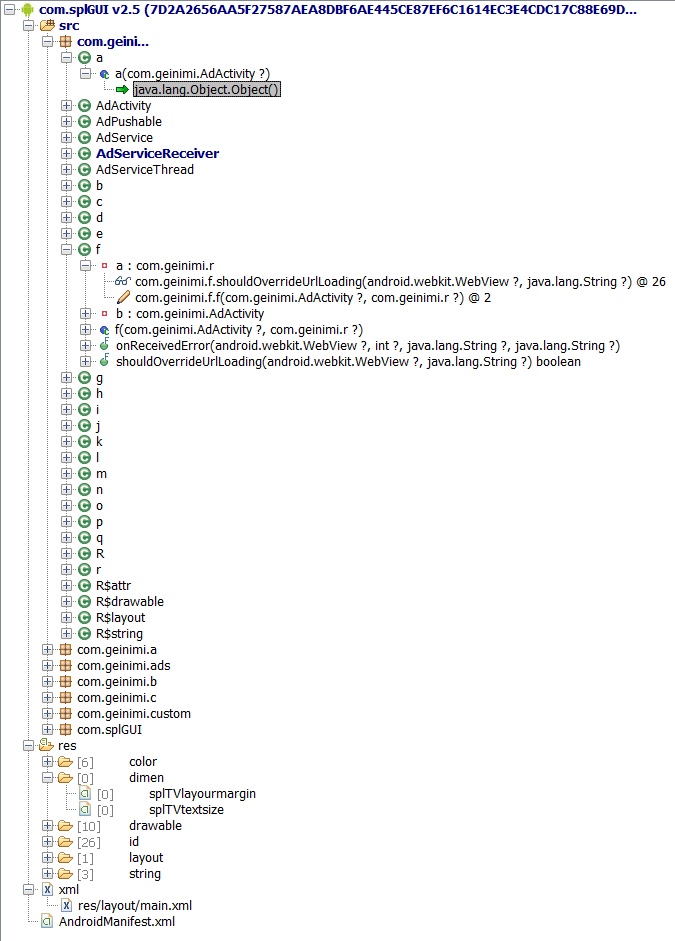
Prikazuje stablasti prikaz knjižnica s numeričkim brojačem istih.



*Slika 2.3. Prozor knjižnica*

## Prozor razreda i resursa

Stablasti prikaz razreda i resursa ciljanog *.apk* paketa. Kada se prošire polja, metode ili resursni identifikacijski brojevi, izlista se priziv metoda, pristup poljima ili reference resursa za svaku stavku.

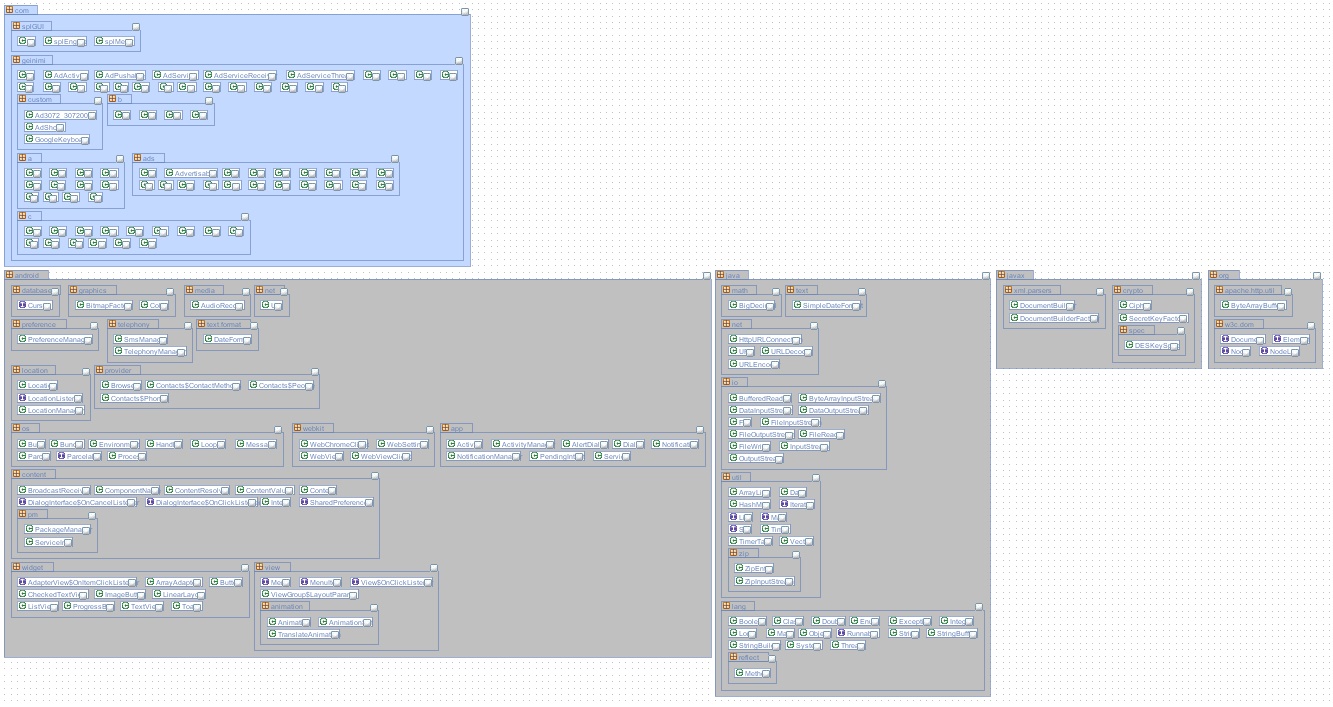


*Slika 2.4. Prikaz prozora razreda i resursa*

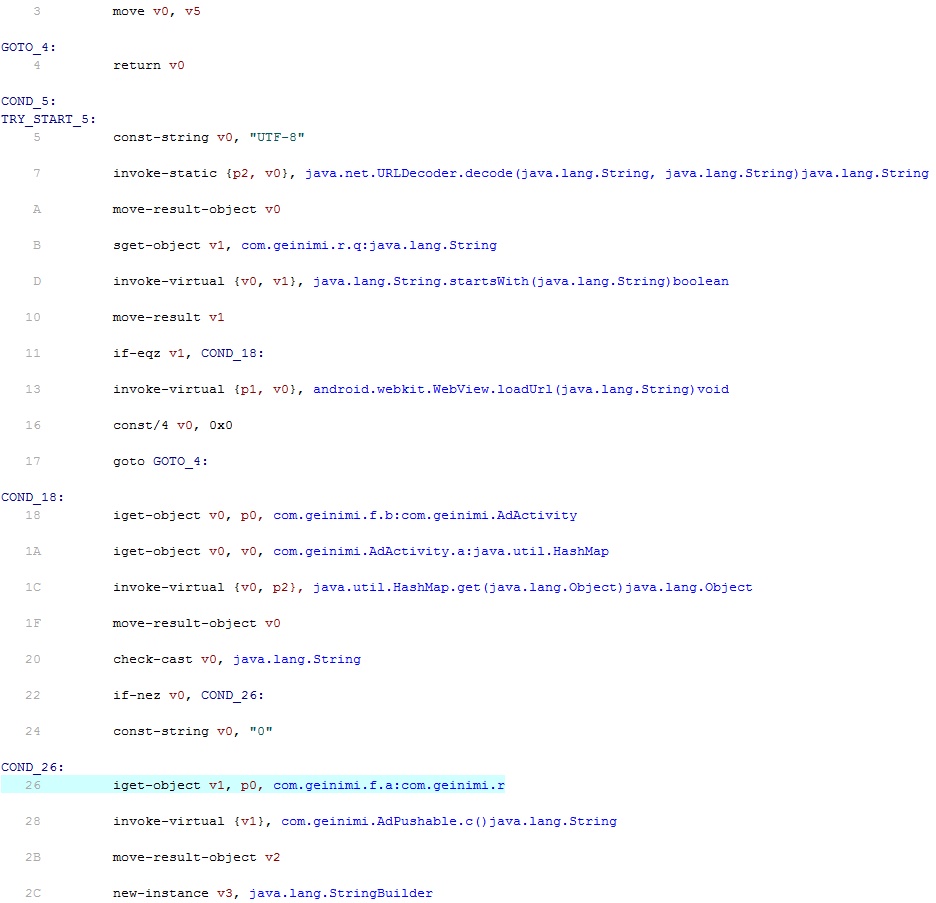
## Prozor pregleda

Prikazuje različite tipove informacija ovisno o odabranome elementu u prozoru knjižnica ili prozoru razreda i resursa. Općenito, postoje 4 vrste ispisa u prozoru pregleda:

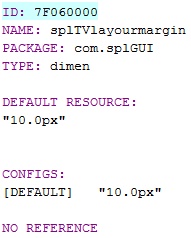
* UML dijagrami za pakete i razrede
* Dalvik disasembler za metode
* Detaljan prikaz resursa za identifikacijski broj resursa
* XML dekoder za XML resurse



*Slika 2.5. UML dijagram paketa*



*Slika 2.6. Dalvik disasembler*



*Slika 2.7. Detaljan prikaz resursa*



*Slika 2.8. XML dekoder*

# Primjer analize zloćudnog programa

U ovom primjeru napravit će se analiza zloćudnog programa FakeInst. On se „maskira“ u popularne programe kao što su Skype i Instagram, a primarno je raširen u Rusiji. Šalje SMS poruke na tzv. premium-rate brojeve telefona, odnosno brojeve za koje se plaća više nego za običan poziv. Primjeri takvih linija su linije ćaskanja za odrasle, pozivi glasanja za kandidate u televizijskim showovima, neke diplomatske usluge itd. FakeInst je jedan od prvih zloćudnih programa namijenjenih operativnom sustavu Android koji je široko otkriven u svijetu. Također je jedan od prvih porodica zloćudnih programa koje imaju nekoliko varijanta: JiFake, RuWapFraud, Opfake i DepositMobi.

Inačica zloćudnog programa se može dohvatiti na sljedećoj web stranici:

<http://contagiominidump.blogspot.com/2011/12/russian-android-malware-fake-installer.html>

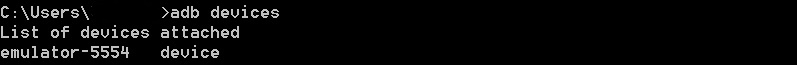
.

## Zadatak

Zadatak ove analize jest analizirati prethodno navedeni zloćudni program FakeInst te se pobliže upoznati s alatom ApkAnalyser.

## Instalacija programa

Radi bolje kompatibilnosti za rad programa preporučuje se korištenje Androidovog virtualnog uređaja (engl. Android Virtual Device - AVD) koji se nalazi u sklopu razvojnih alata Android SDK. Stvori se novi virtualni uređaj klikom na Create... te odabirom minimalnih postavki, a pažnju treba obratiti i na verziju Androida koji će se pokretati. Nakon što se stvori, virtualni uređaj se pokrene i počeka da se podigne operativni sustav. Tada se otvori konzola/command shell te se provjeri čita li alat adb taj uređaj izdavanjem naredbe *adb devices*.



*Slika 3.1. Provjera čita li alat adb virtualni uređaj*

Program se instalira naredbom *adb install <put\_do\_programa>*, a primjer instalacije je na slici 3.2.

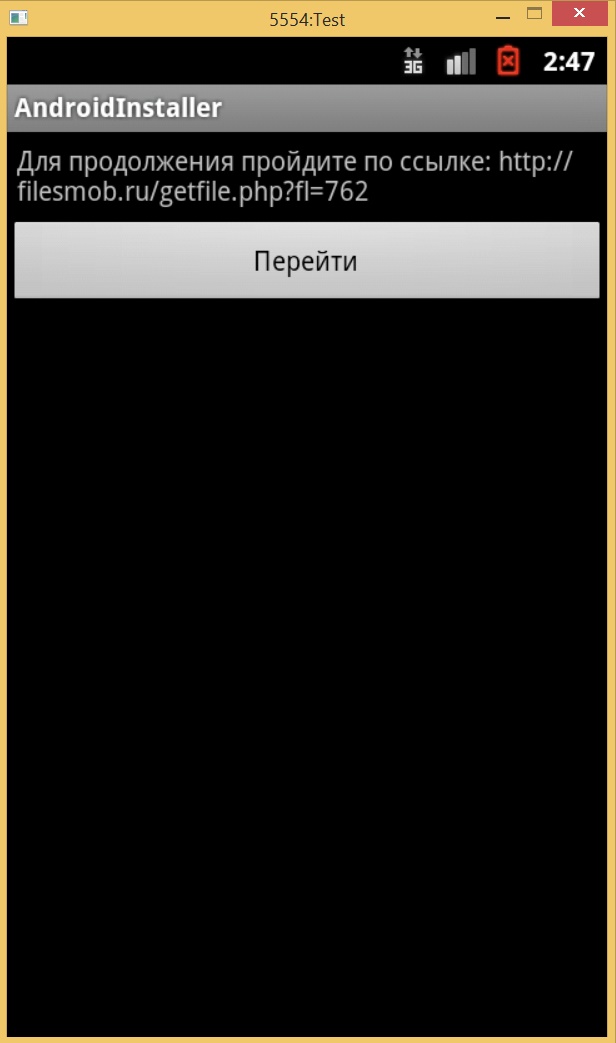
Slika 3

*Slika 3.2. adb install*

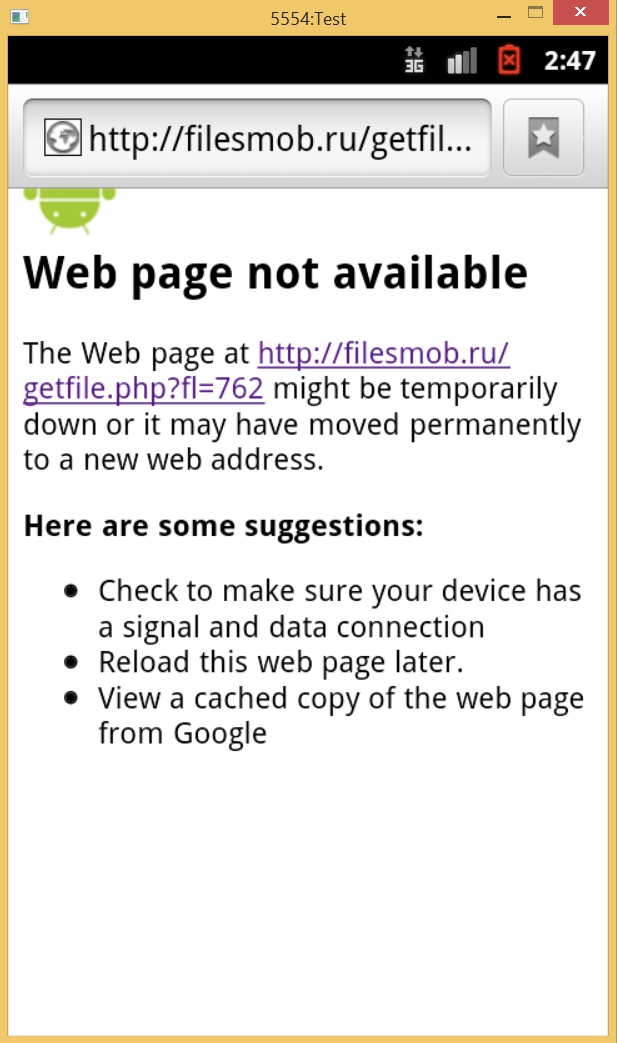
Vidljivo je da je zloćudni kod zamaskiran u program AndroidInstaller, a rad programa je slikovno opisan sljedećim slikama:



*Slika 3.3. Početni prozor programa AndroidInstaller*



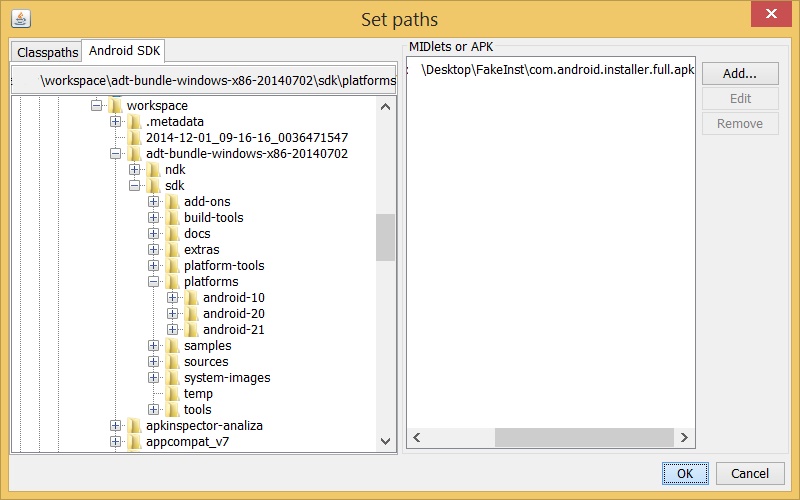
*Slika 3.4. Prozor koji se prikaže pritiskom gumba*



*Slika 3.5. Program otvori web stranice s koje bi se trebalo dohvatiti datoteku*

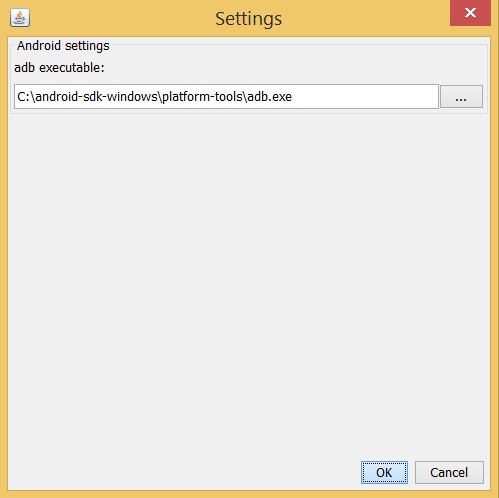
## Analiza programa

Da bi se započelo analizu, potrebno je učitati program nad kojim se ona želi sprovesti. To se uradi klikom na File u Meniju te odabirom Set Paths. Nakon toga se pojavi iskočni prozor. Na iskočnom prozoru se klikne Add... te se odabere *.apk* paket i klikne na OK.



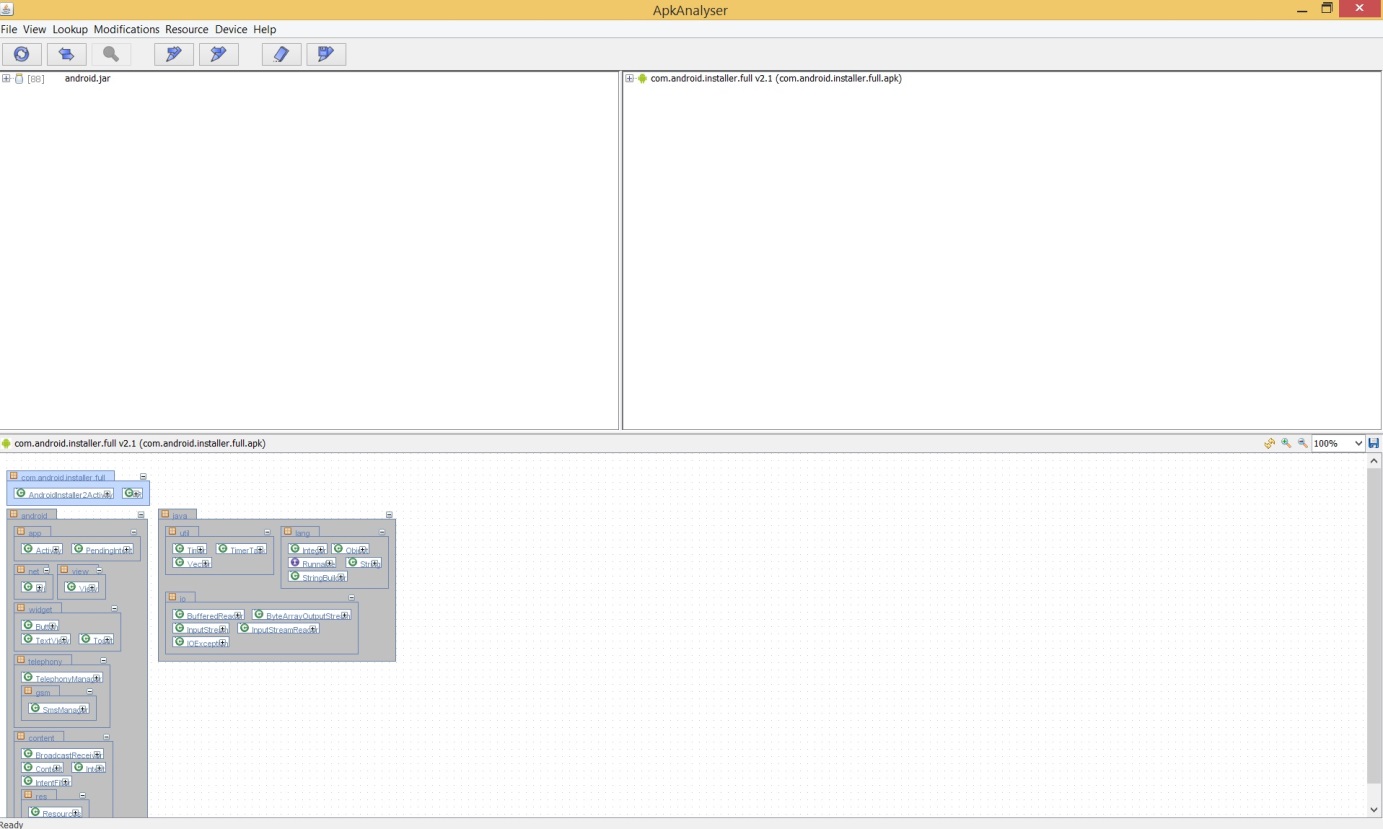
*Slika 3.6. Postavljanje puta*

Zatim se pod File pa Settings namjesti put do datoteke *adb.exe* koja se nalazi u *<SDK\_FOLDER>/platform-tools/* . Ako program ne bude htio prebacivati *.apk* datoteke na telefon, tada se problem rješava stvaranjem folder *android-sdk-windows/platform-tools/* te se tamo doda datoteka *adb.exe* pa se taj put namjesti pod Settings kao što je prikazano na slici 3.7.



*Slika 3.7. Settings – postavke*

Nakon što su se namjestile inicijalne postavke programa, može se pokrenuti analiza klikom na tipku „Load midlets and resolve against classpath“ u glavnoj alatnoj traci ili preko menija na File pa „Analyse“. Dobije se sljedeće:



*Slika 3.8.Stanje alata ApkAnalyser nakon učitavanja .apk datoteke*

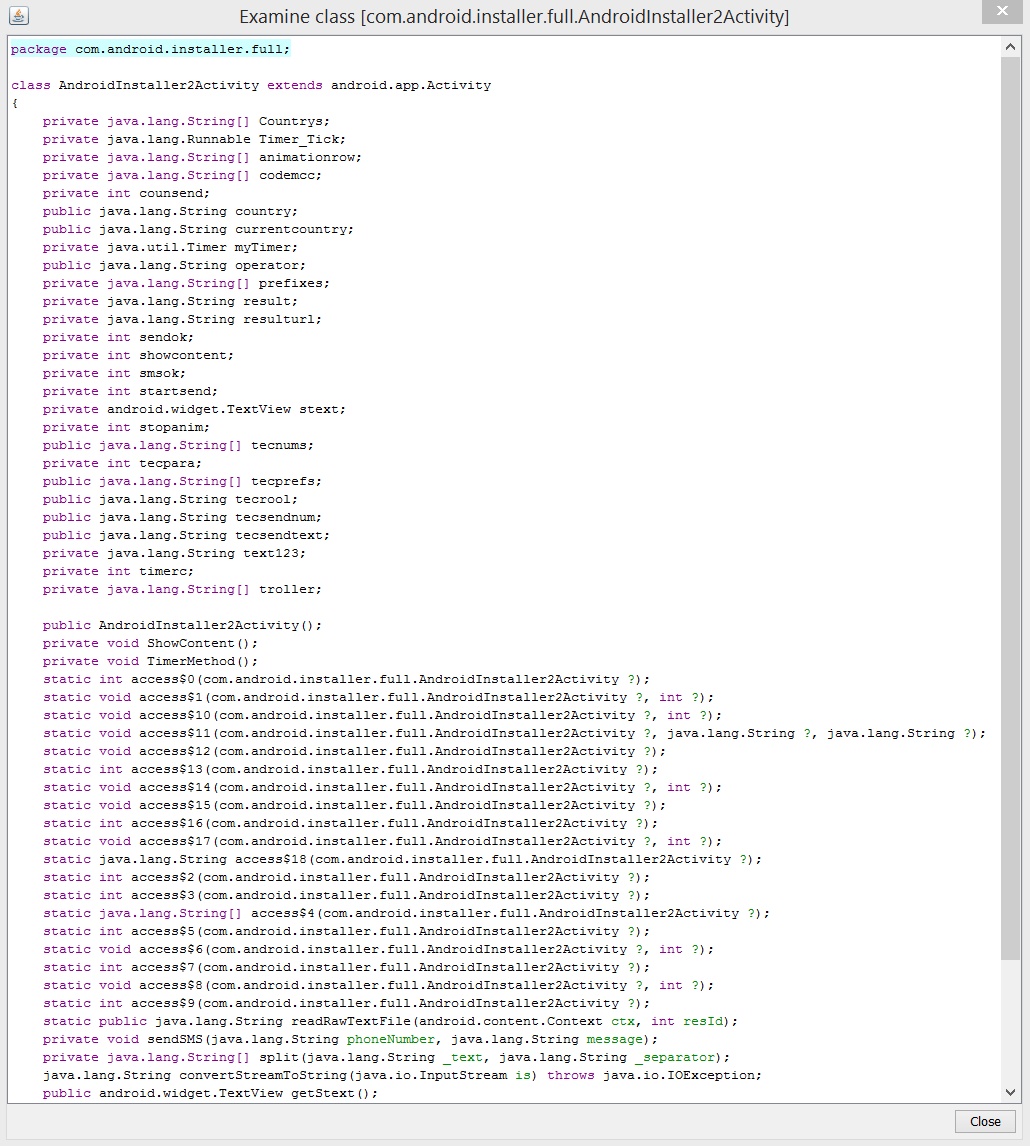
Početak analize počinje čitanjem datoteke AndroidManifest.xml te interpretacijom njene vrijednosti.



*Slika 3.9. AndroidManifest.xml*

Iz datoteke se može pročitati da koristi dopuštenja za Internet, kameru, statistike baterije, modificiranje audio postavki, instaliranje paketa, bluetooth, čitanje stanja telefona te primanje, slanje i pisanje SMS-a. Ime programa je AndroidInstaller, a glavna aktivnost je *AndroidInstaller2Activity* tako da se nju prvu krene analizirati.

Može se započeti klikanjem na razred aktivnosti u prozoru razreda i resursa te zatim odabirom na View pa na Examine class. Otvori se popis atributa i metoda te se analiziraju njihova imena da se dođe do informacija o funkcionalnosti.



*Slika 3.10. Prozor Examine Class*

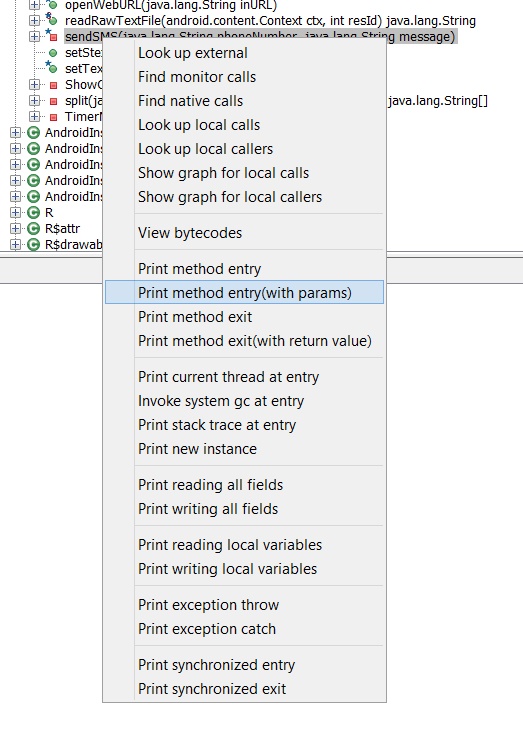
Zanimljivi atribut je *private java.lang.String text123;*

A zanimljive metode su:

* *static public java.lang.String readRawTextFile(android.content.Context ctx, int resId);*
* *private void sendSMS(java.lang.String phoneNumber, java.lang.String message);*
* *java.lang.String getStringFromRawFile(android.app.Activity activity, int ttr) throws java.io.IOException;*
* *public void openWebURL(java.lang.String inURL);*
* *public void setText123(java.lang.String text123);*

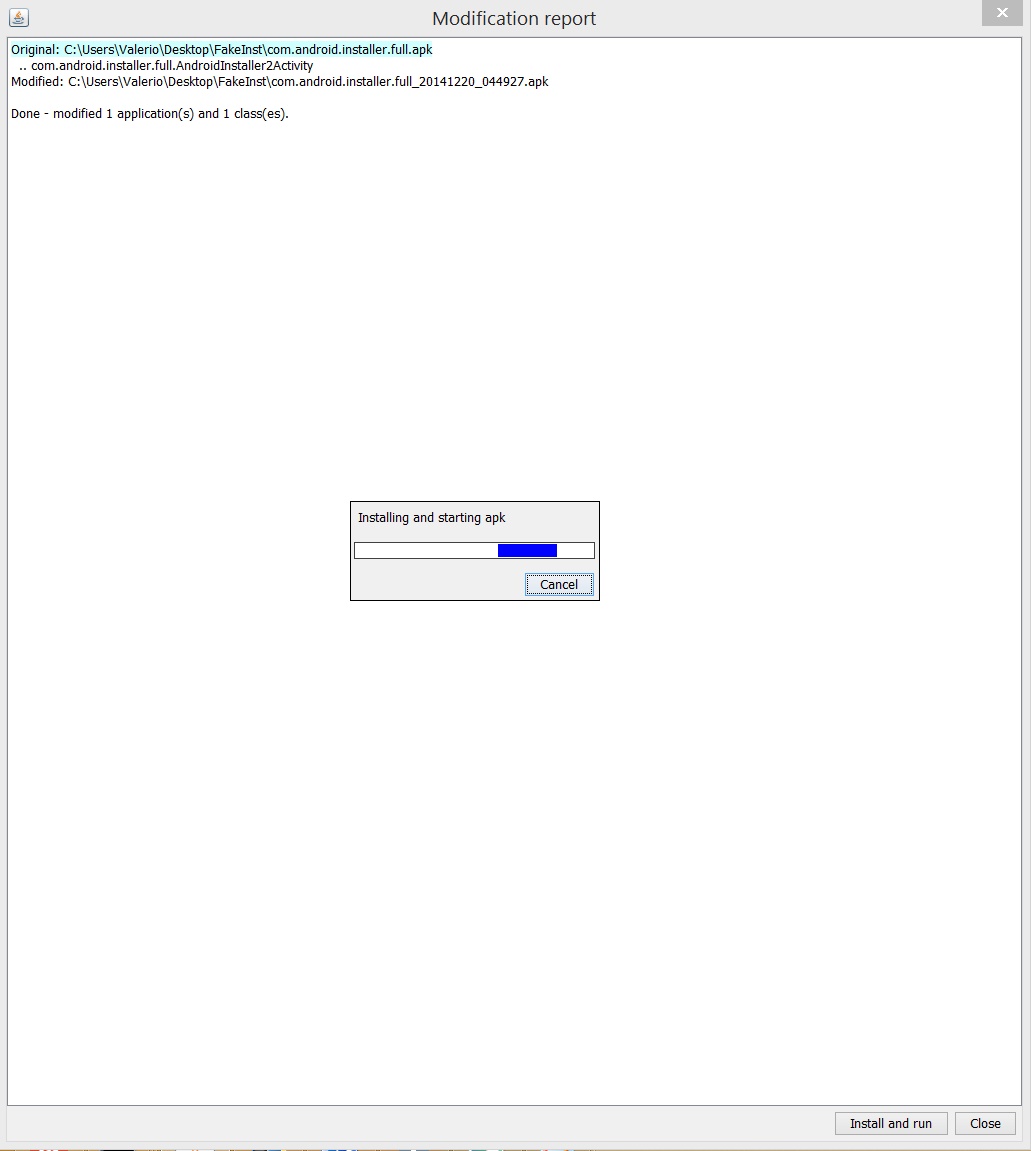
Budući da je inicijalni rad pokazao da program služi samo za dohvaćanje datoteka, navedene metode se ne čine da upadaju u program, a pogotovo metoda *sendSMS()*. Ostale datoteke upućuju da se stvara novi String, otvara neka Web lokacija ili postavlja tekst. Da bi se shvatilo što se događa za vrijeme rada programa može se napraviti ispis, primjerice pri ulazu ili izlazu iz metode. To se uradi tako da se klikne desnom tipkom miša na svaku od navedenih metoda te se odabere Print method entry (with params) ili Print method exit (with return value) ovisno dodaju li se u metodu Stringovi kao parametri ili se String vraća, a ako su ispunjena oba uvjeta, tada se može kliknuti na oboje. Npr. za metodu *sendSMS()* se odabere Print method entry (with params).

Za atribut se može odabrati Print writing this field da se vidi kada mu se mijenja vrijednost.



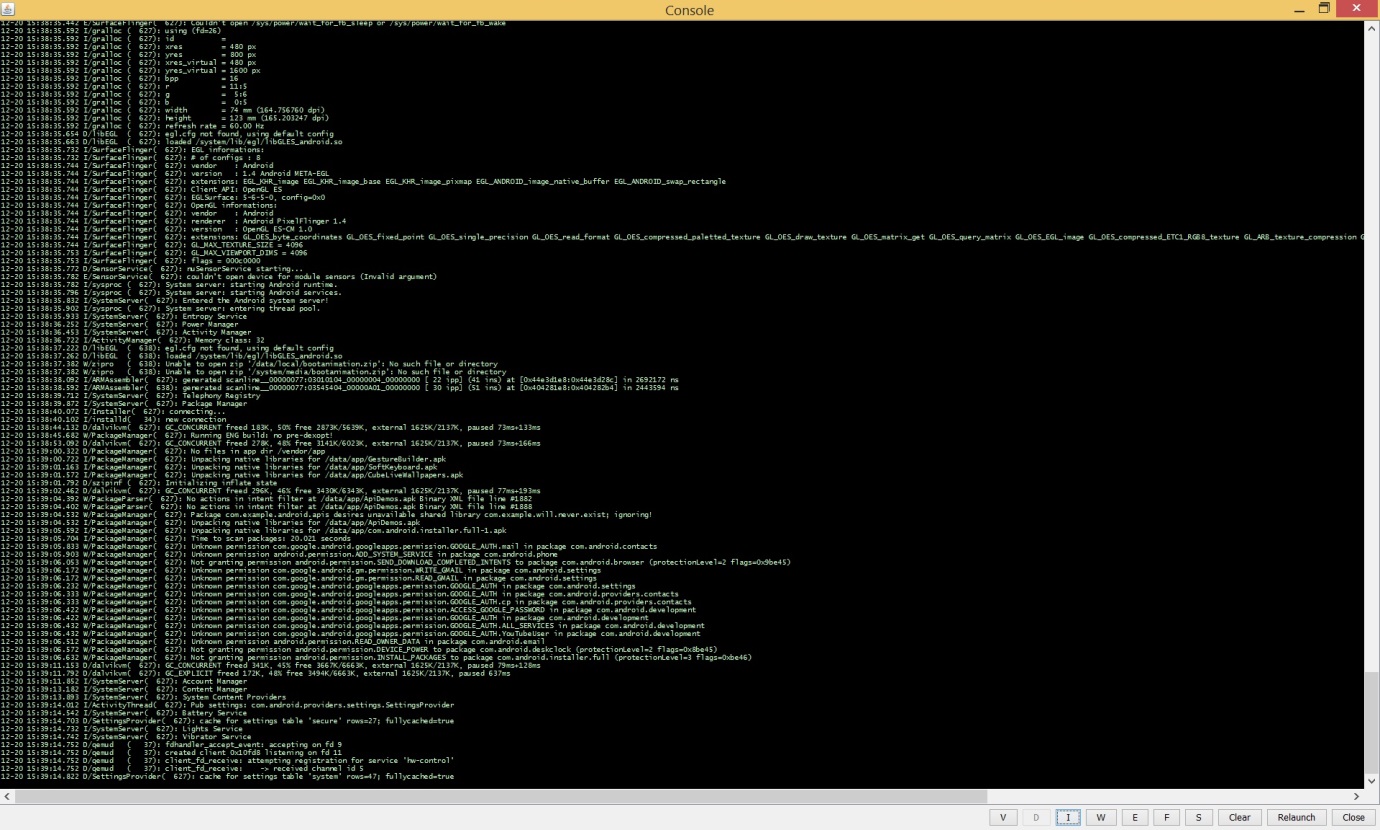
*Slika 3.11. Postavljanje ispisa metode*

Nakon postavki ispisa, treba se izvesti modifikacija bytecodea što se uradi klikom na Modifications pa na Perform bytecode modifications čime se otvara iskočni prozor s informacijama provedenim modifikacijom, a klikom na gumb „Install and run“ se taj modificirani program može izvesti na virtualnome stroju (ili pak stvarnome ako je priključen).



*Slika 3.12. Prozor s informacijama o modifikacijama te poluga napretka*

Poslije nekog vremena, program se otvori na virtualnom uređaju, a ApkAnalyser otvori prozor logcat.

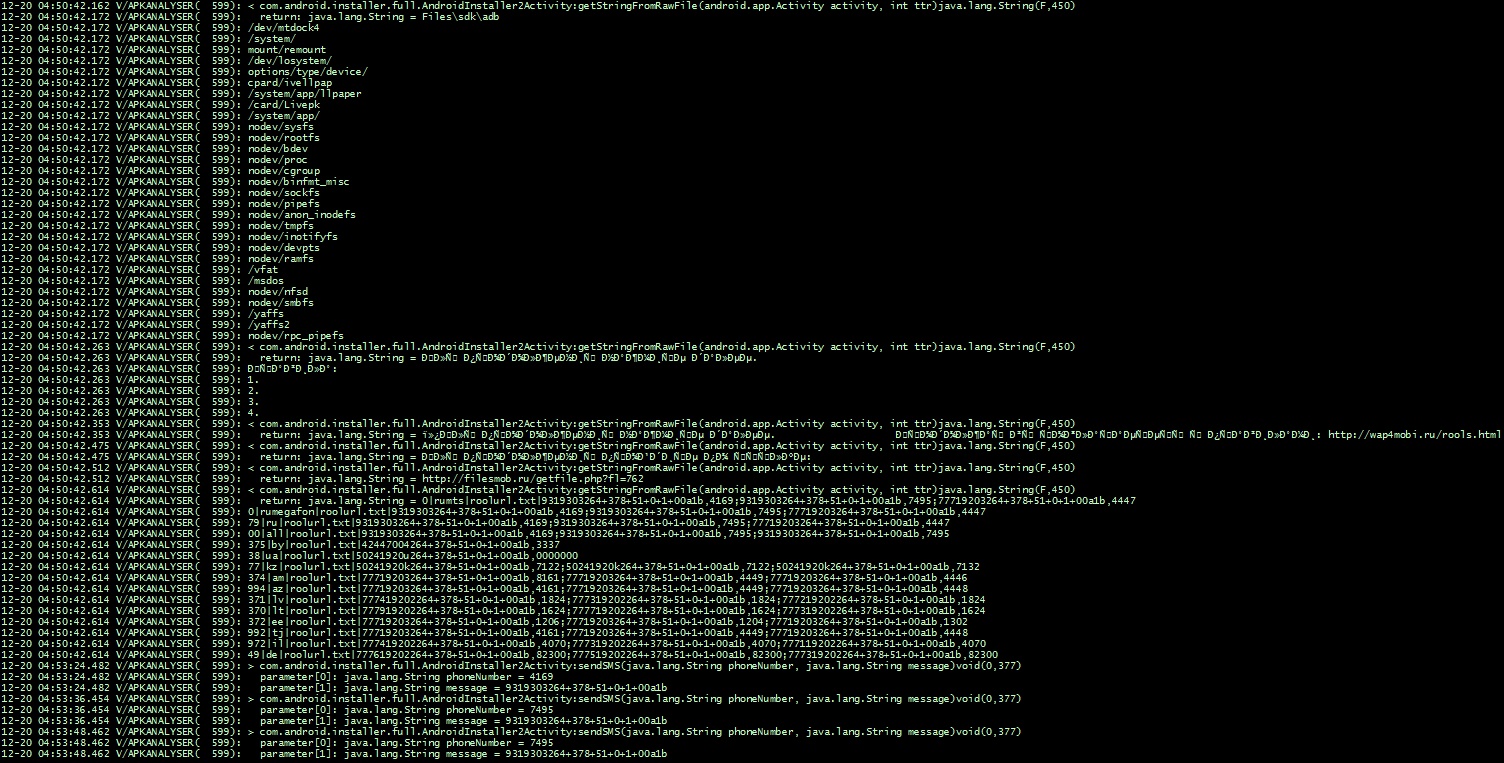


*Slika 3.13. Prozor logcat*

Logcat ispisuje poruke koje se zadaju u Log razredu te ostale sustavske poruke kao što su tragovi stogova kada su bačene iznimke. Logcat prozor ima 10 različitih gumbova, a oni su redom:

* V – Verbose (najmanji prioritet)
* D – Debug
* I – Info (zadani prioritet)
* W – Warning
* E – Error
* F – Fatal
* S – Silent (najveći prioritet)
* Clear
* Relaunch
* Close

U ovom primjeru će se gledati rezultati u prozoru koji se dobije klikom na gumb S da bi se vidjeli svi ispisi osim novopostavljenih bilješki, a ispis je sljedeći:



*Slika 3.14. Prozor Silent modea*

Sljedeći odsječak je vrlo zanimljiv:

12-20 04:53:24.482 V/APKANALYSER( 599): > com.android.installer.full.AndroidInstaller2Activity:sendSMS(java.lang.String phoneNumber, java.lang.String message)void(0,377)

12-20 04:53:24.482 V/APKANALYSER( 599): parameter[0]: java.lang.String phoneNumber = 4169

12-20 04:53:24.482 V/APKANALYSER( 599): parameter[1]: java.lang.String message = 9319303264+378+51+0+1+00a1b

12-20 04:53:36.454 V/APKANALYSER( 599): > com.android.installer.full.AndroidInstaller2Activity:sendSMS(java.lang.String phoneNumber, java.lang.String message)void(0,377)

12-20 04:53:36.454 V/APKANALYSER( 599): parameter[0]: java.lang.String phoneNumber = 7495

12-20 04:53:36.454 V/APKANALYSER( 599): parameter[1]: java.lang.String message = 9319303264+378+51+0+1+00a1b

12-20 04:53:48.462 V/APKANALYSER( 599): > com.android.installer.full.AndroidInstaller2Activity:sendSMS(java.lang.String phoneNumber, java.lang.String message)void(0,377)

12-20 04:53:48.462 V/APKANALYSER( 599): parameter[0]: java.lang.String phoneNumber = 7495

12-20 04:53:48.462 V/APKANALYSER( 599): parameter[1]: java.lang.String message = 9319303264+378+51+0+1+00a1b

Vidljivo je prilikom tri poziva metode sendSMS(), poruka se šalje na brojeve '4169' i '7495' sa sadržajem „9319303264+378+51+0+1+00a1b“. Budući da su brojevi dugački samo 4 znaka, može se zaključiti da se doista radi o premium-rate brojevima.

Iz dnevnika bilješki vrijedi još spomenuti i ispis izlaza iz metode *getStringFromRawFile()*:

12-20 04:50:42.614 V/APKANALYSER( 599): < com.android.installer.full.AndroidInstaller2Activity:getStringFromRawFile(android.app.Activity activity, int ttr)java.lang.String(F,450)

12-20 04:50:42.614 V/APKANALYSER( 599): return: java.lang.String = 0|rumts|roolurl.txt|9319303264+378+51+0+1+00a1b,4169;9319303264+378+51+0+1+00a1b,7495;77719203264+378+51+0+1+00a1b,4447

12-20 04:50:42.614 V/APKANALYSER( 599): 0|rumegafon|roolurl.txt|9319303264+378+51+0+1+00a1b,4169;9319303264+378+51+0+1+00a1b,7495;77719203264+378+51+0+1+00a1b,4447

12-20 04:50:42.614 V/APKANALYSER( 599): 79|ru|roolurl.txt|9319303264+378+51+0+1+00a1b,4169;9319303264+378+51+0+1+00a1b,7495;77719203264+378+51+0+1+00a1b,4447

12-20 04:50:42.614 V/APKANALYSER( 599): 00|all|roolurl.txt|9319303264+378+51+0+1+00a1b,4169;9319303264+378+51+0+1+00a1b,7495;9319303264+378+51+0+1+00a1b,7495

12-20 04:50:42.614 V/APKANALYSER( 599): 375|by|roolurl.txt|42447004264+378+51+0+1+00a1b,3337

12-20 04:50:42.614 V/APKANALYSER( 599): 38|ua|roolurl.txt|50241920u264+378+51+0+1+00a1b,0000000

12-20 04:50:42.614 V/APKANALYSER( 599): 77|kz|roolurl.txt|50241920k264+378+51+0+1+00a1b,7122;50241920k264+378+51+0+1+00a1b,7122;50241920k264+378+51+0+1+00a1b,7132

12-20 04:50:42.614 V/APKANALYSER( 599): 374|am|roolurl.txt|77719203264+378+51+0+1+00a1b,8161;77719203264+378+51+0+1+00a1b,4449;77719203264+378+51+0+1+00a1b,4446

12-20 04:50:42.614 V/APKANALYSER( 599): 994|az|roolurl.txt|77719203264+378+51+0+1+00a1b,4161;77719203264+378+51+0+1+00a1b,4449;77719203264+378+51+0+1+00a1b,4448

12-20 04:50:42.614 V/APKANALYSER( 599): 371|lv|roolurl.txt|777419202264+378+51+0+1+00a1b,1824;777319202264+378+51+0+1+00a1b,1824;777219202264+378+51+0+1+00a1b,1824

12-20 04:50:42.614 V/APKANALYSER( 599): 370|lt|roolurl.txt|777919202264+378+51+0+1+00a1b,1624;777719202264+378+51+0+1+00a1b,1624;777319202264+378+51+0+1+00a1b,1624

12-20 04:50:42.614 V/APKANALYSER( 599): 372|ee|roolurl.txt|77719203264+378+51+0+1+00a1b,1206;77719203264+378+51+0+1+00a1b,1204;77719203264+378+51+0+1+00a1b,1302

12-20 04:50:42.614 V/APKANALYSER( 599): 992|tj|roolurl.txt|77719203264+378+51+0+1+00a1b,4161;77719203264+378+51+0+1+00a1b,4449;77719203264+378+51+0+1+00a1b,4448

12-20 04:50:42.614 V/APKANALYSER( 599): 972|il|roolurl.txt|777419202264+378+51+0+1+00a1b,4070;777319202264+378+51+0+1+00a1b,4070;777119202264+378+51+0+1+00a1b,4070

12-20 04:50:42.614 V/APKANALYSER( 599): 49|de|roolurl.txt|777619202264+378+51+0+1+00a1b,82300;777519202264+378+51+0+1+00a1b,82300;777319202264+378+51+0+1+00a1b,82300

U ovom ispisu se pokazuje na koje premium-rate brojeve se može slati SMS, a vidi se da je za sve brojeve sadržaj SMS poruke isti.

Ostali ispisi ne daju previše informacija jer je rezultat ili nerazumljiv, npr.

12-20 04:50:42.263 V/APKANALYSER( 599): return: java.lang.String = ÐÐ»Ñ Ð¿ÑÐ¾Ð´Ð¾Ð»Ð¶ÐµÐ½Ð¸Ñ Ð½Ð°Ð¶Ð¼Ð¸ÑÐµ Ð´Ð°Ð»ÐµÐµ.

ili ne daju rezultate koji se traže u analizi.

Budući da se došlo do premium-rate brojeva, time je analiza završena.

## Zaključak analize

Analizom se došlo do zaključka da je instalirani program zaista program u kojem je zamaskiran zloćudni program FakeInst. Ispisom povratnih vrijednost metoda su se dobili premium-rate brojevi na koje se šalje SMS-poruka određenog sadržaja. Korištenje drugih dodatnih mogućnosti nije bilo potrebno u ovom slučaju.

# Zaključak

ApkAnalyser je koristan alat za reverzni inženjering aplikacija za Android. Omogućava korištenje alata *adb* koji se nalazi u Android SDK te njime može instalirati te izbrisati programe na pametnom telefonu ili virtualnom stroju. Mogu se modificirati byte kodovi metoda čime se postižu razni ispisa pri radu programa te se jednostavno može vidjeti kako se mijenjaju vrijednosti varijabli, stoga itd. u iskočnom prozoru. To svakako olakšava cijeli proces reverznog inženjerstva aplikacija za Android telefona jer analitičar ne mora previše detaljno prolaziti kroz strojni kod da vidi što metoda vraća, a to posebice dolazi do izražaja kod „zamagljenog“ koda kojega zlonamjerni napadači često pokušaju stvoriti. Sve u svemu, dobar posao tvrtke Sony.

# Literatura

[1] ApkAnalyser – Developer World. <http://developer.sonymobile.com/knowledge-base/tools/analyse-your-apks-with-apkanalyser/>

[2] Home · sonyxperiadev/ApkAnalyser Wiki · GitHub. <https://github.com/sonyxperiadev/ApkAnalyser/wiki>

[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.