CARNet
HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

PREGLED SIGURNOSTI GOOGLE ANDROID OPERACIJSKOG SUSTAVA

NCERT-PUBDOC-2014-07-342

Nacionalni CERT

foi CSS

Sadržaj

Ovaj dokument izradili su studenti Fakulteta organizacije i informatike u Varaždin u sklopu kolegija Sigurnost informacijskih sustava. Dokument je izrađen uz pomoć laboratorija za Otvorene sustave i sigurnost FOI OSS te je recenziran od strane Nacionalnog CERT-a. Rad u seminarskom obliku nalazi se na http://security.foi.hr/wiki/index.php/Pregled_sigurnosti_Google_Android_sustava.

1. Uvod

Slika 1.1 - Android logo [Android Developers, 2014a]

Tvrtka Google je razvila mobilni operacijski sustav Android; dizajnirana je u pogledu otvorene platforme prema i licencirana je Apache licencom verzija 2.01 [Android, 2014a].

Zbog ovakvog pristupa bilo je potrebno definirati robusni sigurnosni model koji bi osiguravao sigurnost podataka, korisnika, aplikacija uređaja i mreže.

Prema [Android, 2014b]: Android je izrađen s fokusom na programere te su ugrađene različite sigurnosne kontrole kako bi smanjile teret prilikom razvoja softvera. No ovo nije unazadilo pristup naprednim sigurnosnim kontrolama, nego je sustav osmišljen u smislu početno dodijeljenih sigurnosnih kontrola tako da su svi programeri zaštićeni u određenom pogledu.

Osim fokusa na programere Android sustav sadrži ugrađene mehanizme za zaštitu korisnika; svakom korisniku dano je na uvid kako radi instalirana aplikacija, odnosno kojim svojstvima i funkcionalnostima uređaja aplikacija pristupa. Na ovaj način otežani su napadi uz pomoć društvenog inženjeringa, instaliranja zlćudnih aplikacija, napade na aplikacije trećih strana i sl.

Alati korišteni prilikom izrade ovog dokumenta su sljedeći:

Eclipse Kepler – integrirano razvojno okruženje sa Android Development Tool (ADT) pluginom

Android SDK – razvojni alati Androida

Android debugging bridge (ADB) – alat za komunikaciju i upravljanje emulatorima i fizičkim uređajem preko naredbene linije (eng. command line)

Dalvik Debug Monitor Server – alat za debuggiranje integriran u ADT

HTC Sensation – Android verzija 4.3.1, CyanogenMod 10.2

[1] Apache softverska licenca verzija 2.0 izvor: http://www.apache.org/licenses/LICENSE-2.0

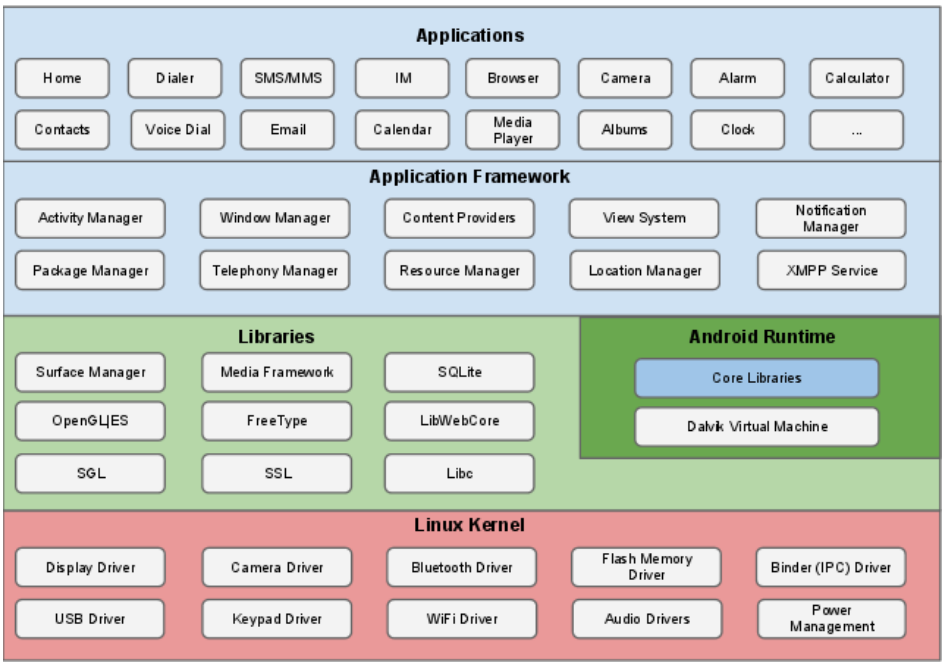
2. SIGURNOSNA ARHITEKTURA ANDROID PLATFORME

U svrhu ostvarivanja [Android, 2014b]: zaštite korisničkih podataka, zaštite sistemskih resursa o izolacije aplikacija. Android sustav pruža sljedeće sigurnosne mehanizme:

- robusna sigurnost na razini operacijskog sustava kroz Linux kernel
- obavezno izvršavanje aplikacija u sandbox[2] okruženju
- potpisivanje aplikacija
- Dopuštenja definirana aplikacijom i dopuštena od strane korisnika

Na **Slika 2.1** mogu se vidjeti komponente Android okoline te sigurnosne komponente kroz različite razine slojeva Android softvera. U sljedećim poglavljima bit će objašnjena okolina, u smjeru od na prema gore.

Slika 2.1 Android okolina [Android, 2014b]



[2] Sandbox- sigurnosni mehanizam zatvaranja aplikacije u kontrolirano okruženje u kojemu ima ograničene resurse, dopuštenja i sl. [Goldberg, Wagner, Thomas, i Brewer, 1996, str. 4]

2.1 Linux jezgra

Pri samom dnu Androidove okoline nalazi se jezgra operativnog sustava temeljena na Security-Enhanced Linux operativnom sustavu koja je dodatno prilagođena od strane Google za mobilne uređaje, tijekom razvoja novijih platformi Google je inkrementalno poboljšavao vlastitu verziju jezgre Linuxa te je pridonio stvaranju raznih sigurnosnih mehanizama koji su danas prihvaćeni te implementirani u glavnu verziju Linux jezgre verzije 3.3 [Kernelnewbies, 2012]
Neka od svojstava Androida implementiranih u Linux 3.3 verziju jezgre:

- *Binder* – interprocesni komunikacijski mehanizam visoke performanse koji služi za omogućavanje komunikacije između procesa.
- *Logger* – dio sustava za logiranje određenih poruka nevezanih za poruke jezgre.
- *Shrinker* – implementacija Androida u pogledu upravljanja aplikacijama, na ovaj način aplikacije ostaju u memoriji dok ih jezgra ne odluči uništiti.
- *Pmem* – daje mogućnost alociranja fizički većeg neprekinutog spremnika kada je potrebno.

Kada sagledamo Linux jezgru sa perspektive sigurnosti postoje mnogi mehanizmi koji osiguravaju siguran rad sustava primjerice [Android, 2014b]:

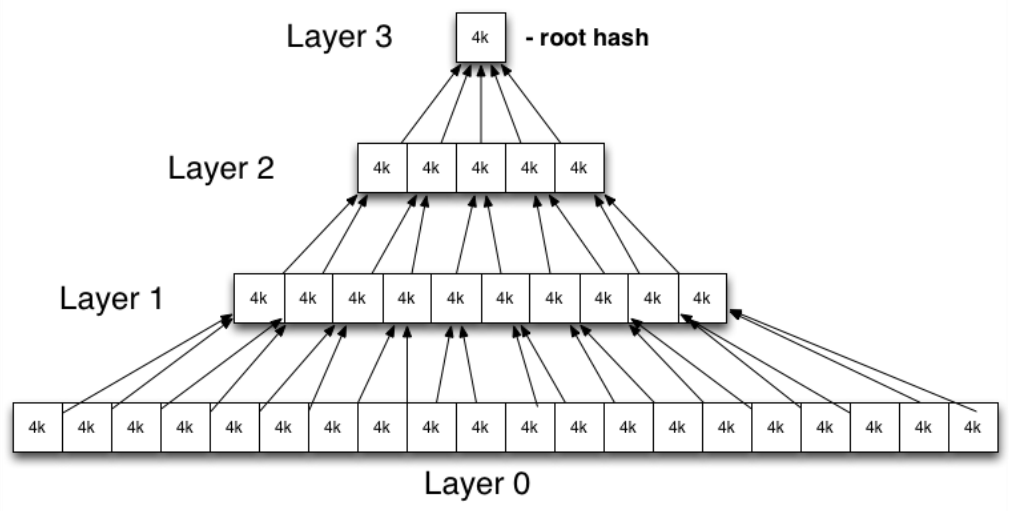
- Sustav dopuštenja temeljen na korisnicima
- Izolacija procesa
- Sigurni mehanizmi interprocesne komunikacije (IPC)
- Mogućnost uklanjanja nepotrebnih i nesigurnih dijelova jezgre

2.2 Dm-verity

Android verzije 4.4 podržava verificirani proces podizanja (engl. Boot) kroz koji provjerava valjanost uređaja (eng. device-mapper-verity - dm-verity), odnosno jezgra provjerava na razini blokova da pojedine funkcionalnosti nisu bile mijenjane [Android, 2014c]. Na ovaj način sprječava se pokretanje rootkit3 zlonamjernog softvera koji se može sakriti prilikom pokretanja samog uređaja bez dm-verity funkcionalnosti.
Valjanost (istintost) blokova se provjerava uz pomoć sažetka (eng. hash) SHA256. Svaki izračunati sažetak bloka se uspoređuje sa stablom sažetaka, gdje korijenski sažetak mora biti pouzdan, na ovaj način cijelo stablo i njegova pouzdanost tj. istinitost je povezana kroz sažetke

[3] Zlonamjeran softver koji se skriva od sustava te omogućava privilegirani pristup uređaju od strane napadača

svih blokova, te ako postoje i najmanja odstupanja sa stablom sažetaka došlo je do promjene na sustavu te dolazi do greške u čitanju i poruke o izmijenjenosti datotečnog sustava (**Slika 2.2**). Kako ne bi došlo do pada performansi prilikom učitavanja uređaja, provjera blokova radi se po principu prvog pristupa, odnosno kada sustav pristupa određenom bloku sustava, paralelno se izvršava provjera bloka naspram stabla sažetaka te se verificira traženi blok i ponaša se u skladu s prethodno navedenim pravilima.



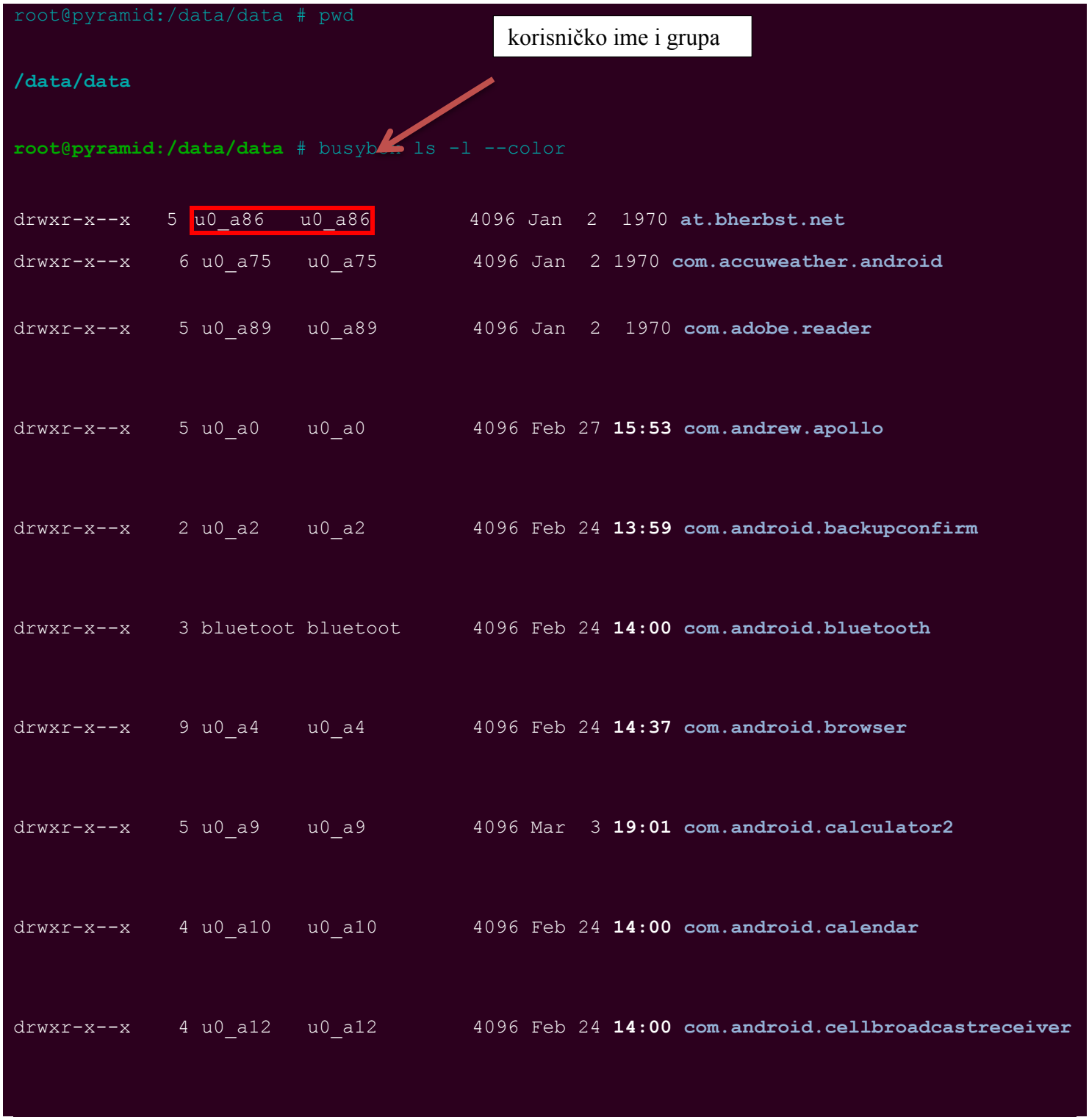
Slika 2.2 Princip rada dm-verity funkcionalnosti [Android, 2014c]

2.3 Koncept korisnika i grupa

Koncept korisnika i korisničkih grupa omogućuje da korisnik A ne može čitati i utjecati na resurse korisnika B ako ne koristi sigurnosne mehanizme provjere. Korisnik u ovome slučaju ne predstavlja nužno čovjeka kao korisnika uređaja nego podrazumijeva i aplikacije odnosno procese koji se izvode u sustavu i sl.

Svaki korisnik odnosno aplikacija sadrži svoj korisnički identifikator (eng. user id – UID), osim korisničkih identifikatora, postoje i grupe kojima se svakom korisniku može dodijeliti eng. group id – GID. Zbog ovakvoga modela te svojstva Linux sustava svaki resurs sustava pohranjen je u obliku datoteke, svaka datoteka pripada određenom korisniku, grupi ili ostalim korisnicima odnosno svim ostalim korisnicima sustava. Ovo se primjerice može vidjeti na slici 2.3 gdje je prikazana struktura datoteka na Android telefonu. Kao što je već navedeno svaka datoteka i direktorij sadrži pripadajućeg korisnika i grupu, kojima njihov vlasnik može upravljati, u našem

slučaju svaka aplikacija sadrži vlastiti direktorij te posjeduje prava na upravljanje samo nad svojim resursima.



```
drwxr-x--x    2 u0_a13   u0_a13        4096 Feb 24 13:59 com.android.certinstaller

drwxr-x--x    9 u0_a84   u0_a84        4096 Mar 18 01:34 com.android.chrome

drwxr-x--x    4 u0_a1    u0_a1         4096 Mar  8 14:06 com.android.contacts
```

Slika 2.3 Struktura datoteka, korisnika i korisničkih grupa

Kad bi običan korisnik pokušao ispisati sadržaj direktorija koji mu ne pripada dobili bi poruku o nedovoljnim pravima za pristup resursu. Kao što se može vidjeti na slici 2.4 nakon prijave kao administrator naredbom su moguće je pristupiti prethodno zabranjenom direktoriju /data te ispisati njegov sadržaj i dopuštenja. Na ovaj način postignut je sigurnosni mehanizam zaštite podataka aplikacija, resursa sustava i ostalih osjetljivih podataka kroz prava pristupa.

Ovim podacima moguće je pristupati na dva načina, kao administratori sustava (slika 2.5) odnosno root korisnik, ili kroz mehanizme sustava koji se nazivaju pružatelji sadržaja (eng. Content Providers).

```
shell@pyramid:/ $ busybox ls --color -l /data

ls: can't open '/data': Permission denied

1|shell@pyramid:/ $ su

root@pyramid:/ # busybox ls --color -l /data

busybox ls --color -l /data
drwxrwxr-x    2 system   system        4096 Mar 18 01:36 anr
drwxrwx--x    2 system   system        4096 Mar 18 11:57 app
drwx------    2 root     root          4096 Feb 24 13:58 app-asec
drwxrwx--x   34 system   system        4096 Mar 18 11:57 app-lib
drwxrwx--x    2 system   system        4096 Feb 24 13:58 app-private
drwxrws---    2 media    audio         4096 Feb 24 13:58 audio
```

nedovoljna prava za pristup direktoriju

root login

ispis podataka kao root

```
drwx------    5 system   system        4096 Jan  2  1970 backup
lrwxrwxrwx       1 root         root                    45  Feb  24  13:58  bugreports  ->
/data/data/com.android.shell/files/bugreports
drwxrwx--x    2 system   system        4096 Mar 18 11:57 dalvik-cache
drwxrwx--x  120 system   system        4096 Mar 18 11:57 data
....
....
```

Slika 2.4 Ispis strukture direktorija kao root korisnik

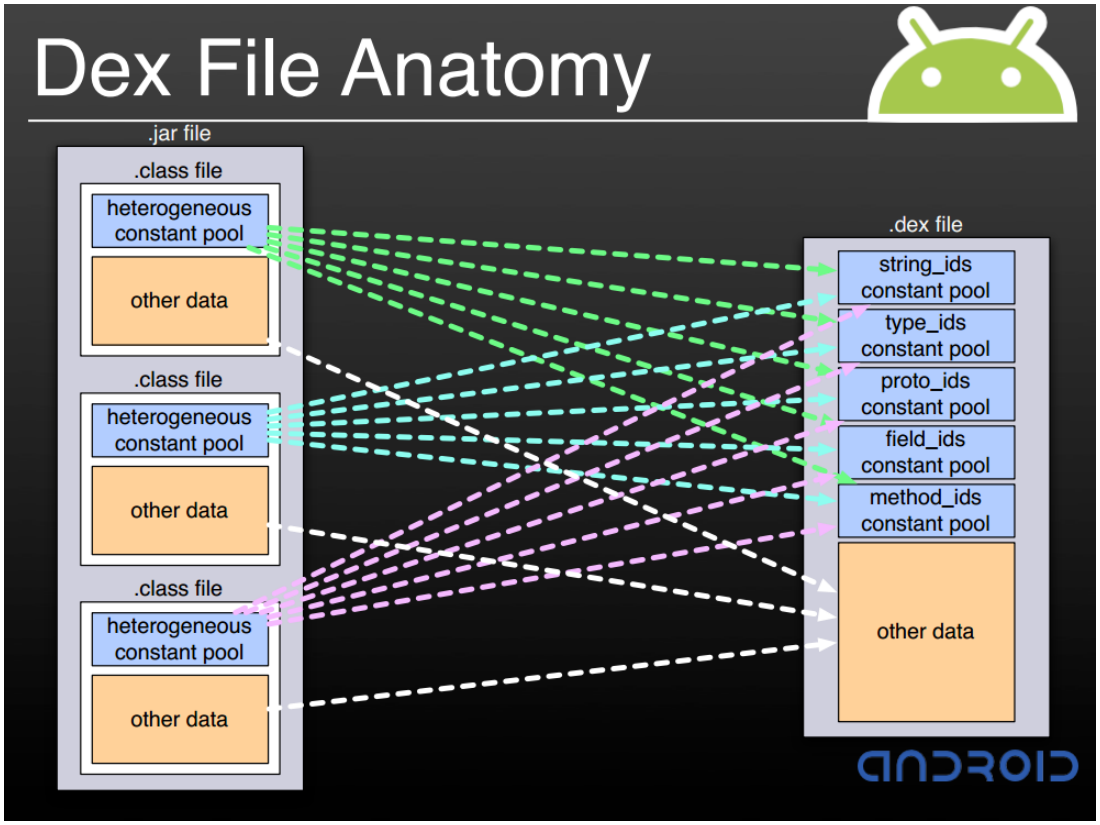
```
root@pyramid:/data # busybox ls -Ra | grep contact -i

com.android.contacts
com.android.providers.contacts
com.google.android.syncadapters.contacts
./data/com.android.contacts:
./data/com.android.contacts/cache:
./data/com.android.contacts/shared_prefs:
com.android.contacts_preferences.xml
contacts.en_GB.dict
contacts.en_US.dict
contacts.hr.dict
....
....
....
```

Slika 2.5 Pristup podacima kontakata kao root korisnik

Sloj iznad jezgre sustava sadrži temeljne biblioteke Androida pisane u C i C++ programskom jeziku te služe za upravljanje uređajem i obradu različitih vrsta podataka. Uz ove biblioteke nalazi se Android Runtime koji sadrži temeljne biblioteke Java programskog jezika zajedno s Dalvik virtualnim strojem (eng. Dalvik Virtual Machine, DVM). Dalvik virtualni stroj napravljen je po uzoru na Javin virtualni stroj skraćeno JVM, ali je optimiziran za mobilne uređaje različitim tehnikama minimalizacije korištenja memorije, brzog kreiranje novih procesa i sl. Na Android platformi izvorni Java kod se kompajlira u datoteke s nastavkom .class, te ih zatim poseban alat pod nazivom „dx“ pretvara u .dex datoteke koje se nazivaju Dalvik Executable file odnosno dalvik

izvršne datoteke.[Brähler,2010, str. 17-19]. Na Slika 2.6 Usporedba .class i .dex formata[Brähler, 2010, str. 4] može se vidjeti usporedba standardne Java arhive s nastavkom .jar koji sadrži .class Java datoteke te Android paketa s nastavkom .apk, Format .dex se razlikuje od standardno kompajliranih Java .class datoteka po bazenu konstanti (eng. constant pool), odnosno lokaciji i načinu spremanja konstantnih vrijednosti prilikom izvršavanja aplikacije. Ovako je spriječeno ponavljanje konstanti kroz različite klase, te sve klase u .dex datoteci dijele ove prostore.



Slika 2.6 Usporedba .class i .dex formata [Brähler,2010, str. 19].

3. ZAŠTITA KORISNIKA

Android sustav održava razinu sigurnosti s ograničenjima pristupa i primjenom enkripcije, no za pravilno funkcioniranje određenih aplikacija sustav im može omogućiti pristup do određenih osjetljivih podataka kao što su kontakti, korisnički profili, certifikati i sl. Ovakvi pristupi osjetljivim podacima bez kršenja sigurnosnih mehanizama osiguravaju se uz pomoć API-ja odnosno aplikacijskog programskog sučelja (eng. application programming interface).

3.1 Content Providers

Content Provider u prijevodu pružatelj sadržaja je mehanizam uz pomoć kojega Android sustav aplikacijama pruža strukturirani skup podataka. [Android Developers, 2014b]

Uz pomoć pružatelja sadržaja aplikacije mogu pristupati osjetljivim ili zaštićenim podacima ako su za to prethodno dobili dopuštenje sustava odnosno korisnika. Moguće je izraditi vlastiti pružatelj sadržaja ako programer želi da njegova aplikacija dijeli podatke s drugim aplikacijama. Pružateljima sadržaja se pristupa uz pomoć content URI[4], gdje se uz unos pružatelja podataka te imena koji pokazuje na određeni tablicu podataka može pristupiti određenim podacima sustava. Primjerice ako korisnik ili developer želi pristupiti rječniku koji se nalazi na sustavu koristit će sljedeći URI: *content://user_dictionary/words*, na ovaj način aplikacija pristupa pružatelju sadržaja pod nazivom user_dictionary (korisnički rječnik) te uz dodatak naziva tablice words (riječi) moguće je pristupiti korisničkom rječniku koji se nalazi u sustavu.

U slučaju da aplikacija pokušava pristupiti pružatelju sadržaja bez prethodnog dopuštenja sustav odbija vratiti tražene podatke te dolazi do sigurnosne iznimke odnosno izvođenje aplikacije se prekida.

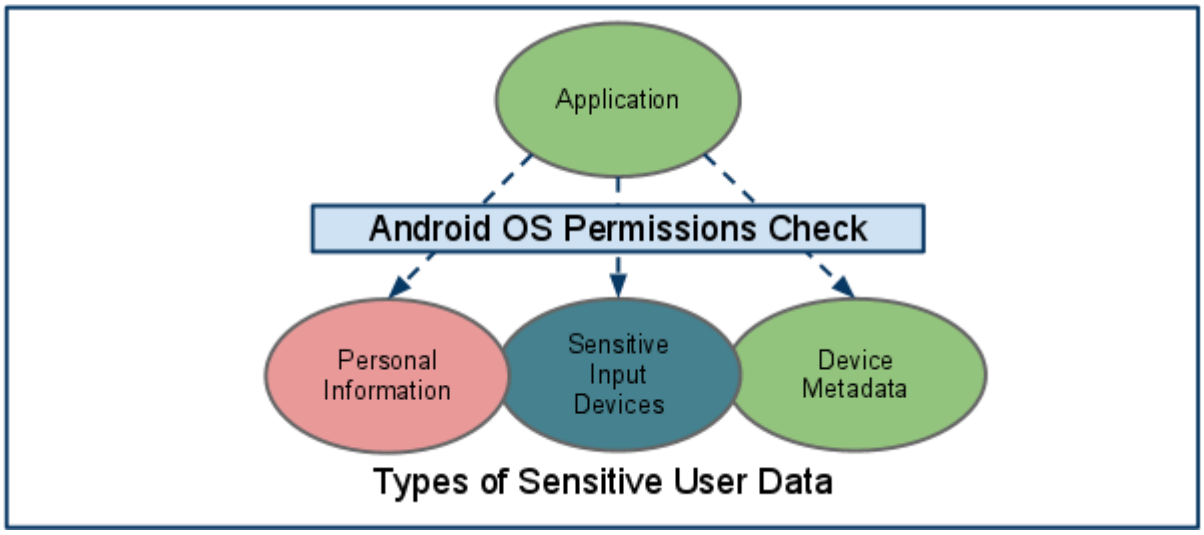
[4] URI – skraćenica od uniformni identifikator resursa (eng. *Uniform Resource Identifier*) niz znakova koji služi za identifikaciju i interakciju s resursima [Thompson, H. S., 2010]

3.2 Osobni podaci

Osobni podaci spremljeni su unutar lokalnog datotečnog sustava Androida u obliku SQLite baza podataka. Pristup ovim bazama ograničen prethodno opisanim sigurnosnim mehanizmima. Na **Slika 3.1** Popis pružatelja sadržajamogu se vidjeti sve baze osobnih podataka korisnika i osjetljivih podataka uređaja uz pomoć root korisnika. Kao što se može zamijetiti svaka baza podataka ima drugog korisnika odnosno različitog pružatelja sadržaja. Iz ovog razloga do navedenih podataka moguće je doći samo kroz sigurnosne mehanizme definirane sustavom uz prethodnu provjeru prava aplikacije (**Slika 3.2** Aplikacijski pristup osjetljivim podacima).

```
root@pyramid:/data # ls -l `find . -name "*.db" | grep providers`
-rw-rw---- u0_a11   u0_a11     122880 2014-03-18 21:38 calendar.db
-rw-rw---- u0_a1    u0_a1     1069056 2014-03-19 11:20 contacts2.db
-rw-rw---- u0_a1    u0_a1      311296 2014-03-01 20:23 profile.db
-rw-rw---- u0_a19   u0_a19      24576 2014-03-18 17:32 downloads.db
-rw-rw---- u0_a19   u0_a19      20480 2014-02-25 02:37 drm.db
-rw-rw---- u0_a19   u0_a19    4898816 2014-03-19 11:04 external-494b1b0c.db
-rw-rw---- u0_a19   u0_a19     225280 2014-03-16 21:23 internal.db
-rw-rw---- system   system      77824 2014-03-19 10:56 settings.db
-rw-rw---- radio    radio      126976 2014-03-18 08:38 mmssms.db
-rw-rw---- radio    radio      176128 2014-03-16 21:23 telephony.db
-rw-rw---- radio    radio       16384 2014-03-04 16:15 blacklist.db
-rw-rw---- u0_a1    u0_a1       16384 2014-03-16 05:50 user_dict.db
root@pyramid:/data #
```

Slika 3.1 Popis pružatelja sadržaja



Slika 3.2 Aplikacijski pristup osjetljivim podacima [Android, 2014b]

3.2.1 Osjetljive osobne hardverske komponente: geolociranje, kamera, mikrofon

Osjetljive komponente koje mogu prikupljati vanjske podatke zaštićene su sigurnosnim mehanizmima sustava te im je moguće pristupati uz pomoć *Intenta* ili aplikacijskog programskog sučelja (API). U slučaju da aplikacija koristi intent, odnosno šalje namjeru sustavu da želi koristiti kameru, mikrofon, geolokaciju i sl; sustav pronalazi pravilnu aplikaciju za rad s traženim hardverom te odabranu aplikaciju prikazuje korisniku. Prilikom slanja intenta nije potrebno eksplicitno tražiti korisničko dopuštenje davanja ovlasti nad traženim hardverom.

No ako aplikacija radi na način da direktno pristupa aplikacijskom programskom sučelju kamere, mikrofona ili geolokacije potrebno je zatražiti sigurnosno ovlaštenje od korisnika,odnosno sustava, prije rada sa ovim hardverskim komponentama.

U slučaju da korisnik ne želi otkriti trenutnu lokaciju svim aplikacijama može zabraniti odnosno isključiti lokacijske servise u postavkama Android pametnog telefona.

3.2.2 Device metadata

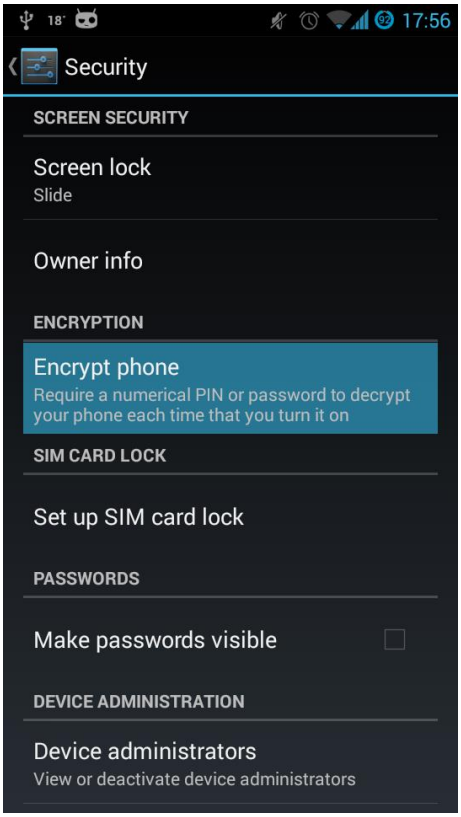
Svaki sustav može sadržavati podatke koji se ne smatraju dovoljno važnim da budu osjetljive naravi, ali indirektno mogu otkriti određene karakteristike i podatke koji se mogu iskoristiti u zlonamjerne svrhe. Iz ovog razloga sam sustav zabranjuje aplikacijama pristup sustavskim logovima, povijesti pregledavanja weba, postavki mreže, identifikacijske oznake softvera i hardvera i sl. U slučaju da aplikacija za vlastiti rad treba neke od meta podataka mora zatražiti eksplicitno dopuštenje korisnika za njihovo korištenje. Tek nakon što korisnik dopusti korištenje

metapodataka uređaja aplikacija ima dopuštenja sustava za pristup ovakvim podacima, zbog ovakvih načina rada korisnik je više svjestan što se događa te kakve aplikacije instalira na vlastiti uređaj.

3.3 Enkripcija pametnog telefona

Prema [Android, 2014b] Android verzije 3.0 te nadalje sustav pruža potpunu enkripciju datotečnog sustava i korisničkih podataka unutar kernela koristeći dmcrypt implementaciju AES128 sa CBC i ESSIV:SHA256. Ključ za kriptiranje je zaštićen AES128 koristeći ključ koji je izveden iz korisničke lozinke te tako zabranjuje pristup uređaju važeće korisničke lozinke.

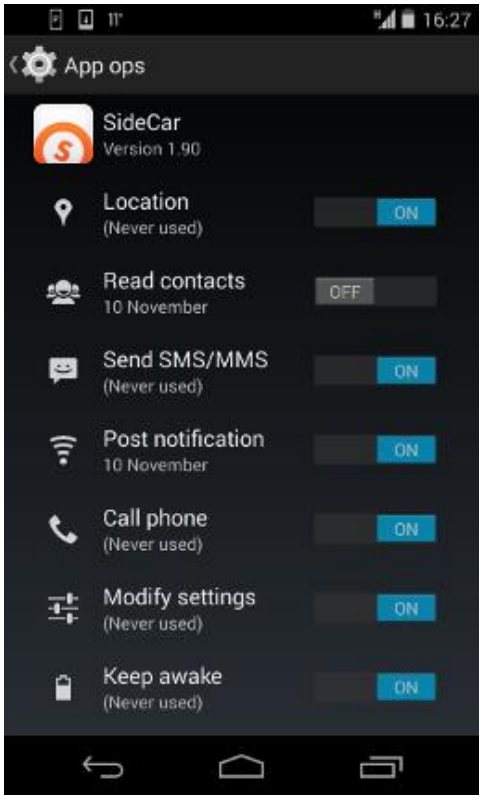
Dodatna zaštita protiv napada i pokušaja pogađanja lozinke je ostvarena slučajnim dodavanjem oznake lozinki (eng. random salt) uz postupak zaštite lozinki jednosmjernom kriptografskom funkcijom (eng. hashing) lozinke sa SHA1 koristeći standardni PBKDF2 algoritam. Moguće je postaviti osnovne zahtjeve kompleksnosti lozinke od strane administratora sustava. Korištenje kriptiranja Android pametnog telefona može se pronaći u Postavkama>Sigurnost>Kriptiranje telefona (Slika 3.)



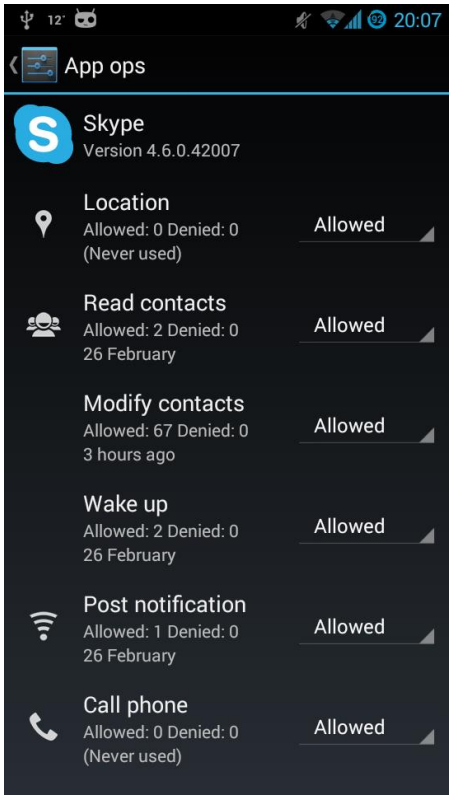
Slika 3.3 Sigurnosne postavke Android 4.3.1 uređaja

3.4 Sigurnosne postavke različitih sustava

U ovisnosti od korištenog tipa i verzije Android sustava mogu se pronaći različite postavke sigurnosti. Prema Electronic Frontier Foundation (2013) tijekom razvoja Google Android sustava u verziji 4.3 pojavila se funkcionalnost parcijalnog upravljanja dopuštenjima aplikacija pod nazivom App ops (Slika 3.4). Tijekom redovnih nadogradnji sustava, ova funkcionalnost je uklonjena uz izjavu Googlea da je ova funkcionalnost bila eksperimentalna te da bi mogla slomiti određene aplikacije sa svojim politikama sigurnosti. Iz ovog razloga je App ops funkcionalnost uklonjena iz sljedećih verzija izvornog Android sustava do daljnjega.



Slika 3.4 App Ops funkcionalnost Androida (4.3) [Electronic Frontier Foundation, 2013]



Slika 3.5 Privacy Guard postavke za Skype aplikaciju - CyanogenMod sustava

No zbog open source politike sustava određene verzije sustava zadržale su ovu funkcionalnost te su je dodatno unaprijedili primjerice CyanogenMod je open source verzija Android sustava održavana od strane zajednice, te je funkcionalnost App Ops funkcionalnosti implementirana pod nazivom *Privacy Guard*. Uz pomoć ove funkcionalnosti moguće je dopustiti ili zabraniti aplikacijama pristup određenim podacima i funkcionalnostima uređaja kao što se može vidjeti na Slika 3.5.

4. SIGURNOST ANDROIDA U PRAKSI

Prilikom praktičnoga istraživanja sigurnost Android sustava fokusirati će se na pokušaj nedopuštenog dohvata sljedećih podataka:

- Razne korisničke račune i lozinke
- Geolokacijske podatke
- Povijesti korištenja Interneta
- Poruke, slike, kontakti

Kako su sigurnosni mehanizmi zaštite Android sustava objašnjeni kroz ovaj dokument, podrazumijeva se da je do ovih podataka moguće doći nekoliko načina:

- Root administratorska prava pristupa uređaju
- Dohvat sadržaja uz pomoć Content Provider mehanizama
- Pregledom raznih sistemske zapise (eng. log)
- Eskalacija privilegija kroz rupe u sustavu
- Iskorištenje rupa u drugim instaliranim aplikacijama
- Ostali propusti?

Testiranje sigurnosti provođeno je na HTC Sensation, Android 4.3.1 uređaju sa CyanogenMod 10.2 sustavom. Kreiranje aplikacije s traženjem dopuštenja za pristup osjetljivim podacima je najjednostavniji način kako napasti određeni sustav, no većina korisnika neće instalirati ovakve aplikacije te su korisnici relativno zaštićeni funkcionalnostima sustava. Kako bi došli do određenih korisničkih podataka moramo znati nešto više o njihovim svojstvima: lokacija, struktura podataka, zaštitni mehanizmi. Uz pretpostavku da nemamo root pristup uređaju, pristup podacima možemo ostvariti uz druge aplikacije ili ugrađene mehanizme pružatelja sadržaja.

Prije vlastitih pokušaja dolaska do osjetljivih podataka te testiranja sigurnosti Androida istraženi su javni izvori sigurnosnih propusta Android sustava. Prema tim izvorima pronađen je

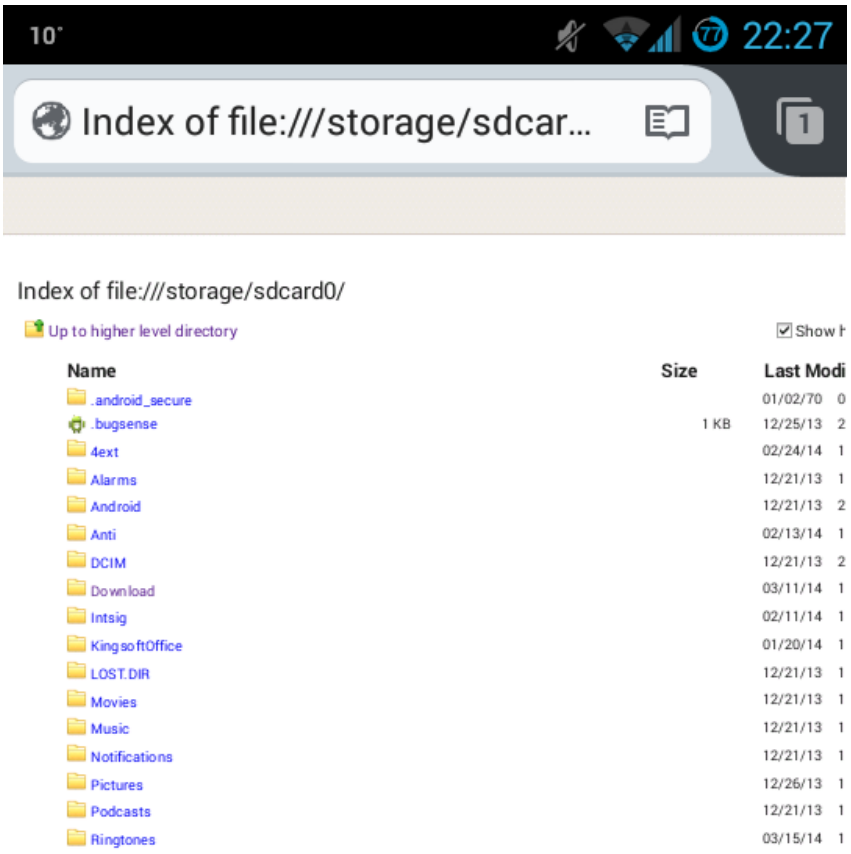
propust u Androidu do verzije 2.3.4 gdje je moguće uz pomoć web preglednika preuzeti osjetljive podatke uređaja tako što bi zlonamjerna skripta pozivala pružatelj sadržaja preko URI adrese unutar samog preglednika. Na ovaj način moguće je pristupiti kontekstu uređaja te preuzeti osjetljive podatke u obliku datoteka. [Cannon, 2011].

Ovaj propust je testiran na sljedećim preglednicima:

- Mozilla Firefox 27.0
- Google Chrome 33.0.1750.136
- Android Browser 4.3.1

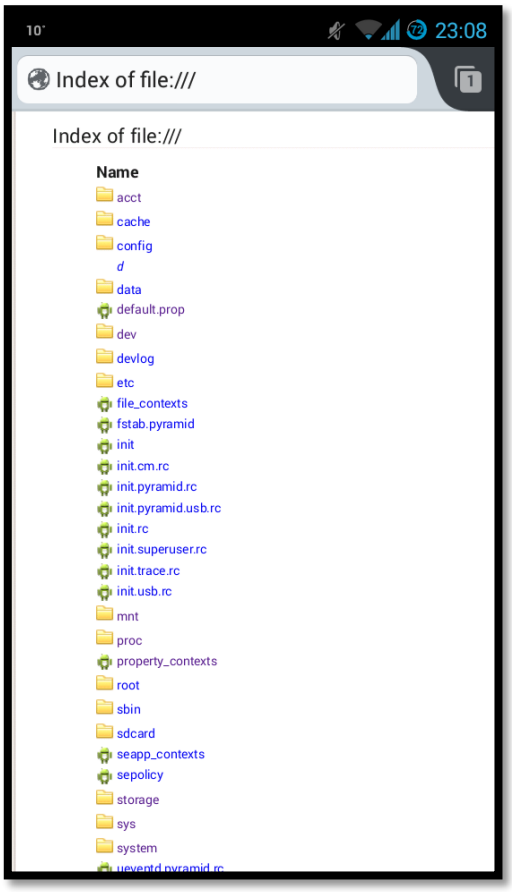
4.1 Mozilla Firefox

Sve verzije preglednika uspješno odbile pokušaj pristupa pružatelju sadržaja, no daljnjim istraživanjem i pokušajima pristupa datotečnom sustavu dva preglednika (Google Chrome i Mozilla Firefox) su uspješno prikazale sadržaj memorijske SD kartice uređaja na lokaciji ***file:///mnt/sdcard/*** prema ovome korisnik je mogao pregledavati i „šetati“ sadržajem SD kartice kao datotečnim preglednikom (Slika 4.1).

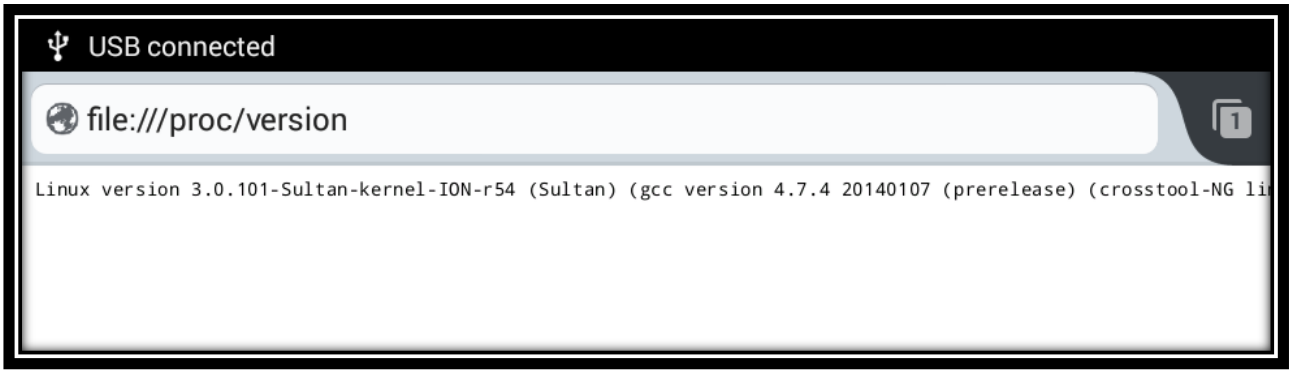


Slika 4.1 Datotečni sustav SD kartice

Proučavajući više *file* protokol Android preglednika zamijećeno je da Firefox preglednik može doći do korijenskog direktorija, te se „šetati“ po datotečnom sustavu (Slika 4.2). Ovakva funkcionalnost može predstavljati sigurnosni propust u slučaju da aplikacija može zapisivati i čitati iz osjetljivih datoteka. Primjerice napadač može dohvatiti verziju jezgre sustava, te na temelju dohvaćenih podataka iskoristiti ranjivosti te verzije sustava(Slika 4.3).



Slika 4.2 Pristup korijenskom direktoriju Mozilla Firefox



Slika 4.3 Pristup osjetljivoj datoteci /proc/version

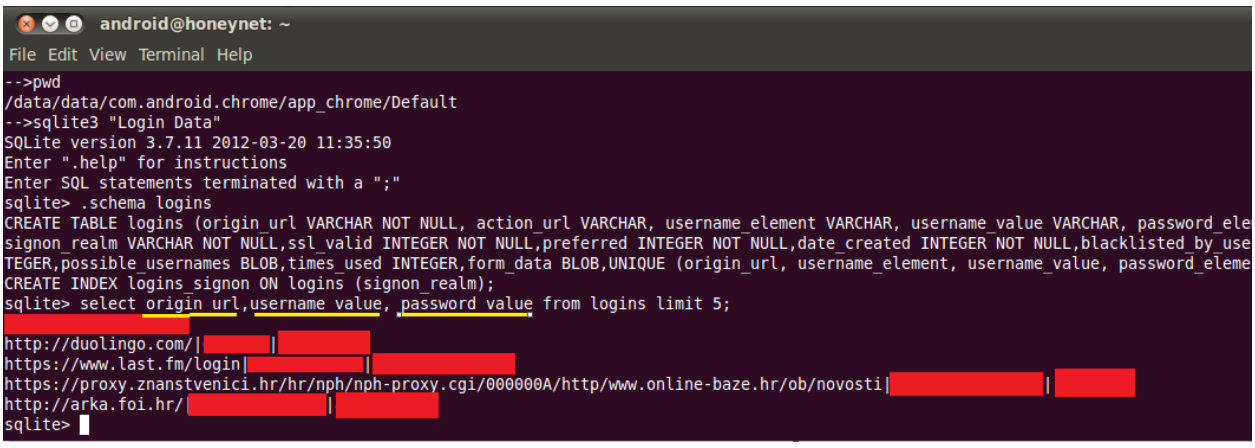
Trenutne verzije preglednika zaštićene su sa *Same Origin Policy (SOP)* mehanizmom koji zabranjuje izvršavanje skripti te dohvat drugih datoteka ako se ne nalaze na istoj lokaciji i protokolu kao izvršavana skripta. Primjerice ako se skripta izvršava na lokaciji: *https://www.foi.hr/skripta1.html* znači da je skripta izvršavana unutar *www.foi.hr* domene sa *https* protokolom. Ovakva zaštita zabranjuje skripti koja se izvršava na stranici *skripta1.html* izvršavanje i dohvat bilo kojih datoteka ili vanjskih skripti.

Daljnjim istraživanjem propusta Android sustava, ranjivost unutar Firefox preglednika je već bila otkrivena prema ViaForensics (2013), bilo je moguće zaobići SOP mehanizam uz pomoć kreiranja simboličkog linka na datoteku koju skripta želi preuzeti. Ovo je bio ozbiljan propust te je bilo moguće dohvatiti bilo koju datoteku kojoj je Firefox preglednik imao pristup unutar korijenskog direktorija. Sigurnosni propust je pravilno prijavljen Mozzili te je prema ViaForensics (2013) datuma 18.09.2013 pravilno riješen.

Tijekom testiranja pronađenog propusta i koristeći metode kreiranja simboličkih linkova Mozzila Firefox 27.0 bila je otporna na navedeni napad, no to ne znači da postoji mogućnost drugačijih vrsta napada te dohvata osjetljivih podataka uz pomoć Firefox preglednika.

4.2 Google Chrome

Osim prethodno navedenih propusta i sigurnosnih zakrpa Firefox preglednika, ustanovljena je još jedna moguća ranjivost koju je moguće iskoristiti s administratorskim pristupom uređaju ili ranjivosti eskalacije privilegija. Google Chrome browser sprema razne osjetljive korisničke podatke u obliku ne kriptiranih SQLite baza na lokaciji /*data/data/com.android.chrome/app_chrome/Default* kao što se može vidjeti na Slika 4.4

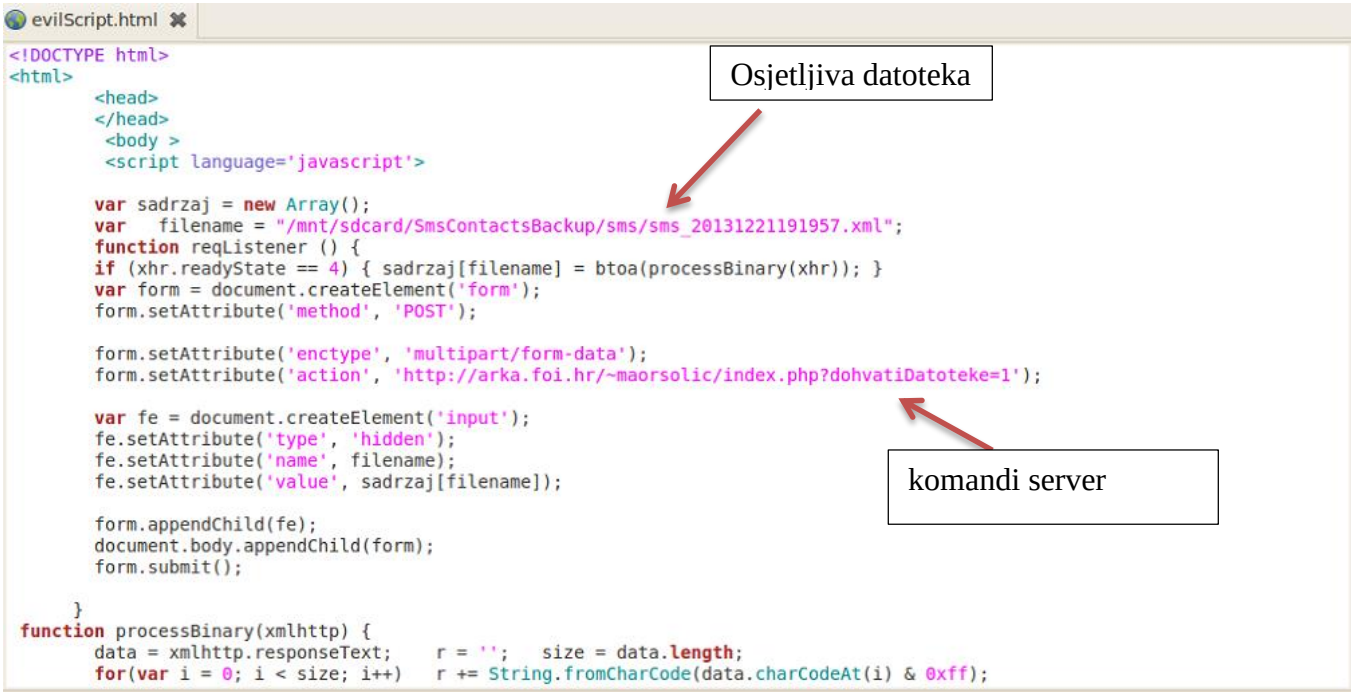


Slika 4.4 Korisnički podaci i lozinke spremljene u običnome tekstu

U slučaju da zlonamjerni korisnik eskalalira privilegije pristupa podacima svi osjetljivi podaci korisnika su dostupni bez ikakve enkripcije, te bi bilo preporučeno napraviti obfuskaciju i enkripciju osjetljivih podataka kao preventivnu mjeru zaštite od zlonamjernih napada.

4.3 Zlonamjerne skripte

Kako bi demonstrirali način rada zlonamernih skripti i krađe osjetljivih podataka kreirana je osnovna skripta koja datoteku s osjetljivim podacima prenosi na komandni server (**Slika 4.5**). Skripta je kreirana po uzoru na prethodno pronađeni i popravljeni propust [Cannon, 2011]. Nakon što skripta dohvati ciljanu datoteku i pokrene prijenos podataka na server, server prihvaća primljene podatke te ih sprema u obliku JSON polja unutar datoteke *payload.txt* na samome serveru *(***Slika 4.6***)*.



Slika 4.5 Zlonamjerna payload skripta

index.php

```
<html>
<body>


</body>
</html>
<?php

if(isset($_GET['dohvatiDatoteke'])){
  $fp = fopen("payload.txt", "w") or die("Ne mogu otvoriti datoteku za zapisivanje");
  fwrite($fp, print_r(json_encode($_POST), TRUE)) or die("Ne mogu zapisati podatke");
  fclose($fp);
  echo "Payload uploadan na <a href=\"payload.txt\">payload.txt</a>!";

}

?>
```

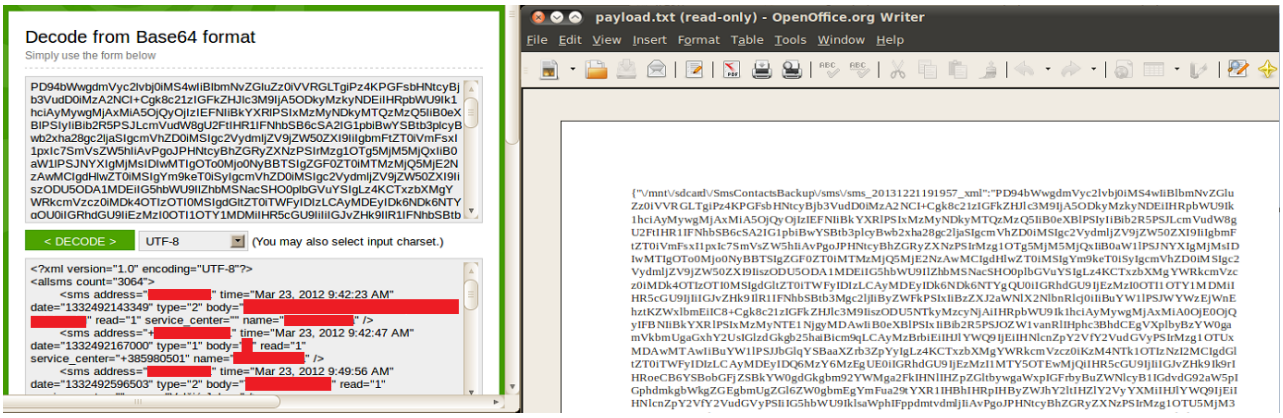
Slika 4.6 Skripta komandnog servera

Ako bi zlonamjerni korisnik kreirao aplikaciju koja traži dopuštenja:

- Pisanje/čitanje na vanjsku memoriju
- Internet dopuštenja

Mogao bi pretražiti vanjsku memoriju u svrhu pronalaska osjetljivih podataka; mnogi korisnici aplikacije prebacuju na SD karticu iz raznih razloga, najčešće je to ušteda interne memorije, no na ovaj način aplikacije izlažu osjetljive podatke koji nisu zaštićeni od raznih napada. Nakon što aplikacija pronađe datoteku s osjetljivim podacima primjerice: certifikati, backup kontakata, fotografije i sl. Aplikacija zlođudnu skriptu kreira u direktoriju ciljane datoteke, na ovaj način zaobilazi se *Same Origin Policy* zaštita te skripta može učitati i prenesti ciljane datoteke na udaljeni server. Primjerice na **Slika 4.7** može se vidjeti krađa kontakata i SMS poruka koja se nalazi u backup datoteci „sms_20131221191957.xml“. Datoteka je spremljena u XML formatu te je skripta učitava i šalje na udaljeni server; server zatim prima ukradenu datoteku i sprema je u novu datoteku s nazivom *payload.txt* u JSON formatu. Moguće je prenesti i više datoteka odjednom, skripta uspješno radi na Google Chrome i Firefox pregledniku. Važno je napomenuti da je prethodno spomenute tehnike moguće izvršiti i samo s zlođudnom aplikacijom koja ima Internet dopuštenja i dopuštenje za rad s memorijom. Na ovaj način aplikacija direktno čita i prenosi osjetljive datoteke na komandni server, no primjer sa skriptom pokazuje kako je moguće prenijeti različite datoteke i uz pomoć preglednika. Kako su određeni preglednici pokazali mogućnost „šetanja“ kroz osjetljive direktorije, postoji rizik izvršavanja skripti unutar

konteksta samog preglednika te krađe osjetljivih podataka iz više zaštićenih direktorija u slučaju da se pronađe određeni propust unutar preglednika koji bi omogućavao upravljanje skriptama i zapisivanje datoteka zaobilaskom politike istog izvora (eng. *Same Origin Policy*) kao što je već prethodno pronađeno u propustu od strane ViaForensics (2013) kojeg je Mozilla ispravila.



Slika 4.7 Prikaz ukradenih podataka

5. ZAKLJUČAK

Kao što se može vidjeti iz prethodno navedenih propusta i njihovih zakrpa, Android sustav je relativno zaštićen od raznih zlonamjernih napada. Koristeći standarde SELinux jezgre te dodatne mehanizame zaštite pristupa osjetljivim podacima kao što su pružatelji sadržaja (eng. Content Provider) platforma je zaštićena od osnovnih napadačkih tehnika. U slučaju instalacije aplikacija koje su nesigurne te sadržavaju sigurnosne propuste, a posjeduju prava pristupa osjetljivim podacima, dovodi do nesigurnosti cijelog sustava.

Ove probleme i rizike sustav pokušava umanjiti prikazom sigurnosnih zahtjeva, odnosno prava pristupa aplikacije prije njene instalacije; na ovaj način korisnik ima veliku kontrolu nad aplikacijama koje instalira. Osim prikaza traženih dopuštenja aplikacije, Google Play sadrži i komentare zajednice na određenu aplikaciju te korisnik iz komentara drugih korisnika može uvidjeti kvalitetu, probleme i nesigurnost određenih aplikacija koje želi instalirati.

Ovisno o verziji Android sustava sigurnost se inkrementalno poboljšavala te uz pravilne postavke korisnik može imati visoku razinu zaštite, no razni proizvođači pametnih telefona ne prate trend nadogradnje softvera dovoljno brzo te na taj način izlažu vlastite korisnike sigurnosnim rupama koje napadači mogu iskoristiti. Iz ovih razloga obični korisnici imaju nižu razinu zaštite od naprednijih korisnika koji samostalno nadograđuju sustave.

6. LITERATURA

1. Android Developers. (2014a). Brand Guidelines. Dostupno 04.03.2014. na http://developer.android.com/distribute/googleplay/promote/brand.html
2. Android Developers. (2014b). Content Providers. Dostupno 12.03.2014. na http://developer.android.com/guide/topics/providers/content-providers.html
3. Android. (2014a). Licenses. Dostupno 04.03.2014. na http://source.android.com/source/licenses.html
4. Android. (2014b). Android Security Overview Guidelines. Dostupno 04.03.2014. na http://source.android.com/devices/tech/security/
5. Android. (2014c). dm-verity on boot. Dostupno 04.03.2014. na http://source.android.com/devices/tech/security/
6. Brähler, S. (2010). Analysis of the Android Architecture. Dostupno 11.03.2014. na http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf
7. Corbet, J. (2011). Bringing Android closer to the mainline [LWN.net]. Dostupno 06.03.2014. na https://lwn.net/Articles/472984/
8. Cannon, T. (2011). 1337day Inj3ct0r Exploit Database : vulnerability : 0day : shellcode by Inj3ct0r Team. Dostupno 15.03.2014. na http://1337day.com/exploit/description/17194
9. Electronic Frontier Foundation (2013). Google Removes Vital Privacy Feature From Android, Claiming Its Release Was Accidental. Dostupno 12.03.2014. na https://www.eff.org/deeplinks/2013/12/google-removes-vital-privacy-features-android-shortly-after-adding-them
10. Goldberg I., Wagner D., Thomas R., i Brewer E. (1996). A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker). Proceedings of the Sixth USENIX UNIX Security Symposium. Dostupno 04.03.2014. na https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/goldberg/goldberg.pdf
11. Kernelnewbies (2012). Linux 3.3. Dostupno 06.03.2014. na http://kernelnewbies.org/Linux_3.3
12. Thompson, H. S. (2010). What's a URI and why does it matter? Dostupno 12.03.2014. na http://www.ltg.ed.ac.uk/~ht/WhatAreURIs/

13. ViaForensics (2013). How I met Firefox: A tale about chained vulnerabilities – viaForensics Dostupno 15.03.2014. na https://viaforensics.com/mobile-security/chained-vulnerabilities-firefox-android-pimp-browser.html