SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI SEMINAR

Sigurnost u operativnom sustavu Android

Matej Vukosav

Voditelj: Krešimir Pripužić

Zagreb, svibanj, 2017.

Sadržaj

1. Uvod

Mobilni uređaju dio su današnje svakodnevice. Skoro svi ih imaju i njihova primjena je velika. Olakšavaju svakodnevne aktivnosti, pomažu pri obavljanju privatnih i poslovnih stvari ali i pružaju zabavu i opuštanje. Njihova sve veća popularnost i brojnost potaknula je razvijanje zlonamjernih alata kako bi se iskoristile njihove mane. Prema izvještaju sigurnosne kompanije *McAfee* svake godine se ukradu podaci intelektualnog vlasništva u vrijednosti preko 160 milijardi dolara. [1]

Uveden je pojam „računalni-kriminal“. Pojam označava zlonamjerne radnje koje su počinjene s namjerom da se našteti ugledu žrtve, učini fizička ili mentalna šteta pojedincu ili gubitak na direktan ili indirektan način korištenjem telekomunikacijske mreže i mobilnih uređaja.“ [2]

Nisu sve prevare i nastale štete posljedica korisničke naivnosti. Ranjivost softvera i nepravilnosti pružile su mogućnost hakerima da se domognu osjetljivih korisničkih podataka i iskoriste ih u vlastite svrhe. Većina ranjivosti softvera ispravi se u kratkom roku od pojavljivanja ali nije uvijek takav slučaj. Mnogi propusti mogli su biti spriječeni da su se zadovoljili određeni sigurnosni standardi.

U ovom radu obradit će se najčešći oblici implementacije zaštite u operativnom sustavu Android kao i načini na koje se sigurnost može narušiti ukoliko nije dobro implementirana te prilagođena namjeni aplikacije.

2. Sigurnost

„Sigurnost je stupanj zaštite od opasnosti, štete, gubitka ili kriminalne aktivnosti“ [2] Implementacija razine sigurnost u aplikacijama ovisi o namjeni aplikacije. Nije potrebna ista razina zaštite igrici, aplikaciji za evidenciju ocjena ili aplikaciji za vođenje bankovnog računa. O domeni aplikacije ovisi način na koji će se određene stvari razvijati.

Google određenim pravilima pokušava nametnuti programerima standarde razvoja aplikacija, čijim uvođenjem radi na mjerama prevencije te štiti korisnike od mogućih štetnih aplikacija.
Sve Android aplikacije prolaze rigorozna sigurnosna testiranja prije objavljivanja na Google Play Store. Programeri čije aplikacije ne zadovoljavaju Googleova pravila, nerijetko bivaju suspendirani ili upozoreni da izmijene aplikaciju prema Googleovim pravilima. Nakon što su sva pravila zadovoljena i aplikacija je prošla testiranja spremna je za objavljivanje u trgovini aplikacija. Objavljivanjem aplikacije u trgovini aplikacija ne znači da je aplikacija sigurna već da trenutnim ispitivanjem nije pokazala loše naznake što ne znači da je Googleovo testiranje bezgrešno. Čak i nakon instalacije aplikacije Google povremeno skenira ponašanje aplikacija te ukoliko utvrdi sumnjivo ponašanje obavještava korisnika i po potrebi blokira aplikaciju.

Sigurnost je veliko područje i potreba za sigurnošću javlja se u raznim situacijama. Npr. kod spremanja lokalnih podataka, kod komunikacije preko mreže, razmjene podataka ili pregleda osjetljivih podataka na uređaju. Sve te radnje imaju različitu potrebnu razinu zaštite.

Najčešći razlozi za narušavanje sigurnost su: neprikladno korištenje platforme, nesiguran spremnik podataka, nesigurna komunikacija, nesigurna autorizacija i autentifikacija, nedovoljna kriptografija, loša kvaliteta koda, te obrnuti inženjering (*engl. reverse engineering*). [3]

U ovom radu biti će obrađeni samo neki od navedenih razloga.

2.1 Narušavanje sigurnosti

Postoji više načina na koja se može pristupiti podacima Android uređaja. Neki od njih su ručne, fizičke i logičke akvizicije. [4]
Ručna akvizicija predstavlja ručno pregledavanje podataka na mobitelu korištenjem ekrana i tipkovnice, gdje se podacima pokušava pristupiti interakcijom s uređajem. Drugi način je fizička akvizicija koja podrazumijeva stvaranje posebne kopije memorije uređaja zaobilaženjem datotečnog sustava i analiziranje određenim metodama. Taj način nije uvijek pogodan jer može nepovratno oštetiti uređaj.

Treći način i ujedno način koji će biti više obrađen u ovom radu je logička akvizicija.

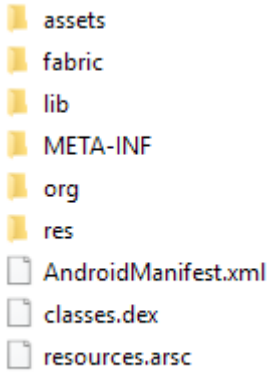
Logička akvizicija obuhvaća izvlačenje podataka s uređaja povezanog s nekim drugim uređajem. To može biti mobilni uređaj povezan USB kabelom sa stolnim računalom. Na taj način se može pristupiti internim podacima samog uređaja i aplikacija na njemu. Najjednostavniji način povezivanja je korištenjem *ADB* (*Android Debug Bridge*) alata. ADB je alat komandne linije koji omogućava konekciju između više uređaja. [5]

Nakon povezivanja s uređajem napadač može napraviti statističku ili dinamičku analizu podataka. [6]

U slučaju statičke analize podataka napadač treba preuzeti instaliranu .apk datoteku i otpakirati je kako bi mogao proučiti njen sadržaj. Apk (*engl. Android application package*) je format datoteke koju koristi Android za distribuciju aplikacija. Najjednostavniji način za pristup podacima je preimenovanje .apk ekstenzije u .zip. Sadržaj raspakirane zip datoteke može varirati ovisno o implementaciji aplikacije ali uvijek sadrži resurse aplikacije i konfiguracijski AndroidManifest datoteku.

```
application.apk -> application.zip
```

Ukoliko aplikacija nije zaštićena alatima za zaštitu koda, otvaranjem zip datoteke dostupni su određeni elementi aplikacije.



U *res* datoteci nalaze se svi resursi aplikacije kao što su xml datoteke izgleda ekrana, animacije, slike, boje i tekstovi korišteni u aplikaciji.

„*AndroidManifest* je xml datoteka u kojoj su definirane osnovne postavke aplikacije koje Android operativni sustav učitava prije pokretanja aplikacije.

Neke od postavki definiranih u AndroidManifestu su naziv paketa aplikacije, dozvole koje korisnik mora prihvatiti ako želi koristiti aplikaciju, minimalni nivo Android sustava na kojem se pokreće aplikacija i nazivi komponenata unutar aplikacije.“ [7] Osim resursa i konfiguracijske datoteke moguć je pristup i izvornom kodu aplikacije.

Najjednostavniji način za dohvat podataka ukoliko aplikacija nije zaštićena alatima za zaštitu koda sastoji se od nekoliko koraka. [8]

1. Prvi korak je promjena formata foldera aplikacije iz apk u zip.

application.apk -> application.zip

U .zip datoteci postoji datoteka classes.dex koja sadržava međukod aplikacije.

2. Potrebno je prevesti .dex datoteku u java izvorni kod. To se može napraviti s alatom dex2jar[1].
3. Nakon preuzimanja alata potrebno je kopirati classes.dex datoteku u raspakirani folder alata i pokrenuti iz komandne linije naredbu d2j-dex2jar.bat classes.dex. Ta naredba će napraviti novu .jar datoteku.

[1] Alat se može preuzeti na stranici https://github.com/pxb1988/dex2jar

```
C:\Users\Vuki\Documents\Programming\Android\dex2jar-2.0>d2j-dex2jar.bat classes.dex
dex2jar classes.dex -> .\classes-dex2jar.jar
```

4. Zatim je potrebno ekstenziju jar datoteke preimenovati u zip i raspakirati zip datoteku. U zip datoteci se sada nalaze izvorne klase aplikacije u .class formatu.
5. .class datoteke je moguće pročitati skidanjem dodatka za Android Studio[2] JD-IntelliJ[3]
6. Nakon instalacije alata .class datoteke se mogu otvoriti u Android Studio alatu gdje je vidljiv njihov izvorni sadržaj.

```
@java.lang.annotation.Retention(java.lang.annotation.RetentionPolicy.RUNTIME)
public @interface Language {
    java.lang.String CROATIAN = "hr";
    java.lang.String ENGLISH = "en";
}
```

Drugi način analize je dinamička analiza. Ona zahtjeva da aplikacija bude pokrenuta i analizira informacije koje su dostupne samo u trenutku izvođenja. Ti podaci su u statičkoj analizi nevidljivi. To mogu biti podaci koji se generiraju u određenom trenutku, ispisi aplikacije ili dekriptirani zaštićeni dijelovi koda.

Kako bi se dobio cjelovit uvid u način funkcioniranja aplikacije, napadači često primjenjuju oba načina.

Osim korištenja podataka aplikacije, u aplikaciju se može biti podmetnut određeni dio koda. Najčešći pokušaj je podmetanje dijela postavki aplikacije.

Prilikom implementacije postavki u aplikaciji prikazuju se predefinirani elementi ekrana. Ti elementi se zovu fragmenti. Ukoliko učitavanje tih elemenata nije

---

[2] Službeno razvojno okruženje za Android aplikacije.

[3] Dodatak je dostupan na stranici http://jd.benow.ca/

zaštićeno zlonamjerna osoba bi mogla prikazati novi vlastiti ekran s drugačijim mogućnostima od zamišljenih.

U nastavku će biti prikazani načini zaštite sigurnosti kojih bi se trebalo pridržavati prilikom izrade Android aplikacije.

2.2 Sigurnosni savjeti kojima se može povećati razina sigurnosti prilikom izrade Android aplikacije.

Kako bi se poboljšala sigurnost aplikacije poželjno je pridržavati se određenih uputa i standarda.

2.2.1 Android dopuštenja

Aplikacije unutar Android operativnog sustava odvojene su u zasebne cjeline. Kako bi aplikacije međusobno komunicirale i dijelile podatke svaka aplikacija mora eksplicitno podijeliti podatke i dopustiti njihovo korištenje. To se obavlja deklariranjem potrebnih dopuštenja koja omogućavaju korištenje dodatnih mogućnosti uključujući pristup svojstvima uređaja kao što su npr. kamera, mikrofon ili lokacija uređaja.

2.2.2 Zahtijevanje dopuštenja

Prilikom izrade aplikacije trebalo bi svesti broj zahtijevanih osjetljivih dopuštenja na minimum. Ograničavajući nepotreban pristup dijelovima uređaja sprječava se zlouporaba dopuštenja i poboljšava korisnička pristupačnost čime aplikacija postaje manje izložena vanjskim napadima. U pravilu ako aplikacija nema potrebe za određenom funkcionalnošću onda ne bi trebala ni zahtijevati njeno dopuštenje. Osim dopuštenja unutar aplikacije mogu se navesti i svojstva koja aplikacija koristi korištenjem <uses-feature> elementom u Manifest datoteci aplikacije.

```
<uses-feature android:name="android.hardware.bluetooth" />
```

Ako postoji svojstvo bez koje aplikacije ne može raditi onda ju je potrebno deklarirati korištenjem <uses-feature> s dodatkom <android:required>. Element deklarira jednu hardversku ili softversku značajku koju aplikacija koristi. Deklariranje elementa svojstva je samo informativno što znači da Android sustav ne provjerava podržava li uređaj tražena svojstva. [9]
Primjer korištenja svojstva u manifestu aplikacije.

```
<uses-feature android:name="android.hardware.camera" android:required="true" />
```

Iako je svojstvo deklarirano to ne znači da ga aplikacija ima pravo koristiti. Svejedno treba zatražiti korisničko dopuštenje.
Android dopuštenja svrstana su u dvije kategorije: normalna i opasna dopuštenja. Normalna dopuštenja su omogućena na razini sustava dok je opasna dopuštenja potrebno eksplicitno zatražiti.
U opasna dopuštenja spadaju dopuštenja za kalendar, kameru, kontakte, lokaciju, mikrofon, podatke mobilnog uređaja, senzore, sms i eksterni spremnik podataka.

CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

[4]

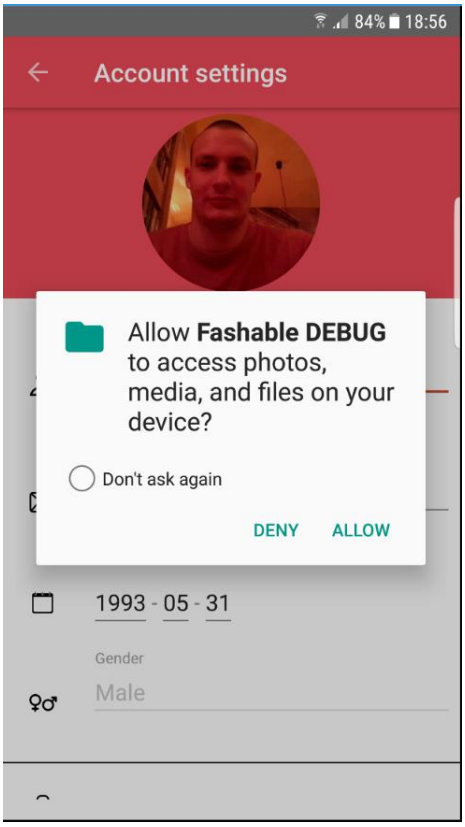
Slika 1 Opasna dopuštenja

Prije Android verzije 5.0 sva zahtijevana dopuštenja tražila su se prilikom instalacije aplikacije. Nakon Androida 5.0 uvedeno je eksplicitno traženje pojedinačnog dopuštenja u trenutku njegove potrebe. Sada je moguće odbiti pojedinačno dopuštenje i koristiti samo određene dijelove aplikacije. [10]

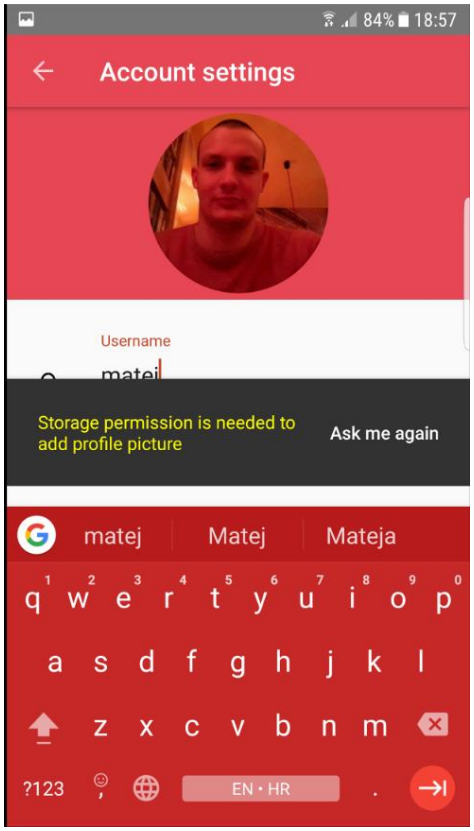
[4] Slika preuzeta s https://developer.android.com/guide/topics/permissions/requesting.html

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

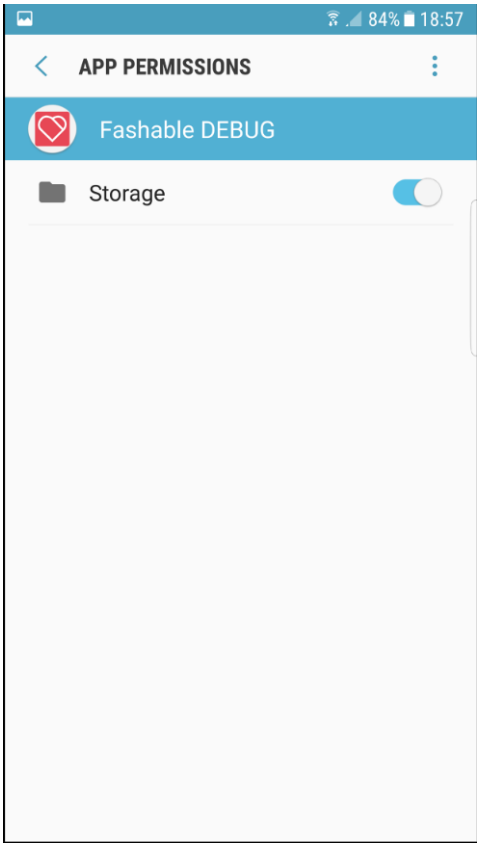
U nastavku su slike koje pokazuju korištenje zahtijevanja dozvola.
U trenutku kada je određena dozvola potrebna korisnika se pita za njeno dopuštenje. U slučaju da korisnik odbije može mu se prikazati poruka koja ga obavještava zašto je dopuštenje potrebno unutar aplikacije. Ako korisnik opet odbije, onda neće imati pristup određenom dijelu aplikacije koji zahtjeva odbijeno dopuštenje za rad. Ako korisnik prihvati dopuštenje moći će dalje nesmetano koristiti aplikaciju. Sva dopuštena dopuštenja mogu se vidjeti u postavkama aplikacija mobilnog uređaja.



Slika 2 Aplikacija traži korisnika dopuštenje za pristup skladištu podataka



Slika 3 Prikaz objašnjenja zašto je dopuštenje potrebno aplikaciji



Slika 4 Prikaz odobrenih dopuštenja u aplikaciji

Prilikom razvijanja aplikacije programeri često koriste priručne ispise kako bi vidjeli trenutno stanje aplikacije. Priručni ispisi zapisuju se u logički dnevnik koji može biti izvor osjetljivih podataka aplikacije.

2.2.3 Logički dnevnik

Prilikom zapisivanja podataka u logički dnevnik potrebno je biti oprezan. Podaci logičkog dnevnika su dijeljeni resurs kojemu može pristupiti bilo koja aplikacija koja ima „*READ_LOGS*“ dopuštenje. Iako su podaci uređaja privremeni i brišu se prilikom resetiranja uređaja, neodgovornim zapisivanjem osjetljivih podataka u logički dnevnik mogu se korisnički podaci zlonamjerno preuzeti i iskoristiti. Kako bi se spriječilo slučajno zapisivanje podataka u logički dnevnik poželjno je unutar aplikacije koristiti „*Log.d*“ oznaku koja zapisuje podatke samo u razvojnoj verziji.

Primjer korištenja logiranja: Log.d("tag", "message");

```
Log.d("Playground", "Username: Matej");
Log.d("Playground", "Password: MojPassword");
Log.d("Playground", "Ali ove poruke su vidljive samo u razvojnoj verziji jer se koristi oznaka Log.d ");
```

Slika 5 Unos unutar koda

```
-28908/? D/UserAnalysis.SDBH: SecurePlaceDbHelper()
-28799/vuki.com.playground D/Playground: Username: Matej
-28799/vuki.com.playground D/Playground: Password: MojPassword
-28908/? D/UserAnalysis: Create SecureDbHelper
-28799/vuki.com.playground D/Playground: Ali ove poruke su vidljive samo u razvojnoj verziji jer se koristi oznaka Log.d
-28908/? D/UserAnalysis.App: onCreate()
```

Slika 6 Prikaz logičkog dnevnika prilikom razvoja

Nakon što je aplikacija napravljena nije ju dovoljno samo zapakirati za korištenje jer se na taj način predaje izvorni kod u nezaštićenom obliku. Takav oblik pogodan je za razne manipulacije i jednostavan pristup načinu implementacije aplikacije. Kako bi se to spriječilo potrebno je kod zaštiti, a već ugrađeni i preporučeni način je zaštita korištenjem ProGuarda.

2.2.4 ProGuard

ProGuard je alat koji smanjuje, optimizira i prikriva Java izvorni kod. Prikriva kod na način da zamjenjuje nazive varijabli, metoda i klasa unutar projekta nasumice generiranim riječima ili slovima kako bi neovlaštenog korisnika spriječio u pristupu izvornom kodu.

ProGuard se prilikom izrade aplikacije može uključiti ili isključiti iz verzije aplikacije jednostavnom promijenom boolean vrijednosti „*minifyEnabled*“ varijable u build.gradle datoteci aplikacije.

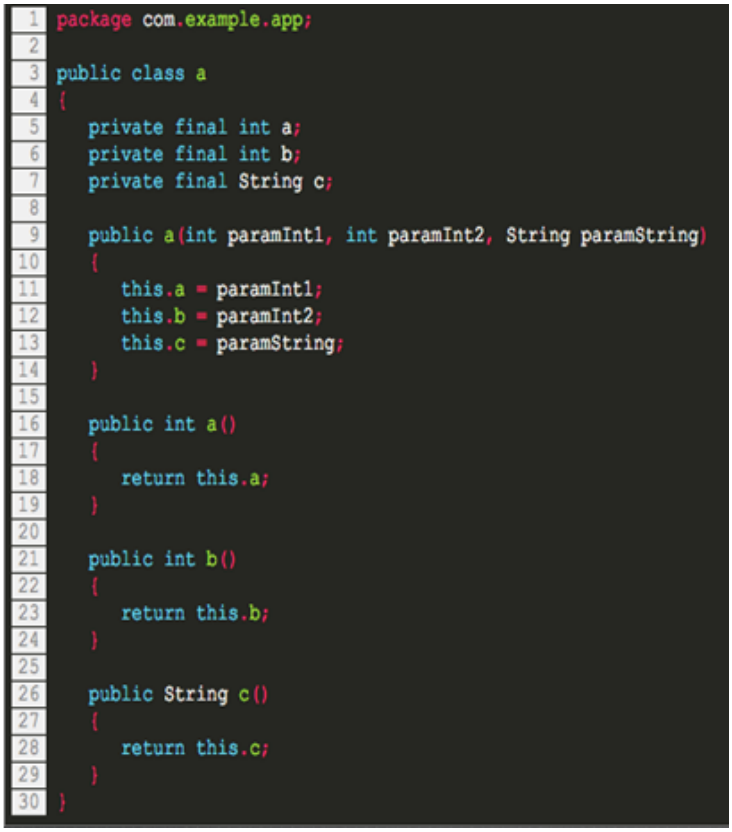
minifyEnabled true ← **prikriva kod**
shrinkResources true ← **smanjuje kod**
proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro' ← **definira datoteku za spremanje instrukcija za kasniji povrat skrivenog koda programerima s dozvoljenim pristupom**

Korištenje ProGuarda je preporučljivo koristiti samo prilikom izrade produkcijske verzije aplikacije jer usporava samu izgradnju aplikacije. Ako se koristi prilikom razvoja aplikacija će se svaki put optimizirati i obfuscirati iako za to nema potrebe.

Naveden je primjer korištenja Proguarda. [11]

```java
package com.example.app;

public class Data
{
    public final static int RESULT_ERROR = -1;
    public final static int RESULT_UNKNOWN = 0;
    public final static int RESULT_SUCCESS = 1;

    private final int mId;
    private final int mResult;
    private final String mMessage;

    public Data(int id, int result, String message) {
        mId = id;
        mResult = result;
        mMessage = message;
    }

    public int getId() {
        return mId;
    }

    public int getResult() {
        return mResult;
    }

    public String getMessage() {
        return mMessage;
    }
}
```

Slika 7 Primjer originalnog koda



```java
package com.example.app;

public class a
{
    private final int a;
    private final int b;
    private final String c;

    public a(int paramInt1, int paramInt2, String paramString)
    {
        this.a = paramInt1;
        this.b = paramInt2;
        this.c = paramString;
    }

    public int a()
    {
        return this.a;
    }

    public int b()
    {
        return this.b;
    }

    public String c()
    {
        return this.c;
    }
}
```

Slika 8 Primjer obfusciranog koda

Ukoliko zlonamjerna osoba preuzme instalacijsku datoteku aplikacije koja je zaštićena ProGuardom to uvelike otežava napadaču pristup izvornom kodu aplikacije i dohvat osjetljivih podataka.

Korištenje ProGuarda samo otežava pristup izvornom kodu. On se i dalje može dohvatiti korištenjem alata kao što je „apktool“ ali je izvorni kod i dalje u nečitljivom obliku.

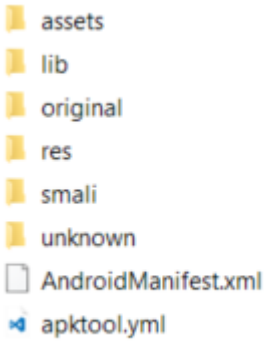
2.2.5 Apktool

Apktool je alat za dekodiranje zatvorenih binarnih android aplikacija. [12] Korištenje alata Apktool je jednostavno. Nakon preuzimanja datoteke potrebno je otvoriti komandnu liniju i upisati sljedeću naredbu koja dekodira .apk datoteku aplikacije.

```
$ apktool d testapp.apk
```

```
I: Using Apktool 2.0.0 on testapp.apk

I: Loading resource table...

I: Decoding AndroidManifest.xml with resources...

I: Loading resource table from file: 1.apk

I: Regular manifest package...

I: Decoding file-resources...

I: Decoding values */* XMLs...

I: Baksmaling classes.dex...

I: Copying assets and libs...
```

Izlaz naredbe je datoteka u kojoj se nalaze raspakirane komponente aplikacije.



U smali folderu nalazi se obfuscirani kod aplikacije.

```
.method protected onDraw(Landroid/graphics/Canvas;)V
    .locals 4

    const/high16 v3, 0x40000000    # 2.0f

    invoke-super {p0, p1}, Landroid/widget/FrameLayout;->onDraw(Landroid/graphics/Canvas;)V

    iget-object v0, p0, L co/vuki/app/views/PlayPauseView;->c:Landroid/graphics/Paint;

    iget v1, p0, L co/vuki/app/views/PlayPauseView;->g:I

    invoke-virtual {v0, v1}, Landroid/graphics/Paint;->setColor(I)V

    iget v0, p0, L co/vuki/app/views/PlayPauseView;->h:I

    iget v1, p0, L co/vuki/app/views/PlayPauseView;->i:I

    invoke-static {v0, v1}, Ljava/lang/Math;->min(II)I

    move-result v0

    int-to-float v0, v0

    div-float/2addr v0, v3

    iget v1, p0, L co/vuki/app/views/PlayPauseView;->h:I

    int-to-float v1, v1

    div-float/2addr v1, v3

    iget v2, p0, L co/vuki/app/views/PlayPauseView;->i:I

    int-to-float v2, v2
```

Slika 9 Primjer smali koda

„Smali“ kod kojeg je proizveo apktool je teško čitljiv i za njegovo potpuno razumijevanje potrebno je dosta vremena i truda, a rezultat je upitan.

Iako je izvorni kod obfusciran često postoji potreba za spremanjem raznih korisničkih unosa prilikom korištenja aplikacije. Takvi podaci se mogu spremiti na uređaj na razne načine.

2.2.6 Spremanje podataka

Pogrešan način spremanja osjetljivih podataka unutar aplikacija najčešći je i najlakši način zlouporabe korisničkih podataka.

Postoje tri osnovna načina za spremanje podataka unutar aplikacije. Podaci se mogu spremiti u internu memoriju aplikacije, vanjska memorija ili koristeći davatelj sadržaja (engl. *Content Provider*).

2.2.6.1 Interna memorija

Uobičajeno su podaci koji su spremljeni u internu memoriju aplikacije dostupni samo unutar aplikacije. Način na koji interni podaci mogu biti dostupni drugim aplikacijama je korištenje ključnih riječi MODE_WORLD_WRITEABLE i MODE_WORLD_READABLE. [13]

Korištenje tih ključnih riječi ne dozvoljava kontrolu nad dijeljenim sadržajem niti omogućava ograničenje pristupa samo određenim aplikacijama

```
  FileOutputStream fos = context.openFileOutput("SensitiveData.txt",
Context
                        .MODE_WORLD_WRITEABLE);
```

Na ovaj način bilo koja aplikacija može mijenjati sadržaj datoteke „SensitiveData“ što u pravilu nije dobra praksa jer se može promijeniti očekivano ponašanje aplikacije. Ključna riječ *MODE_WORLD_READABLE* dopušta bilo kojoj aplikaciji čitanje njenog sadržaja.

```
FileOutputStream fos = context.openFileOutput("SensitiveData.txt",
Context
                        .MODE_WORLD_READABLE);
```

Spremanje podataka u internu memoriju trebalo bi se obavljati korištenjem ključne riječi *MODE_PRIVATE* čime se zabranjuje pristup ostalim aplikacijama spremljenoj datoteci.

```
FileOutputStream fos = context.openFileOutput("SensitiveData.txt",
Context
                        .MODE_PRIVATE);
```

Ukoliko je potrebno podatke dijeliti s drugim aplikacijama ispravan način je korištenje davatelja sadržaja koji putem sučelja dopušta korištenje pripremljenih podataka.

2.2.6.2 Davatelj sadržaja (Content Provider)

Davatelj sadržaja je mehanizam spremnika koji se može ograničiti na samo jednu aplikaciju ili može dopustiti korištenje sadržaja i ostalim aplikacijama. Ako se podaci u

davatelju sadržaja ne žele podijeliti s ostalim aplikacijama potrebno je u Manifestu aplikacije dodati ključne riječi `android:exported=false` .
Davatelj sadržaja može ograničiti pristup određenim aplikacijama svim podacima, može ograničiti podatke kojima će aplikacije moći pristupati korištenjem posebnih dopuštenja. Korištenje podataka potrebno je omogućiti parametriziranim metodama kao što su „query“, „update“ i „delete“ kako bi se spriječilo zlonamjerno ubacivanje dijelova SQL koda.

Treći način i najmanje siguran je spremanje podataka u vanjsku memoriju.

2.2.6.3 Vanjska memorija

Podaci spremljeni u eksternu memoriju kao npr. SD kartica, globalno su vidljivi i dohvatljivi. Takve podatke korisnik može ručno brisati i svaka aplikacija im može pristupiti stoga se osjetljivi podaci ne bi trebali spremati u nju.

Kako bi se spremljeni podaci dodatno zaštitili lokalni podaci se mogu kriptirati i koristiti preko ključa koji nije direktno dostupan unutar aplikacije. Ključ se može spremiti u specijalni razred KeyStore i zaštititi lozinkom koja nije spremljena na uređaju.

2.2.7 KeyStore

KeyStore je razred unutar Androidove biblioteke koji predstavlja skladišni prostor za kriptografske ključeve i certifikate. [14]
Ne koristi se za direktno spremanje osjetljivih podataka aplikacije kao što su korisničko ime ili lozinka već pruža sigurno mjesto za spremanje njihovih privatnih ključeva.
Sastoji se od tri glavna elementa. To su privatni ključ, tajni ključ i certifikat. Svaka aplikacija može spremiti ključeve u *KeyStore* ali može pristupiti samo vlastitima. Idealno, aplikacija bi trebala izgenerirati ili primiti par podataka, privatni i javni ključa i spremiti ih u *KeyStore*. Javni ključ se može koristit za kriptiranje osjetljivih podataka aplikacije, prije spremanja u interni spremnik aplikacije, a privatni ključ za dekriptiranje podataka kada su potrebni.

Sam *KeyStore* se kriptira koristeći korisničku lozinku uređaja ili pin. Kada je uređaj zaključan pristup *KeyStoreu* je onemogućen. Nakon što korisnik otključa uređaj korištenjem lozinke ili pina, *KeyStore* postane dostupan aplikacijama.
Treba pripaziti da ako aplikacija koristi pozadinski servis koji radi i dok je uređaj zaključan, aplikacija neće imati pristup spremljenim podacima u *KeyStoreu*. [15]

Iako je ideja *KeyStorea* bila dobra i dalje ne pruža unificiran način zaštite podataka. [16]
Problemi koji se pojavljuju su:

1. KeyStore je podržan tek od Android verzije 4.3 što znači da ga starije verzije sustava ne mogu koristiti.
2. U Androidu verzijama manjima od 7.x pristup *KeyStore* podacima se onemogućava ako korisnik promijeni način zaključavanja ekrana dok su u Android verzijama 7.x podaci i dalje dostupni.
   Dostupni načini zaključavanja ekrana su:
   a. Klizanje (*engl. Swipe*)
   b. Uzorak(*engl. Pattern*)
   c. PIN
   d. Lozinka (*engl. Password*)
   e. Otisci prstiju (*engl. Figerprint*) i
   f. Nema (*engl. None*)

   Nakon što se promijeni način zaključavanja, *KeyStore* podaci se brišu i aplikaciji dojavljuju *InvalidKeyException* pogrešku u trenutku čitanja *KeyPair* podatka iz *KeyStore* datoteke.

```
System.err  W  java.security.InvalidKeyException: javax.crypto.BadPaddingException:
    error:0407106B:rsa routines:RSA_padding_check_PKCS1_type_2:block type is not 02
      at com.android.org.conscrypt.OpenSSLCipherRSA.engineUnwrap(OpenSSLCipherRSA.java:340)
      at javax.crypto.Cipher.unwrap(Cipher.java:1409)
    ...<some app specific code>

    Caused by: javax.crypto.BadPaddingException: error:0407106B:rsa routines:RSA_padding_check_PKCS1_type_2:block type is not 02
      at com.android.org.conscrypt.NativeCrypto.RSA_private_decrypt(Native Method)
      at com.android.org.conscrypt.OpenSSLCipherRSA.engineDoFinal(OpenSSLCipherRSA.java:273)
      at com.android.org.conscrypt.OpenSSLCipherRSA.engineUnwrap(OpenSSLCipherRSA.java:325)
      ... 23 more
```

Slika 10 Pogreške prilikom pokušaja enkripcije uređaja bez zaštite ekrana.

3. Zahtijeva da korisnik mora imati postavljen bilo koji način zaključavanja ekrana osim „Nema“ i „Klizanje“. Ako korisnik nema postavljen način zaključavanja i pokuša se postaviti enkripcija aplikacije će dojaviti pogrešku:

```
java.lang.IllegalStateException: Android keystore must be in
initialized     and unlocked state if encryption is
required.

java.lang.IllegalStateException: Encryption at rrst using
secure lock screen credential requested for key pair, but
the user has not yet entered the credential.
```

   Provjeru postoji li zaključavanje potrebno je ručno napraviti korištenjem *„Device Administration“* sučelja koje prikazuje određene sigurnosne postavke uređaja. [17] [18]

4. setEncryptionRequired() metoda koja radi softversku enkripciju se smatra zastarjelom od Android API verzije 23. (Trenutno je zadnja verzija 25). Što znači da ju nije preporučljivo koristiti u novijim verzijama, a ako se ne koristi onda će uzrokovati slabiju enkripciju na starijim verzijama. Shawn Willden, zaposlenik Googlea, je autor KeyStorea i predviđa da će ta metoda biti skroz izbačena iz upotrebe u Android verziji P. *„Enkripcija zapravo nikada nije bila vrlo sigurna. Formalno je zastarjela u Androidu M.(Marshmallow, v.6.x) Očekujem da će kompletno biti izbačena u Android verziji P. Htio sam je izbaciti u Android verziji O. ali su se našle druge stvari na putu.*“ [19]

Iako KeyStore nije pružio unificirani i siguran način zaštite podataka na uređaju podaci moraju biti zaštićeni i u slučaju njihovog ručnog unosa. Nakon što se ručno unesu podaci oni mogu biti poslani na udaljene poslužitelje kako bi se iskoristili. U slučaju plaćanja korištenjem kreditnih kartica slanje osjetljivih podataka o kartici nije preporučljivo. Zato su u Googleu razvili Android Pay.

2.2.8 Android Pay

Prilikom kupovine Android Pay ne šalje prave podatke kreditne kartice prodavaču već šalje virtualni broj računa koji predstavlja korisnika. Na taj način podaci korisničke kartice ostaju zaštićeni i u slučaju loše zaštite prodavačeve strane.

Iako bi kod aplikacije mogao biti dobro napisan i sve dodatne mjere poduzete kako bi se zaštitili podaci potrebno je osigurati aplikaciju od podmetanja i korištenja tuđih elemenata.

2.2.9 Podmetanje elemenata aplikacije

Unutar postavki aplikacije prikazuju se ekrani koji korisniku dopuštaju prilagodbu aplikacije prema vlastitom načinu korištenja. Određeni dijelovi mogu biti uključeni, drugi isključeni, može biti omogućeno spremanje lozinki ili spajanje na Internet samo ukoliko je dostupna Wireless konekcija.

Kako napadači ne bi podmetnuli svoj ekran potrebno je u dio aplikacije postaviti ograničenje mogućih ekrana. To se radi na način da se definira koji ekrani se mogu otvoriti ukoliko je potrebno prikazati postavke aplikacije.

Na taj način se sprječava otvaranje ekrana koji nisu od autora aplikacije.

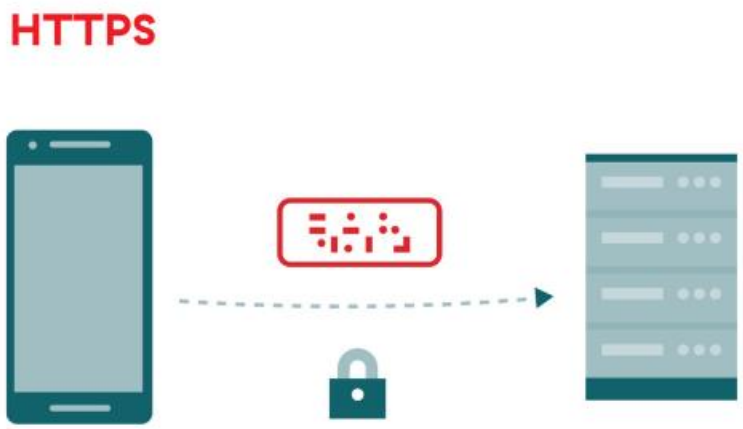
```
/**
 * This method stops fragment injection in malicious applications.
 * Make sure to deny any unknown fragments here.
 */
@Override
protected boolean isValidFragment( String fragmentName ) {
    return PreferenceFragment.class.getName().equals( fragmentName )
            || SettingsPreferenceFragment.class.getName().equals( fragmentName )
            || DataUsagePreferenceFragment.class.getName().equals( fragmentName )
            || AuthorizationPreferenceFragment.class.getName().equals( fragmentName );
}
```

Slika 11 Metoda za provjeru elemenata ekrana postavki

Ponekad se u postavkama aplikacije mogu mijenjati i osobni podaci korisnika. Takvi podaci bi se potom slali na server kako bi korisnik na svim uređajima s istim korisničkim imenom imao iste podatke. Podaci koji se šalju putem interneta na udaljeni poslužitelj mogu također biti meta napada. Podaci se šalju na Internet najčešće koristeći HTTP protokol.

HTTP protokol je nesigurna verzija protokola u kojoj su podaci koji se šalju svima dostupni. HTTPS je sigurna verzija protokola gdje S označava „siguran“ (*Secured*) označavajući da su podaci koji se šalju između klijenta i poslužitelja enkriptirani.

Slika 12 Primjer nezaštićene komunikacije koristeći HTTP protokol [5]



Slika 13 Podaci su enkriptirani korištenjem HTTPS protokola

[5] Slike su preuzete s https://news.realm.io/news/360andev-ana-baotic-best-practices-app-security-android/.

Naposljetku ukoliko su svi sigurnosti savjeti implementirani uvijek postoji mogućnost od gubitka samog korisničkog uređaja na kojem se mogu nalaziti osjetljivi podaci. S dostupnim uređajem napadač može pokušati pronaći sigurnosne propuste uređaja ili aplikacija na uređaju. Ako korisnik izgubi mobilni uređaj ili mu se otuđi, sigurnost privatnih podataka je glavni prioritet.

Unutar android operativnog sustava implementiran je način na koji korisnik može upravljati svojim uređajem na daljinu koristeći *Android Device Manager*.

2.2.10 Android Device Manager

Android Device Manager je menadžer uređaja implementiran na razini Android sustava kojim korisnik može upravljati uređajem na daljinu. Korisnik mora biti povezan Google korisničkim računom kako bi usluga bila moguća. Podatke je moguće i obrisati koristeći povezani račun. Na taj način se mogu zaštititi privatni podaci na uređaju da ne dođu u posjed neovlaštene osobe. Korisnik može s menadžerom uređaja također locirati gdje se uređaj nalazi i poduzeti mjere kako bi se uređaj vratio vlasniku.

3. Zaključak

U operativnom sustavu Android još uvijek ne postoji siguran način zaštite podataka unutar aplikacije. Postoje samo načini na koji se može zlonamjernoj osobi otežati pristup podacima. Ideja zaštite podataka unutar aplikacije je natjerati napadača da odustane jer proces dohvata podataka traje predugo.
U zadnjim inačicama android biblioteka postoje pokušaji da se napravi zaštićeni unificirani spremnik podataka aplikacije ali zbog velike količine različitih uređaja i verzija Androida to još uvijek ne radi kvalitetno.
Sigurnost je velik pojam i u ovom radu nisu pokriveni svi sigurnosti aspekti. Postoji centralizirana stranica „OWASP Mobile Security Project“ namijenjena sigurnosnim timovima koja pruža informacije o mogućim sigurnosnim prijetnjama i način kako još bolje unaprijediti sigurnost u aplikacijama. [20]
Zlonamjerni napadači konstantno traže nove načine kako bi narušili sigurnost aplikacija. Praćenje sigurnosnih trendova sigurno je jedan od boljih načina kako osigurati kvalitetu i integritet aplikacije.

4. Literatura

[1] H. D and J. K, Cyber crime and the Victimization of Women: Laws, Rights, and Regulations, Hershey, PA, USA: IGI Global. ISBN 978-1-60960-830-9, 2011.

[2] Wikipedia, "https://hr.wikipedia.org/wiki/Sigurnost," [Online]. Available: https://hr.wikipedia.org/wiki/Sigurnost. [Accessed 10 5 2017].

[3] »OWASP - Mobile top 10,« [Mrežno]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10. [Pokušaj pristupa 11 5 2017].

[4] V. Šimić, Proces prevođenja forenzičke analize Android operacijskog sustava. Završni rad br. 3478, Zagreb, 2014.

[5] »Android developers,« [Mrežno]. Available: https://developer.android.com/studio/command-line/adb.html.

[6] V. Berger, Zaštita mobilnih aplikacija od zlonamjernih izmjena, Zagreb: Diplomski rad br. 709, 2014.

[7] M. Vukosav, 3D igra za testiranje refleksa za Android operacijski sustav, Zagreb: Završni rad br. 4472, 2016.

[8] Trinimon, »Is there a way to get the source code from an apk file,« Stacoverflow, 13 5 2013. [Mrežno]. Available: http://stackoverflow.com/a/15395456. [Pokušaj pristupa 11 5 2017].

[9] »Android developers,« [Mrežno]. Available: https://developer.android.com/guide/topics/manifest/uses-feature-element.html.

[10] »Android developers,« [Mrežno]. Available: https://developer.android.com/guide/topics/permissions/requesting.html.

[11] »Fabric,« [Mrežno]. Available: https://fabric.io/blog/2014/02/18/mastering-proguard-for-building-lightweight-android-code/ .

[12] »Apktool,« [Mrežno]. Available: https://ibotpeaches.github.io/Apktool.

[13] »Android developers,« [Mrežno]. Available: https://developer.android.com/reference/android/content/Context.html.

[14] »Android developers,« [Mrežno]. Available: https://developer.android.com/reference/java/security/KeyStore.html.

[15] »Android Authority,« [Mrežno]. Available: http://www.androidauthority.com/use-android-keystore-store-passwords-sensitive-information-623779/.

[16] Doridori, »Android security and the forgetful keystore,« [Mrežno]. Available: https://doridori.github.io/android-security-the-forgetful-keystore/#sthash.72eEykoZ.dpbs. [Pokušaj pristupa 11 5 2017].

[17] Dori, »Stackoverflow - How to detect if pin password pattern is required to unlock phone,« [Mrežno]. Available: http://stackoverflow.com/questions/7879143/how-to-detect-if-pin-password-pattern-is-required-to-unlock-phone/27801128#27801128.

[18] »Android developers - Device Administration,« [Mrežno]. Available: https://developer.android.com/guide/topics/admin/device-admin.html.

[19] S. Willden, »Android security the forgetful keystore - Comments,« [Mrežno]. Available: http://disq.us/p/1hv3mll. [Pokušaj pristupa 11 5 2017].

[20] »OWASP Mobile Security Project,« [Mrežno]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project. [Pokušaj pristupa 11 5 2017].

[21] A. Baotić, »Realm - Best Practices in App Security,« 27 8 2016. [Mrežno]. Available: https://news.realm.io/news/360andev-ana-baotic-best-practices-app-security-android/. [Pokušaj pristupa 11 5 2017].

5. Sažetak

Sigurnost je važan element u današnjem svijetu. Velik broj kriminalnih radnji na mobilnim uređajima događa se zbog sigurnosnih propusta. Sigurnost korisničkih podataka na mobilnim uređajima s Android operativnim sustavom se može narušiti lošom implementacijom samih aplikacija ali i poboljšati korištenjem određenih uputa prilikom njihove izrade. U ovom radu su navedeni primjeri na koje načine napadač može pristupiti osjetljivim podacima korisnika te ih zloupotrijebiti. Navedeni su i načini na koje se korisnik može zaštiti od takvih napada te smjernice koje bi se trebale koristiti prilikom implementacije aplikacija kako bi se spriječio neovlašteni pristup osjetljivim podacima.