Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima

Štrbac, Renato

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:527567

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima

Rijeka, rujan 2024.

Renato Štrbac
0069082242

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima

Mentor: prof. dr. sc. Jonatan Lerga

Rijeka, rujan 2024.

Renato Štrbac
0069082242

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 12.03.2024.

Zavod: Zavod za računarstvo

Predmet: Kodiranje i kriptografija

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Renato Štrbac (0069082242)**

Studij: Sveučilišni diplomski studij računarstva (1400)

Modul: Programsko inženjerstvo (1441)

Zadatak: **Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima / An Application for Scanning for Potential Spyware on Mobile Devices**

Opis zadatka:

Potrebno je razviti mobilnu aplikaciju za pretragu potencijalno špijunskog softvera na mobilnom uređaju temeljem dopuštenja dodijeljenih pojedinim instaliranim aplikacijama. Temeljem analize sumnjivih dopuštenja koja se obično koriste za maliciozna djelovanja potrebno je označiti takve aplikacije kao potencijalne opasnosti. Aplikacija također treba omogućavati filtriranje i izliste instaliranih aplikacija po pojedinim dopuštenjima (npr. one koje mogu koristiti mikrofon, kameru i dr.).

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Mentor:
prof. dr. sc. Jonatan Lerga

Predsjednik povjerenstva za diplomski ispit:
prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2024.

_____
Ime Prezime

Zahvala

Zahvaljujem prof. dr. sc. Jonatanu Lergi na podršci i pomoći tijekom pisanja ovoga rada te korisnim raspravama i savjetima.

Sadržaj

Popis slika

Popis Isječaka

Poglavlje 1

Uvod

Sve više aplikacija danas zahtijeva brojna dopuštenja kako bi aplikacije mogle normalno funkcionirati. S rastućom kompleksnošću mobilnih *Android* aplikacija, korisnici postaju sve izloženiji raznim napadima na sigurnost njihovih osobnih podataka ili sigurnost njihovog mobilnog uređaja [1]. Većina korisnika *Android* mobilnih uređaja nije upoznata s različitim dopuštenjima ili opasnostima kojima ih ta dopuštenja izlažu. Vrlo često, kada aplikacija zatraži određena dopuštenja, korisnici bez sumnje na ikakvu opasnost odobre korištenje tih dopuštenja [2]. Ako korisnik malicioznoj aplikaciju dopusti određena dopuštenja, njegovi podaci mogu biti kompromitirani bez njihovog znanja. Iako platforme poput *Google Play Storea* ne dopuštaju plasiranje na tržište određenih aplikacija koje zahtijevaju neke od dopuštenja koja predstavljaju visoki rizik za sigurnost osobnih podataka ili uređaja korisnika, velika količina dopuštenja mogu upravljati privatnim podacima korisnika te ako korisnik nije informiran o njima, postoji mogućnost kompromitiranja njegovih osobnih podataka. Generalno gledajući, nije moguće garantirati sigurnost podataka korisnika ako se malicioznim aplikacijama svojevoljno dodijele dopuštenja stoga je potrebno prepoznati moguću opasnost.

Cilj ovog diplomskog rada je izrada aplikacije s pomoću koje se prosječni korisnik *Android* mobilnog uređaja može informirati o različitim dopuštenjima koje koriste aplikacije instalirane na njegovom mobilnom uređaju. S jednostavnim i intuitivnim korisničkim sučeljem, korisnik može pristupiti listi instaliranih aplikacija na njegovom mobitelu te za svaku aplikaciju ručno provjeriti dopuštenja koja ta aplikacija koristi. Cilj ovog rada je također bio omogućiti korisniku pristup bazi podataka najčešće

zatraženih dopuštenja kojoj se može pristupiti unutar izrađene aplikacije. Korisnik u bilo kojem trenutku može pristupiti ovoj bazi podataka kako bi se dodatno informirao o svrsi i opasnosti koje određena dopuštenja predstavljaju.

Poglavlje 2

Tehnološki stog implementacije aplikacije

Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima izrađena je isključivo za *Android* mobilne uređaje odnosno za *Android* 14 inačicu. Aplikaciju je moguće pokrenuti i na starijim inačicama *Androida*, ali njezina funkcionalnost nije zajamčena. Aplikacija je razvijena koristeći programski jezik Java, dok je XML korišten za definiranje korisničkog sučelja. Kako bi se uspostavila baza podataka raznih dopuštenja koje aplikacija može zatražiti, koristi se *Googleova* usluga *Firebase Cloud Firestore*.

2.1 *Android Studio*

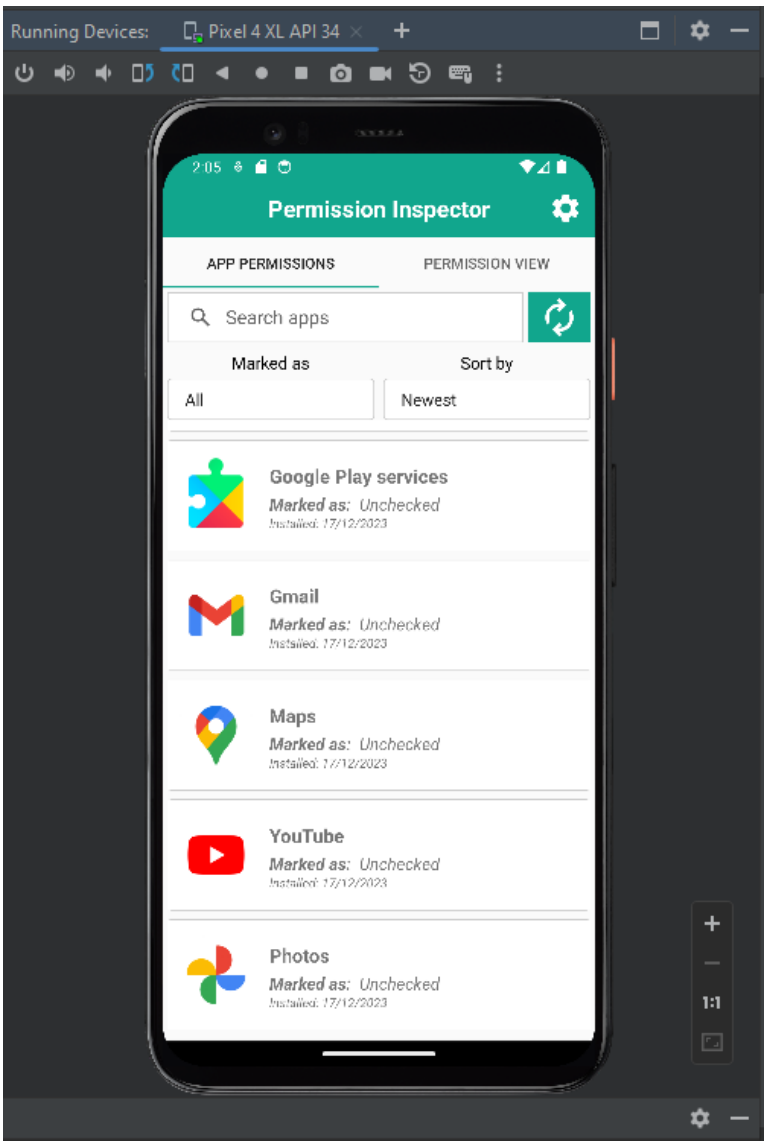
Android Studio je službeno integrirano razvojno okruženje (eng. *IDE* akronim za: *integrated development environment*) za razvoj aplikacija za *Android*, koje je razvila tvrtka *Google*. Ovo okruženje pruža sve potrebne alate za stvaranje, testiranje, otklanjanje pogrešaka i distribuciju aplikacija za *Android*. *Android Studio* se temelji na *IntelliJ IDEA*, integriranom razvojnom okruženju razvijenom od strane tvrtke *JetBrains* i podržava razvoj na programskim jezicima Java i Kotlin, koji su glavni programski jezici za razvoj *Android* aplikacija [3]. Uz to, *Android Studio* pruža potpunu podršku za XML za definiranje korisničkog sučelja. U svrhu izrade ove aplikacije, korišteni su sljedeći alati *Android Studioa*:

- **Uređivač koda** - Napredni uređivač koda s podrškom za programske jezike Java, Kotlin i XML koji su korišteni u izradi aplikacije. Neke od opcija koje omogućuje su automatsko dovršavanje koda, refaktoriranje, pregledavanje de-

finicija metoda i klasa, te alate za navigaciju kroz programski kod.

- **Emulator** - Ugrađeni virtualni emulator koji omogućuje jednostavno i brzo testiranje aplikacije bez potrebe za povezanim fizičkim uređajem. S pomoću emulatora moguće je testirati aplikacije na različitim inačicama *Android*a te simulirati različite uvjete rada ovisno o potrebi. Primjer ovog emulatora prikazan je na slici 2.1.
- **Podrška za *Gradle*** - Efikasan alat za automatizaciju izgradnje koji se koristi za upravljanje projektom i njegovim ovisnostima. Služi kao pomoć u konfiguraciji projekta, izgradnji aplikacija, rukovanju bibliotekama i vanjskim ovisnostima.

Osim alata korištenih u izradi aplikacije, *Android Studio* još pruža mnoge druge alate poput alata za analizu performansi, integraciju s *Gitom* te vizualni uređivač za korisničko sučelje aplikacije [3].



Slika 2.1 Emulator *Android Studioa* tijekom testiranja aplikacije

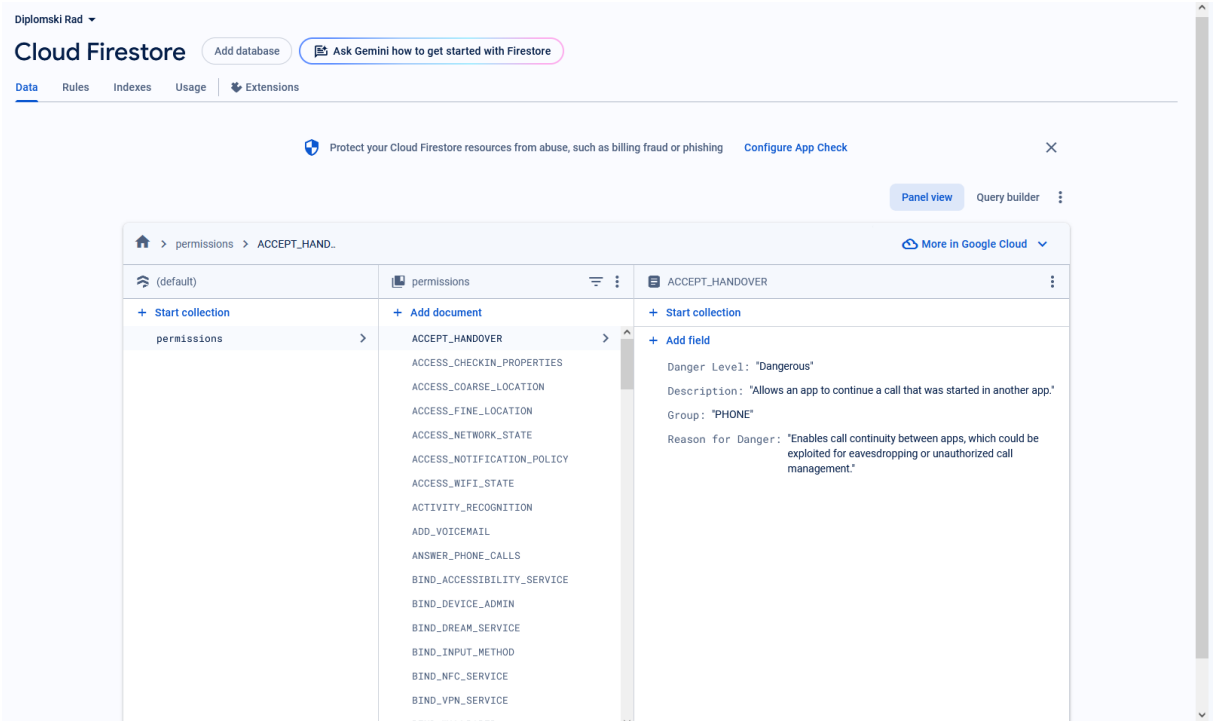
2.2 *Firebase Cloud Firestore*

Firebase Cloud Firestore je moderna i skalabilna baza podataka u oblaku koju pruža *Google* kao dio šire *Firebase* platforme. Baza podataka funkcionira na modelu *NoSql* baze podataka dokumenata koja omogućuje pohranu i dohvaćanje podataka u stvarnom vremenu na različitim platformama. Neki od najvažnijih atributa *Firebase*

Cloud Firestore baze podataka su:

- **Skalabilnost i performanse** - *Firestore* je dizajniran za skalabilnost i može podržavati aplikacije u prijelazu u veće projekte s milijunima korisnika i podataka. Distribuirana infrastruktura *Firestore* baze podataka osigurava veliku dostupnost i nisku latenciju pristupa podacima [4].
- **Integracija s različitim platformama** - *Firestore* pruža široku podršku za različite platforme poput *Androida*, *iOSa*, *Weba* te *Node.jsa*. Ova široka podrška omogućava razvoj aplikacija koje dijele poslužiteljsku stranu, te se tako smanjuje kompleksnost i vrijeme razvoja [4].
- **Rad u stvarnom vremenu** - Podrška sinkronizaciji podataka u stvarnom vremenu omogućuje korisnicima automatsko ažuriranje podataka, bez potrebe za ručnim osvježavanjem [4].
- **Ugrađeni sigurnosni model** - Omogućuje kontrolu pristupa podacima na temelju autentičnosti korisnika i vrijednosti u samim dokumentima [4].
- **Podrška za rad izvan mreže (eng. *Offline*)** - Izmjene podataka izvan mreže se automatski sinkroniziraju pri ponovnom povezivanju na mrežu [5].
- **Integracija s umjetnom inteligencijom** - *Firestore* ekstenzije omogućuju integraciju *Firestore* baze podataka s funkcionalnostima umjetne inteligencije, poput prevođenja jezika, klasifikacije slika i tako dalje [5].

Firebase Cloud Firestore se vrlo jednostavno može integrirati u *Android* aplikaciju s pomoću *Firebase* SDK (akronim od eng. *Software Development Kit*: Komplet za razvoj softvera). *Firestore* također pruža korisničko sučelje unutar *Firebase* konzole za pregled i upravljanje podacima, što olakšava administraciju i razvoj baze, prikazano na slici 2.2.



Slika 2.2 Sučelje za upravljanje bazom podataka *Firebase Cloud Firestore*

Poglavlje 3

Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima

Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima sastoji se od dvije glavne funkcionalnosti: Pregled po dopuštenjima pojedinih instaliranih aplikacija i pregled svih dopuštenja koja aplikacija može zatražiti. Nakon pregleda, korisnik može označiti aplikaciju prema jednoj od ponuđenih opcija sigurnosnih nivoa. Međutim, korisnici mobilnih uređaja *Android* često ne posjeduju dovoljno znanja o pojedinim dopuštenjima, stoga aplikacija pruža mogućnost pregleda najčešćih dopuštenja koja se mogu zatražiti. Da bi aplikacija mogla nesmetano izvršavati navedene funkcionalnosti, ključno je pri instalaciji ili prvom pokretanju aplikacije odobriti dopuštenje *android.permission.QUERY_ALL_PACKAGES*.

3.1 *QUERY_ALL_PACKAGES*

QUERY_ALL_PACKAGES ili *android.permission.QUERY_ALL_PACKAGES* dopuštenje je *Android* dopuštenje koje omogućuje korištenje *Android PMa* (akronim od eng. *Package Manager*: upravitelj paketa), odnosno izvršavanje upita o svim instaliranim aplikacijama [6]. S pomoću *Android PMa* omogućuje se temeljna funkcionalnost aplikacije odnosno pregled instaliranih aplikacija i njihovih dopuštenja. Ako se ovo dopuštenje ne odobri, aplikacija neće funkcionirati.

Android PM se koristi za dohvaćanje informacija o svim instaliranim aplikacijama na *Android* uređaju. Za ovu aplikaciju se specifično koristi kako bi se dohvatila sva

dopuštenja koja su zatražena od strane instaliranih aplikacija, neovisno o njihovom statusu odnosno odobrenju. Sljedeći isječak koda 3.1 prikazuje inicijalizaciju *Android PMa*:

```
1 packageManager = requireActivity().getPackageManager();
```

Isječak koda 3.1 Inicijalizacija *Android PMa*

Osim dopuštenja, *Android PM* se također koristi kako bi se dohvatile informacije pomoću kojih korisnik može filtrirati u pretraživanju instaliranih aplikacija.

3.2 Pregled dopuštenja pojedinih aplikacija

U ovom poglavlju prvo će se opisati sve informacije koje se dohvaćaju i koje su potrebne za prikaz i pregled svih instaliranih aplikacija. Nakon osnovnih informacija će se opisati ostale funkcionalnosti poput filtriranja aplikacija, pregleda detalja aplikacija te označavanje aplikacija ovisno o sigurnosti te aplikacije.

3.2.1 Prikaz instaliranih aplikacija

Temeljna funkcionalnost aplikacije je pregled svih instaliranih aplikacija na korisnikovom mobilnom *Android* uređaju. Aplikacija pri učitavanju dohvaća informacije za sve instalirane pakete aplikacija kako je prikazano u isječku koda 3.2.

```
1 packages = packageManager.getInstalledPackages(0);
```

Isječak koda 3.2 Primjer dohvaćanja svih instaliranih paketa aplikacija

Pomoću paketa, mogu se dohvatiti sve potrebne informacije o aplikacijama. Isječak koda 3.3 prikazuje dohvaćanje informacija pojedinih aplikacija za odgovarajući paket:

```
for (PackageInfo packageInfo : packages) {
      ApplicationInfo appInfo = packageInfo.applicationInfo;
      ...
      appList.add(appInfo);
      }
```

Isječak koda 3.3 Dohvaćanje pojedinih informacija aplikacije za svaki paket te spremanje tih informacija u listu

Nakon što se informacije za svaku aplikaciju dohvate i pohrane, podaci se filtriraju kako bi se sve instalirane aplikacije mogle prikazati u grafičkom sučelju aplikacije. Filtrirani podaci su:

- **Ime aplikacije** - Dohvaća se koristeći *getApplicationLabel(appInfo)* metodu, gdje *appInfo* predstavlja pohranjene informacije za jednu aplikaciju. Potpuni prikaz se može vidjeti u sljedećem isječku koda 3.4:

```
    String appName = (String) packageManager.
   getApplicationLabel(appInfo);
```

Isječak koda 3.4 Dohvaćanje imena pojedine aplikacije

- **Ime paketa** - Dohvaća se naredbom *appInfo.packageName* gdje *appInfo* predstavlja pohranjene informacije za jednu aplikaciju. Potpuni prikaz se može vidjeti u sljedećem isječku koda 3.5:

```
PackageInfo pkgInfo = packageManager.getPackageInfo(appInfo
   .packageName, 0);
```

Isječak koda 3.5 Dohvaćanje paketa kojemu pripada aplikacija

- **Vrijeme instalacije** - Dohvaća se u milisekundama koristeći naredbu *pkgInfo.firstInstallTime* gdje *pkgInfo* predstavlja informacije o paketu kojem aplikacija pripada. Vrijeme se zatim pretvara u format "dan/mjesec/godina". Potpuni prikaz se može vidjeti u sljedećem isječku koda 3.6:

```
1     long firstInstallTime = pkgInfo.firstInstallTime;
2     Date installDate = new Date(firstInstallTime);
3     SimpleDateFormat dateFormat =
4         new SimpleDateFormat("dd/MM/yyyy", Locale.
    getDefault());
5     appInstallDate = dateFormat.format(installDate);
```

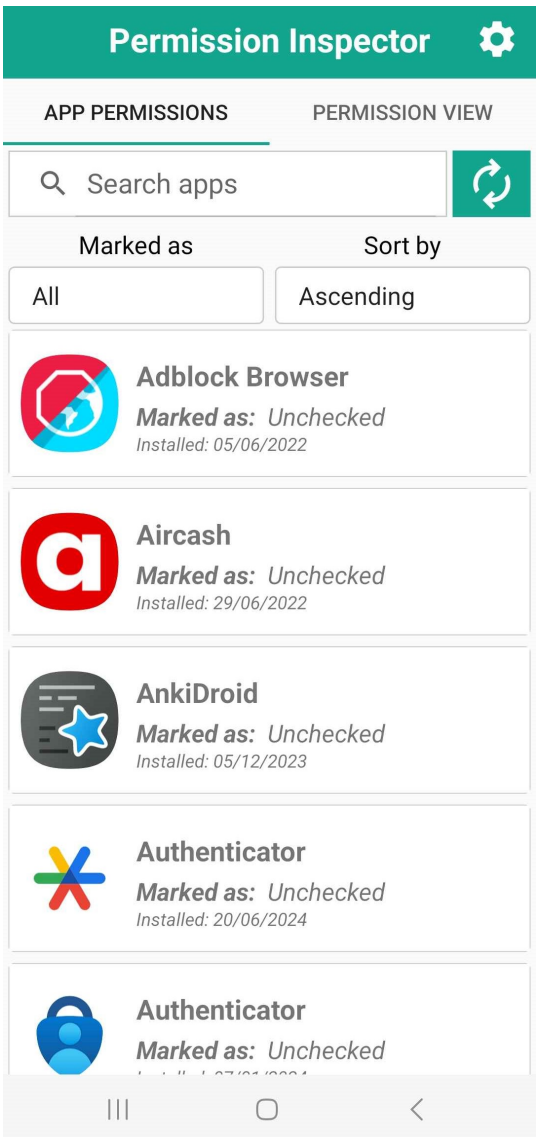
Isječak koda 3.6 Dohvaćanje vremena instalacije aplikacije te pretvorba u format "dan/mjesec/godina"

- **Ikona aplikacije** - Dohvaća se metodom *getApplicationIcon(appInfo)* gdje *appInfo* predstavlja pohranjene informacije za jednu aplikaciju. Potpuni prikaz se može vidjeti u sljedećem isječku koda 3.7:

```
1     Drawable appIcon = packageManager.getApplicationIcon
2         (appInfo);
```

Isječak koda 3.7 Dohvaćanje ikone aplikacije

Nakon dohvaćanja podataka iz *Android PMa*, aplikacija također dohvaća lokalno pohranjene informacije o sigurnosnom statusu svake instalirane aplikacije, koje je korisnik prethodno odredio. Ti podaci se pohranjuju lokalno putem *Android SharedPreferences* funkcionalnosti koja će detaljnije biti objašnjena u sljedećim poglavljima. Instalirane aplikacije se zatim prikazuju kao na slici 3.1:

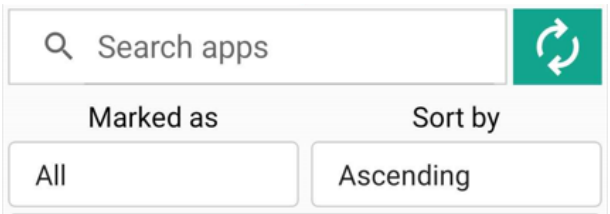


Slika 3.1 Prikaz svih instaliranih aplikacija

Na slici 3.1 se nalazi opcija pretraživanja aplikacije po imenu te mogućnost filtriranja aplikacija po sigurnosnom statusu, a ispod se mogu primijetiti aplikacije poredane abecednim redoslijedom. Klikom u područje koje predstavlja određenu aplikaciju otvorit će se detaljni pregled aplikacije koji između ostalog uključuje i dopuštenja koja je ta aplikacija zatražila. U sljedećim poglavljima detaljnije će se opisati funkcionalnost filtriranja aplikacija te sadržaj detaljnijeg pregleda aplikacije.

3.2.2 Opcije filtriranja aplikacija

Pri vrhu samog prozora za prikaz instaliranih aplikacija nalaze se opcije za pretraživanje i filtriranje instaliranih aplikacija, kao što je prikazano na slici 3.2. Instalirane



Slika 3.2 Opcije za pretraživanje i filtriranje instaliranih aplikacija

aplikacije je moguće filtrirati na sljedeća 3 načina:

- **Pretraživanje po imenu** - Filtriraju se one aplikacije koje u imenu sadrže uneseni tekstualni uzorak.
- **Filtriranje po sigurnosnom statusu** - Stupac "Marked as" predstavlja sigurnosni status kojim je korisnik označio određenu aplikaciju nakon ručnog pregleda dopuštenja koja aplikacija koristi. Predodređene opcije uključuju oznake: "Sve" (eng. *All*), "Provjerene" (eng. *Checked*), "Sumnjive" (eng. *Suspicious*), "Opasne" (eng. *Dangerous*) i "Neprovjerene" (eng. *Unchecked*).
- **Generalno filtriranje** - Sadrži opcije filtriranja prema najstarijim i najnovijim aplikacijama te silazno ili uzlazno redanje po abecedi.

Pored polja za pretraživanje nalazi se i tipka za osvježavanje informacija o aplikacijama s pomoću koje se može ručno pokrenuti ponovno dohvaćanje podataka za aplikacije ako su neki podaci zastarjeli te ih je potrebno ažurirati.

Filtriranje aplikacija prema spomenutim kriterijima provodi se u nekoliko koraka. Informacije se u trenutku promjene šalju u klasu *FilterUtils* koja sadrži metodu *filterApps* s pomoću koje se vrši filtriranje aplikacija. Aplikacije se prvo filtriraju s pomoću programskog koda koji je prikazan u sljedećem isječku koda 3.8:

```
1  if (query == null || query.isEmpty()) {
2      filteredAppList.addAll(appList);
3  } else {
4      String lowerCaseQuery = query.toLowerCase();
5      for (ApplicationInfo appInfo : appList) {
6          String appName = packageManager.getApplicationLabel(
       appInfo).toString().toLowerCase();
7          if (appName.contains(lowerCaseQuery)) {
8              filteredAppList.add(appInfo);
9          }
10     }
11 }
```

Isječak koda 3.8 Filtriranje prema imenu aplikacije

Nakon filtriranja prema imenu, aplikacije se zatim filtriraju prema sigurnosnom statusu kojim su označene. Programski kod filtriranja prema sigurnosnom statusu je prikazan u isječku koda 3.9.

```
if (markedAs != null && !markedAs.equalsIgnoreCase("All")) {
    List<ApplicationInfo> statusFilteredList = new ArrayList
   <>();
    for (ApplicationInfo appInfo : filteredAppList) {
        String appName = packageManager
            .getApplicationLabel(appInfo).toString();
        boolean shouldInclude = false;
        for (Map<String, String> map :
   applicationCheckedByUserList) {
            if (map.containsKey(appName)) {
                String status = map.get(appName);
                if(status == null) status = "Unchecked";
                if (status.equalsIgnoreCase(markedAs)) {
                    shouldInclude = true;
                    break;
                }
            }
        }
        if (shouldInclude) {
            statusFilteredList.add(appInfo);
        }
    }
    filteredAppList = statusFilteredList;
}
```

Isječak koda 3.9 Filtriranje prema sigurnosnom statusu aplikacije

Posljednji korak filtriranja obuhvaća sortiranje aplikacija prema vremenu instalacije ili abecednom redoslijedu imena. Programski kod koji izvršava ovo filtriranje treba biti shvaćen kao jedna cjelina, međutim radi jednostavnijeg prikaza podijeljen je u dvije cjeline. Isječak koda 3.10 prikazuje filtriranje prema vremenu instalacije aplikacija, dok isječak koda 3.11 prikazuje filtriranje prema o abecednom redoslijedu imena aplikacija.

```
if (sortOrder != null && (sortOrder.equalsIgnoreCase("Oldest")
        || sortOrder.equalsIgnoreCase("Newest"))) {
    Collections.sort(filteredAppList, new Comparator<
    ApplicationInfo>() {
        @Override
        public int compare(ApplicationInfo app1,
    ApplicationInfo app2) {
            try {
                PackageInfo pkgInfo1 = packageManager
                    .getPackageInfo(app1.packageName, 0);
                PackageInfo pkgInfo2 = packageManager
                    .getPackageInfo(app2.packageName, 0);
                long firstInstallTime1 =
                    pkgInfo1.firstInstallTime;
                long firstInstallTime2 =
                    pkgInfo2.firstInstallTime;
                Date date1 = new Date(firstInstallTime1);
                Date date2 = new Date(firstInstallTime2);
                int comparison = date1.compareTo(date2);
                return sortOrder.equalsIgnoreCase
                    ("Oldest") ? comparison : -comparison;
            } catch (PackageManager.NameNotFoundException e) {
                e.printStackTrace();
                return 0;
            }
        }
    });
}
```

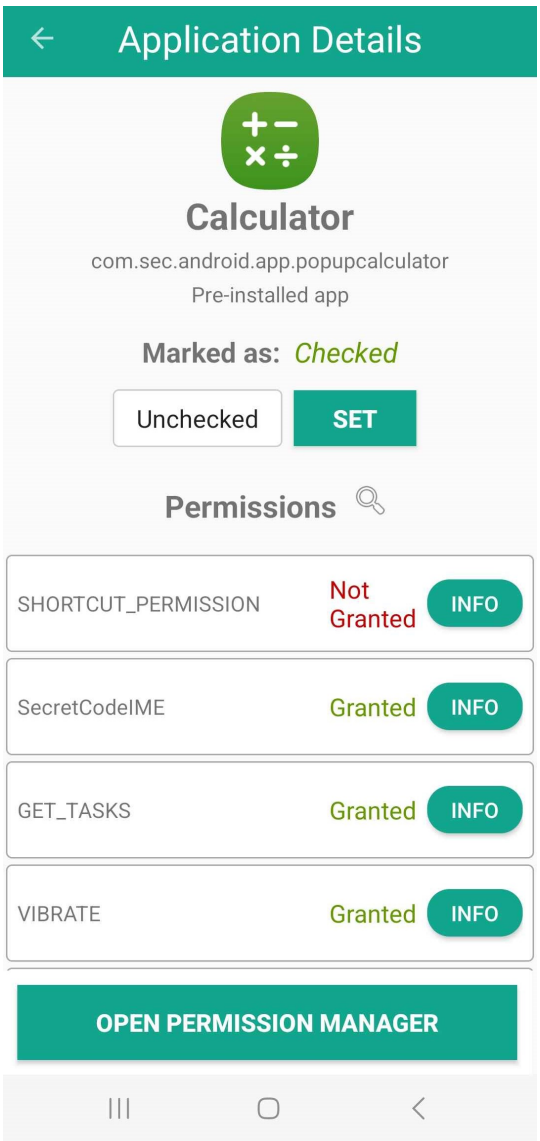
Isječak koda 3.10 Filtriranje prema vremenu instalacije aplikacija

```
else if (sortOrder.equalsIgnoreCase("Ascending")
        || sortOrder.equalsIgnoreCase("Descending")) {
    Collections.sort(filteredAppList,
            new Comparator<ApplicationInfo>() {
        @Override
        public int compare(ApplicationInfo app1,
                ApplicationInfo app2) {
            String appName1 = packageManager
                    .getApplicationLabel(app1).toString().
   toLowerCase();
            String appName2 = packageManager
                    .getApplicationLabel(app2).toString().
   toLowerCase();
            int comparison = appName1.compareTo(appName2);
            return sortOrder.equalsIgnoreCase
                    ("Ascending") ? comparison : -comparison;
        }
    });
}
```

Isječak koda 3.11 Filtriranje prema abecednom redoslijedu imena aplikacija

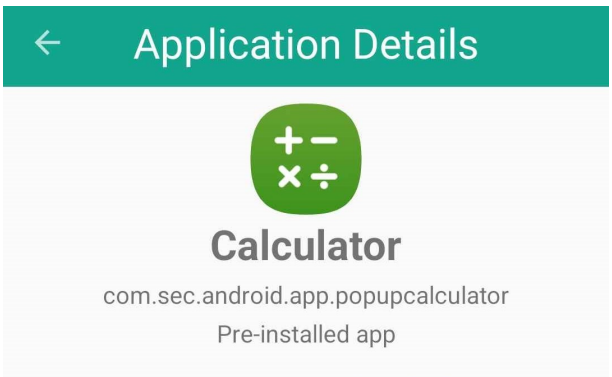
3.2.3 Detaljni prikaz aplikacije

Otvaranjem detaljnog prikaza aplikacije korisniku se nudi mogućnost pregleda pojedinih dopuštenja koja je ta aplikacija zatražila, a koja mogu ali ne moraju biti odobrena toj aplikaciji. Također, ispisuju se osnovne informacije o aplikaciji i postavljen sigurnosni status aplikacije. Za demonstraciju detaljnog prikaza aplikacije, koristi se standardna *Android* aplikacija "Kalkulator" (eng. *Calculator*). Slika 3.3 prikazuje izgled detaljnog prikaza aplikacije "Kalkulator".



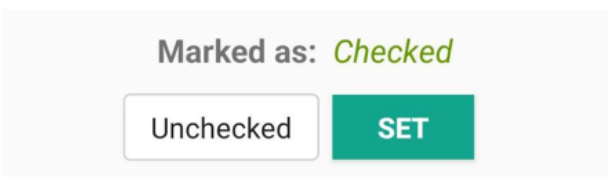
Slika 3.3 Detaljni prikaz aplikacije

Osnovne informacije o aplikaciji uključuju ikonu aplikacije, ime aplikacije, ime paketa te vrijeme instalacije aplikacije. Te se informacije tim redoslijedom mogu vidjeti na slici 3.4:



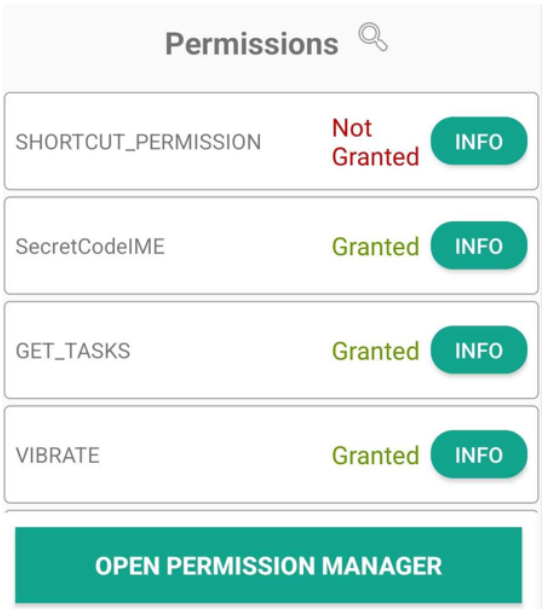
Slika 3.4 Osnovne informacije o aplikaciji

Ispod osnovnih informacija aplikacije nalazi se sigurnosni status aplikacije. Sigurnosni status odražava razinu sigurnosti aplikacije koju korisnik postavlja nakon ručnog pregleda dopuštenja. Za aplikaciju "Calculator" u primjeru je sigurnosni status označen kao "Checked", te je prikazan na slici 3.5. Ostatak detaljnog prikaza



Slika 3.5 Sigurnosni status aplikacije

aplikacije se sastoji od prikaza dopuštenja i opcije otvaranja *Androidovog* "Upravitelja dopuštenja" (eng. *Permission Manager*). Dopuštenja su prikazana u redovima, gdje jedan redak predstavlja jedno dopuštenje koje je aplikacija, u ovom slučaju "Calculator", zatražila. Svako prikazano dopuštenje sastoji se od imena dopuštenja, trenutnog stanja dopuštenja te mogućnosti prikaza dodatnih informacija o tom dopuštenju putem klika na tipku "INFO". Takav prikaz dopuštenja prikazan je na slici 3.6:



Slika 3.6 Prikaz dopuštenja instalirane aplikacije "Calculator"

Ime aplikacije predstavlja jedinstveni identifikator dopuštenja po kojem se dohvaćaju dodatne informacije o dopuštenju iz *Firestorea*. S desne strane imena dopuštenja nalazi se status o tom dopuštenju za trenutnu aplikaciju. Aplikacija može zatražiti dopuštenje, ali pravo na korištenje tog dopuštenja ne mora nužno biti odobreno stoga status dopuštenja može poprimiti vrijednosti "Granted" (hrv. *Odobreno*) ili "Not Granted" (hrv. *Nije odobreno*). Kako bi se prikazala dopuštenja promatrane aplikacije koristi se ime paketa promatrane aplikacije za koju pretražujemo dopuštenja. U isječku koda 3.12 prikazan je postupak dohvaćanja paketa kojemu aplikacija pripada s pomoću kojeg se mogu zatražiti sva dopuštenja koja je promatrana aplikacija zatražila.

```
1 PackageInfo packageInfo = packageManager.getPackageInfo
2                 (packageName, PackageManager.GET_PERMISSIONS);
3 String[] requestedPermissions =
4                 packageInfo.requestedPermissions;
```

Isječak koda 3.12 Dohvaćanje svih zatraženih dopuštenja promatrane aplikacije

Metodom *packageManager.getPackageInfo(packageName)* moguće je dohvatiti sve

podatke o specificiranom paketu, ali u ovom slučaju dodavanjem *PackageManager.GET_PERMISSIONS* kao drugi argument dohvaćaju se samo informacije o dopuštenjima koja taj paket koristi. Nakon što se dohvate sve informacije o svim dopuštenjima, ime pojedinog dopuštenja se može dohvatiti metodom *.getName()*. Status svakog pojedinog dopuštenja se provjerava naredbom *PackageManager.PERMISSION_GRANTED* kako je prikazano u sljedećem isječku koda 3.13:

```
boolean isGranted = packageManager.checkPermission
                  (permission.getFullName(), packageName)
                  == PackageManager.PERMISSION_GRANTED;
permissionStatusTextView.setText
                 (isGranted ? "Granted" : "Not Granted");
```

Isječak koda 3.13 Provjera statusa dopuštenja za promatranu aplikaciju

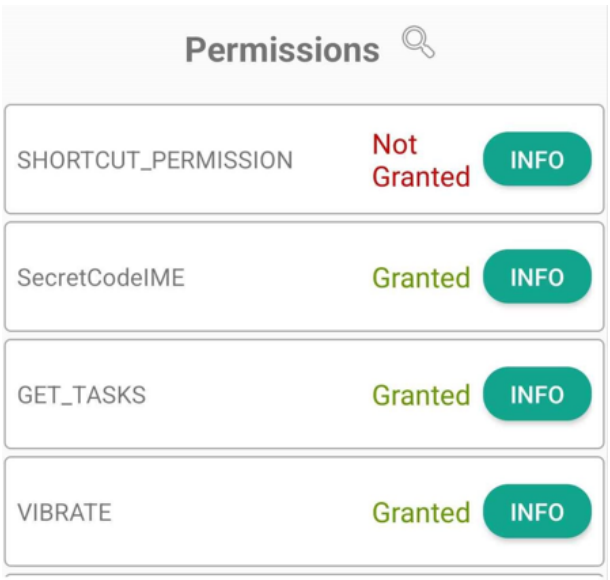
Prikupljene informacije svakog zatraženog dopuštenja se spremaju u novonastali objekt *Permission*, čija se metoda *fetchPermissionDataFromFirestore* koristi kako bi se dohvatili svi podaci o specifičnom dopuštenju koji su zapisani u *Firestoreu*. Ovi podaci o dopuštenju koriste se pri prikazu dodatnih informacija klikom na tipku "INFO", koja otvara novi prozor s detaljnim opisom dopuštenja. Programski kod koji za pojedinačno dopuštenje dohvaća podatke iz *Firestorea* je prikazan u sljedećem isječku koda 3.14:

```
public void fetchPermissionDataFromFirestore() {
    String truncatedPermissionName =
        extractTruncatedPermissionName(name);
    db.collection("permissions")
        .document(truncatedPermissionName).get()
        .addOnCompleteListener
            (new OnCompleteListener<DocumentSnapshot>() {
            @Override
             public void onComplete
                (@NonNull Task<DocumentSnapshot> task) {
                if (task.isSuccessful()) {
                    DocumentSnapshot document
                        = task.getResult();
                    if (document.exists()) {
                        description = document
                            .getString("Description");
                        danger = document
                            .getString("Danger Level");
                        reason = document
                            .getString("Reason for Danger");
                        group  = document
                            .getString("Group");
                    }
                }else {
                    Log.d(TAG, "Error getting documents: ",
                        task.getException());
                }
            }
        }
    );
}
```

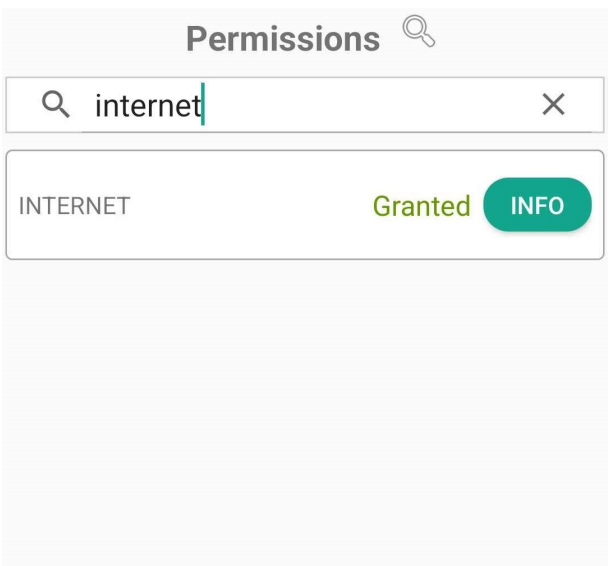
Isječak koda 3.14 Dohvaćanje informacije iz *Firestore* baze podataka za promatrano dopuštenje

Klikom na ikonicu povećala pokraj naslova "Permissions" (hrv. *Dopuštenja*) otvara

se mogućnost filtriranja po imenu zatraženih dopuštenja promatrane aplikacije. Slika 3.7 prikazuje pregled dopuštenja bez aktivnog filtra, dok se na slici 3.8 može vidjeti filtriranje zatraženih dopuštenja aplikacije s ključnom riječi "internet":



Slika 3.7 Prikaz dopuštenja bez aktivnog filtra



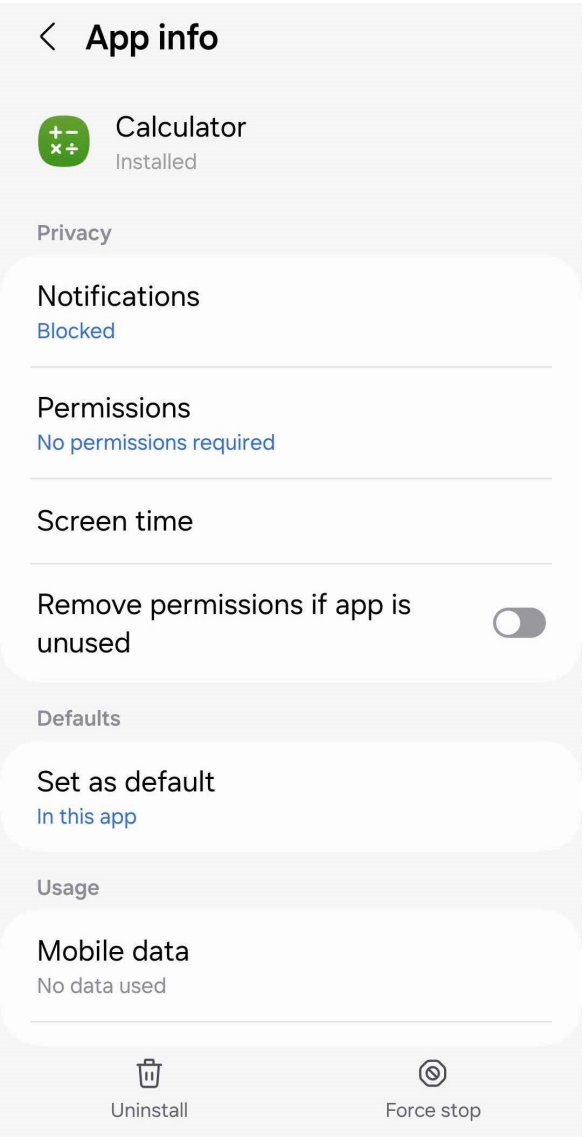
Slika 3.8 Prikaz filtriranih dopuštenja ključnom riječi "internet"

Programski kod za filtriranje dopuštenja po imenu je prikazan u sljedećem isječku koda 3.15:

```
public static List<Permission> filterPermissions(List<
    Permission> permissions, String query) {
     if (query == null || query.isEmpty()) {
         return new ArrayList<>(permissions);
     }
     List<Permission> filtered = new ArrayList<>();
     for (Permission permission : permissions) {
         if (permission.getName().toLowerCase().contains(query.
    toLowerCase())) {
             filtered.add(permission);
         }
     }
     return filtered;
}
```

Isječak koda 3.15 Filtriranje dopuštenja aplikacije po imenu

Na dnu prikaza detalja aplikacije još se nalazi tipka "Open Permission Manager" (hrv. *Pokreni upravitelja dopuštenja*) koja korisnika preusmjerava na *Android* aplikaciju "Upravitelj dopuštenja" ako želi oduzeti određena dopuštenja promatranoj aplikaciju. "Upravitelj dopuštenja" koristi se za jednostavno upravljanje određenim dopuštenjima aplikacija [7]. S pomoću "Upravitelja dopuštenja" moguće je ostvariti određenu kontrolu nad dopuštenjima aplikacije. Međutim, "Upravitelj dopuštenja" ne omogućava zabranu korištenja svih dopuštenja za promatranu aplikaciju, stoga je njegovo djelovanje ograničeno. U takvim slučajevima, preostaje jedino mogućnost uklanjanja aplikacije kako bi se oduzela određena dopuštenja. Na slici 3.9 prikazan je "Upravitelj dopuštenja" za aplikaciju "Calculator".



Slika 3.9 "Upravitelj dopuštenja" za aplikaciju "Calculator"

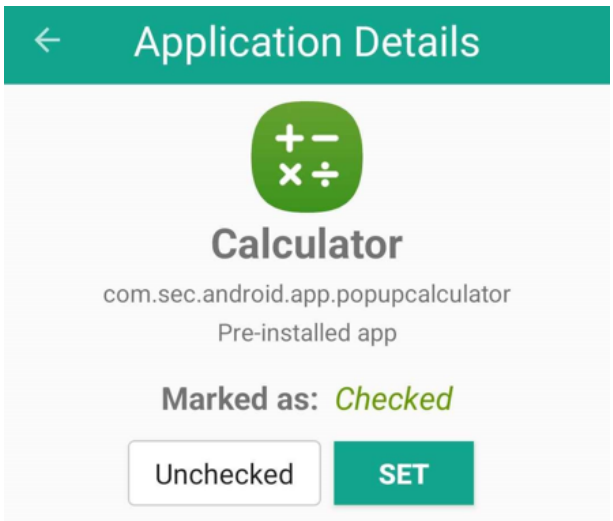
3.2.4 Postavljanje sigurnosnog statusa aplikacije

Postavljanje sigurnosnog statusa aplikacije omogućuje korisnicima da označe aplikacije prema njihovoj sigurnosti i potencijalnim prijetnjama koje predstavljaju. S obzirom na to da korisnik ima uvid u sva zatražena dopuštenja i u detaljni opis za svako dopuštenje, nakon pregleda pojedinih dopuštenja može iskoristiti mogućnosti

označavanja aplikacije kao jednu od četiri sigurnosne razine. Te sigurnosne razine su:

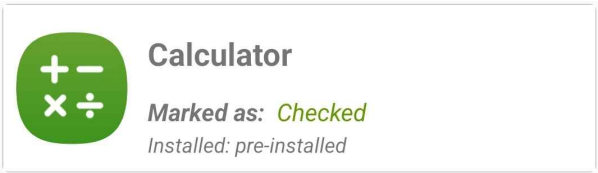
- ***Unchecked*** - Početno stanje svake instalirane aplikacije koje se postavlja pri prvom učitavanju aplikacije za pretragu potencijalno špijunskog softvera na mobilnim uređajima.
- ***Checked*** - Stanje predviđeno za aplikacije koje je korisnik ručno pregledao i utvrdio da ne postoji prijetnja.
- ***Suspicious*** - Stanje predviđeno za aplikacije koje korisnik nije u potpunosti pregledao ili koje predstavljaju potencijalnu prijetnju sigurnosti uređaja ili korisničkoj privatnosti.
- ***Dangerous*** - Stanje predviđeno za one aplikacije koje predstavljaju očitu prijetnju sigurnosti uređaja ili korisničkoj privatnosti.

Ova funkcionalnost je u potpunosti svojevoljna, te primarno omogućuje korisnicima da brzo identificiraju aplikacije koje mogu predstavljati rizik za sigurnost uređaja ili privatnost osobnih podataka. Nakon odabira sigurnosne razine, potrebno je pritisnuti tipku "SET" kako bi se pohranila izabrana vrijednost. Sljedeća slika 3.10 predstavlja primjer aplikacije "Calculator" koja je označena kao "Checked" u detaljima instalirane aplikacije.



Slika 3.10 Prikaz sigurnosnog statusa "Checked" aplikacije "Calculator" iz pogleda prikaza detalja aplikacije

Novo postavljeni sigurnosni status se također može iščitati iz glavnog prikaza instaliranih aplikacija, kako je prikazano na slici 3.11:



Slika 3.11 Prikaz sigurnosnog statusa aplikacije "Calculator" iz pogleda glavnog prikaza instaliranih aplikacija

Podaci o sigurnosnom statusu aplikacije spremaju se lokalno na uređaj korisnika koristeći *Android SharedPreferences* API (akronim od eng. *Application Programming Interface*: Programsko sučelje aplikacije). *SharedPreferences* omogućuje jednostavan i efikasan način za skladištenje podataka na uređaju, koji se pri zatvaranju aplikacije ili ponovnom pokretanju uređaja očuvaju [8]. Nakon što se prvi put dohvate sve instalirane aplikacije, one se označavaju s "Unchecked" te se spremaju lokalno na uređaj u parovima ključ-vrijednost gdje je ključ ime aplikacije a vrijednost sigurnosni status aplikacije. Isječak koda 3.16 provjerava ako postoje već spremljeni podaci o sigurnosnom statusu aplikaciju te ih potom učitava ako postoje, a ako ne postoje onda postavlja vrijednost "Unchecked" za sve dohvaćene aplikacije.

```
1  private void checkFirstLoadAppCheckedByUserList(List<
     ApplicationInfo> appList){
2      boolean firstLoadIsCompleted = preferences.getBoolean(
     FIRST_LOAD_APPS_CHECKED_BY_USER_KEY, false);
3      if(firstLoadIsCompleted){
4          applicationCheckedByUserList =
     getApplicationCheckedByUserData(getContext());
5      }else{
6          SharedPreferences.Editor editor = preferences.edit();
7          editor.putBoolean(FIRST_LOAD_APPS_CHECKED_BY_USER_KEY,
     true);
8          editor.apply();
9          for (PackageInfo packageInfo : packages) {
10             ApplicationInfo appInfo = packageInfo.
     applicationInfo;
11             Map<String, String> map = new HashMap<>();
12             String appName = (String) packageManager.
     getApplicationLabel(appInfo);
13             map.put(appName, "Unchecked");
14             applicationCheckedByUserList.add(map);
15             }
16         saveAppCheckedByUserList(getContext(),
     applicationCheckedByUserList);
17     }
18 }
```

Isječak koda 3.16 Provjera i dohvaćanje podataka o sigurnosnom statusu aplikacije

Ako ne postoje podaci, odnosno aplikacija se pokreće po prvi put, poziva se metoda *saveAppCheckedByUserList* koja sprema novonastale podatke. S obzirom na to da *SharedPreferences* ne podržava kompleksne oblike podataka, podaci se pretvaraju u *JSON* oblik podataka koji se može spremiti u primitivni tip podatka poput *Stringa*. Metoda *saveAppCheckedByUserList* je prikazana u isječku koda 3.17.

```
1 public static void saveAppCheckedByUserList(Context context,
  List<Map<String, String>> listOfMaps) {
2       SharedPreferences.Editor editor = preferences.edit();
3       Gson gson = new Gson();
4       String json = gson.toJson(listOfMaps);
5       editor.putString(APP_CHECKED_BY_USER_KEY, json);
6       editor.apply();
7    }
```

Isječak koda 3.17 Pohrana podataka o sigurnosno statusu aplikacije

Ako već postoje pohranjeni podaci, onda se poziva metoda *getApplicationChecked-ByUserData* koja dohvaća podatke u *JSON* obliku i pretvara ih u odgovarajući oblik. Programski kod ove metode prikazan je u isječku koda 3.18.

```
1 public static List<Map<String, String>>
  getApplicationCheckedByUserData(Context context) {
2       Gson gson = new Gson();
3       String json = preferences.getString(
  APP_CHECKED_BY_USER_KEY, null);
4       Type type = new TypeToken<List<Map<String, String>>>()
  {}.getType();
5       List<Map<String, String>> listOfMaps = gson.fromJson(
  json, type);
6       return listOfMaps == null ? new ArrayList<>() :
  listOfMaps;
7    }
```

Isječak koda 3.18 Dohvaćanje lokalno spremljenih podataka o sigurnosnom statusu aplikacija

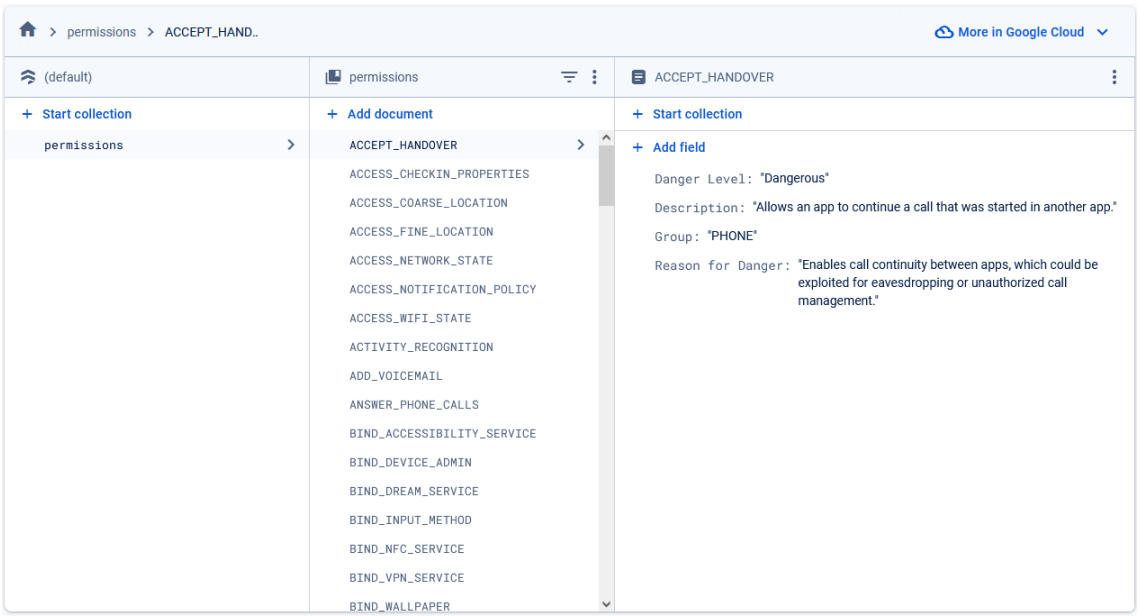
3.3 Pregled svih dopuštenja

U ovom poglavlju opisat će se druga temeljna funkcionalnost aplikacije za pretragu potencijalno špijunskog softvera na mobilnim uređajima, a to je pregled svih

dopuštenja koje aplikacije mogu zatražiti. Osim pregleda dopuštenja, moguć je i uvid u detalje svakog dopuštenja kako bi se korisnik mogao informirati o svrsi i potencijalnim prijetnjama pojedinačnih dopuštenja. Opisat će se i struktura pojedinog dokumenta *Firestore* baze podataka koji predstavlja jedno dopuštenje.

3.3.1 *Firestore* baza podataka

Kako bi se na efikasan i pristupačan način spremali podaci o dopuštenjima koje aplikacije mogu zatražiti, koristi se *Firestore* baza podataka u oblaku. U *Firestoreu* je izrađena kolekcija "permissions" koja sadrži dokumente koji predstavljaju dopuštenja. Slika 3.12 grafički prikazuje kolekciju "permissions" kojoj se može pristupiti putem *Firebase Google* konzole.



Slika 3.12 Grafički prikaz strukture baze podataka u *Google* konzoli

Dopuštenja su opisana kroz četiri atributa, a koja predstavljaju:

- **Razina opasnosti** (eng. *Danger Level*) - Opisuje razinu opasnosti koju dopuštenje predstavlja. Razine opasnosti definirao je *Android* razvojni tim, a za potrebe aplikacije opisane će biti dvije glavne razine: "Normalna" (eng. *Normal*) i "Opasna" (eng. *Dangerous*)

  - "Normalna" - Standardna vrijednost za dopuštenja niskog rizika koja služi za odobravanje pristupa izoliranim funkcijama na aplikacijskoj razini koje predstavljaju minimalnu opasnost za sigurnost uređaja ili korisnika. Ovaj tip dopuštenja se automatski odobrava aplikacijama koje ga zatraže bez potrebe posredovanja korisnika [9].
  - "Opasna" - Vrijednost koja opisuje dopuštenja visokog rizika. Ova dopuštenja omogućuju aplikaciji pristup osobnim podacima korisnika ili omogućuju kontrolu nad funkcijama koje mogu negativno utjecati na sigurnost uređaja. S obzirom na to da ovaj tip dopuštenja predstavlja veći rizik od običnih, "Normalnih" dopuštenja, u čestim slučajevima potrebno je odobrenje samog korisnika uređaja kako bi se odobrilo korištenje tog dopuštenja [9].

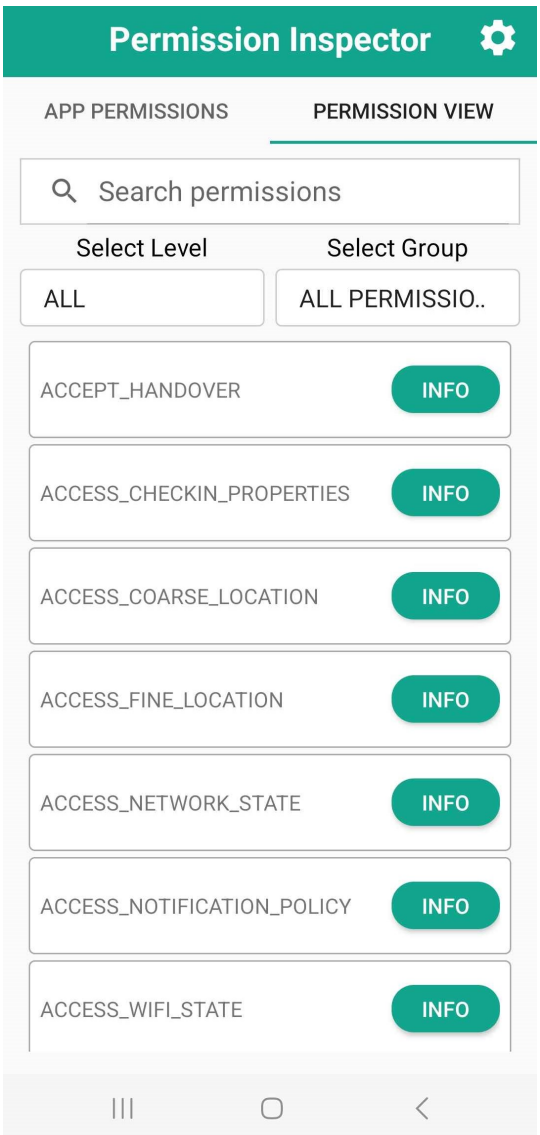
- **Opis** (eng. *Description*) - Opis dopuštenja sadrži kratak opis svrhe tog dopuštenja kako bi se korisnik mogao upoznati s funkcijom.
- **Grupacija** (eng. *Group*) - Grupacija se sastoji od 16 grupa često korištenih ili pretraživanih dopuštenja. Grupe su definirane od strane *Androida* [10], a u aplikaciji koriste isključivo za pomoć korisniku pri pretrazi specifičnih dopuštenja. Te grupe su:
  - CALENDAR (hrv. *Kalendar*) - Sadrži dopuštenja koja omogućuju pristup za čitanje i pisanje podataka iz kalendara.
  - CALL_LOG (hrv. *Zapis_poziva*) - Sadrži dopuštenja koja omogućuju pristup za čitanje i pisanje zapisa poziva (tj. povijesti poziva)
  - CAMERA (hrv. *Kamera*) - Sadrži dopuštenja koja omogućuju pristup kameri uređaja za snimanje fotografija i videozapisa
  - CONTACTS (hrv. *Kontakti*) - Sadrži dopuštenja koja omogućuju pristup za čitanje, pisanje i upravljanje kontaktima.
  - LOCATION (hrv. *Lokacija*) - Sadrži dopuštenja koja omogućuju pristup lokaciji uređaja.
  - MICROPHONE (hrv. *Mikrofon*) - Sadrži dopuštenja koja omogućuju pristup za snimanje zvuka pomoću mikrofona uređaja.

  - PHONE (hrv. *Telefon*) - Sadrži dopuštenja koja omogućuju pristup za obavljanje poziva, čitanje stanja telefona i upravljanje pozivima.
  - SENSORS (hrv. *Senzori*) - Sadrži dopuštenja koja omogućuju pristup podacima sa senzora, isključujući mikrofon. Senzori mogu biti kompas, akcelerometar, žiroskop i tako dalje.
  - SMS (hrv. *Poruke*) - Sadrži dopuštenja koja omogućuju pristup za slanje, primanje i čitanje *SMS* poruka.
  - STORAGE (hrv. *Skladište*) - Sadrži dopuštenja koja omogućuju pristup za čitanje i pisanje na vanjskoj pohrani uređaja.
  - ACTIVITY_RECOGNITION (hrv. *Prepoznavanje_aktivnosti*) - Sadrži dopuštenja koja omogućuju pristup podacima o fizičkoj aktivnosti za aplikacije za fitnes (npr. hodanje, trčanje).
  - BLUETOOTH - Sadrži dopuštenja koja omogućuju pristup za uparivanje s *Bluetooth* uređajima i kontrolu *Bluetooth* postavki.
  - NEARBY_DEVICES (hrv. *Obližnji_uređaji*) - Sadrži dopuštenja koja omogućuju interakciju i otkrivanje obližnjih uređaja.
  - PHYSICAL_ACTIVITY (hrv. *Fizička_aktivnost*) - Slično kao ACTIVITY_RECOGNITION, sadrži dopuštenja koja omogućuju pristup podacima vezanim za fizičke aktivnosti korisnika.
  - READ_MEDIA (hrv. *Čitanje_medija*) - Sadrži dopuštenja koja omogućuju pristup za čitanje raznih vrsta medijskih datoteka, kao što su slike, videozapisi ili audio datoteke.
  - POST_NOTIFICATIONS (hrv. *Obavijesti*) - Sadrži dopuštenja koja omogućuju aplikacijama da postavljaju obavijesti u ime korisnika.

- **Razlog za opasnost** (eng. *Reason for Danger*) - Ako je razina određenog dopuštenja "Opasna" onda se ovaj atribut koristi kako bi se korisniku objasnila potencijalna prijetnja koju to dopuštenje predstavlja.

3.3.2 Pregled po dopuštenjima

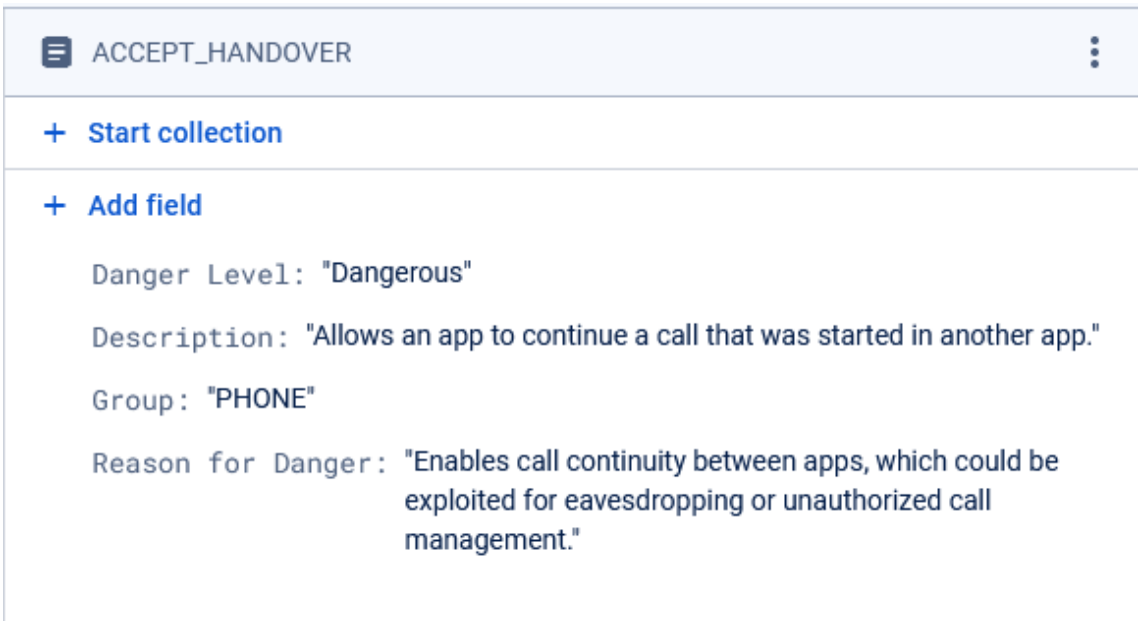
Pregled po dopuštenjima omogućuje pregled po većini dopuštenja koje se mogu zatražiti. Dopuštenja se učitavaju iz *Firestorea* i prikazuju korisniku. Slika 3.13 prikazuje izlistana dopuštenja bez aktivnog filtra:



Slika 3.13 Prikaz svih dopuštenja iz *Firestore* baze podataka bez aktivnog filtra

Dopuštenja su određena jedinstvenim identifikatorom koji je ujedno i ime dopu-

štenja, a koji služi i kao identifikator odgovarajućeg dokumenta u kolekciji *Firestore* baze podataka. Na slici 3.14 grafički je prikazana struktura dopuštenja "ACCEPT_HANDOVER" u *Firebase Google* konzoli. Kako bi se dopuštenja prikazala u



Slika 3.14 Grafički prikaz strukture dokumenta u *Firebase Google* konzoli koji predstavlja dopuštenje "ACCEPT_HANDOVER"

aplikaciji, potrebno ih je dohvatiti iz *Firestorea*. Pomoću sljedećeg isječka koda 3.19 dopuštenja se programski dohvaćaju iz baze podataka.

```
private void fetchPermissionsFromFirestore() {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    db.collection("permissions").get()
        .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
        permissionsList.clear();
        filteredPermissionsList.clear();
            for (QueryDocumentSnapshot document : task.
  getResult()) {
                String name = document.getId();
                String description = document
                    .getString("Description");
                String danger = document
                    .getString("Danger Level");
                String reason = document
                    .getString("Reason for Danger");
                String group = document
                    .getString("Group");
                Permission permission = new Permission
                    (name, description, danger, reason, group);
                permissionsList.add(permission);
            }
            filteredPermissionsList.addAll(permissionsList);
            adapter.notifyDataSetChanged();
        } else {
            Log.e("Firestore", "Error getting documents: ",
                task.getException());
        }
    });
}
```

Isječak koda 3.19 Dohvaćanje dopuštenja iz *Firestore* baze podataka

Ovaj programski kod dohvaća svako dopuštenje spremljeno u *Firestoreu* koja se zatim mogu dodatno filtrirati ovisno o potrebi korisnika. Također, klikom na tipku "INFO" otvara se novi prozor koji sadrži informacije o pojedinom dopuštenju. Ove

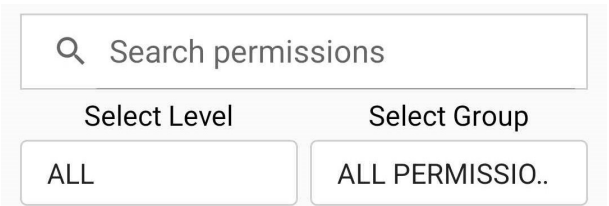
funkcionalnosti bit će detaljnije opisane u sljedećim poglavljima.

3.3.3 Opcije filtriranja dopuštenja

Kao i kod filtriranja aplikacija, postoji nekoliko opcija i za filtriranje dopuštenja. Opcije za filtriranje dopuštenja nalaze se na samom vrhu prozora za pregled po dopuštenjima, a one su:

- **Pretraživanje po tekstualnom uzorku** - Filtriraju se ona dopuštenja koja u imenu, opisu, razini opasnosti i razlogu za opasnost sadrže uneseni tekstualni uzorak.
- **Filtriranje po razini opasnosti** - Filtriranje dopuštenja prema grupi za razinu opasnosti kojoj pripada. Razine opasnosti dijele se na "Normalne" i "Opasne". Ako se odabere "ALL" prikazuju se sva dopuštenja neovisno o razini opasnosti.
- **Filtriranje po grupama** - Dopuštenja su podijeljena u predefiniranih 16 grupa radi jednostavnosti pretrage učestalih dopuštenja. Ona dopuštenja koja ne spadaju u ove grupe pretražuju se putem grupe "OTHER" (hrv. *Ostalo*) ili "ALL PERMISSIONS" (hrv. *Sva dopuštenja*).

Slika 3.15 prikazuje opcije filtriranja dopuštenja. Na vrhu je opcija pretraživanja prema tekstualnom uzorku dok se ispod redom nalaze opcije za filtriranje po razini opasnosti, označena s "Select Level" (hrv. *Odaberi razinu*) i opcija za filtriranje po grupama, označena s "Select Group" (hrv. *Odaberi grupu*). Programski kod koji



Slika 3.15 Opcije za pretraživanje i filtriranje dopuštenja

se koristi za filtriranje dopuštenja bit će prikazan u nekoliko kodnih isječaka radi jednostavnosti, ali se treba shvatiti kao jedna cjelina. Prvi kodni isječak 3.20 filtrira

dopuštenja ovisno o tekstualnom uzorku.

```
boolean matchesQuery = query.isEmpty() ||
    permissionName.contains(query.toLowerCase()) ||
    permissionDescription.contains(query.toLowerCase()) ||
    permissionDanger.contains(query.toLowerCase()) ||
    permissionReason.contains(query.toLowerCase());
```

Isječak koda 3.20 Filtriranje dopuštenja po tekstualnom uzorku

Sljedeći krug filtriranja prikazan je u kodnom isječku 3.21, a predstavlja programsko rješenje za filtriranje po razini opasnosti.

```
boolean matchesDangerLevel = isAllDangerLevel ||
    (isOtherDangerLevel && !allDangerLevels.contains(
   permissionDanger)) ||
    permissionDanger.equals(selectedDangerLevel);
```

Isječak koda 3.21 Filtriranje dopuštenja ovisno o razini opasnosti

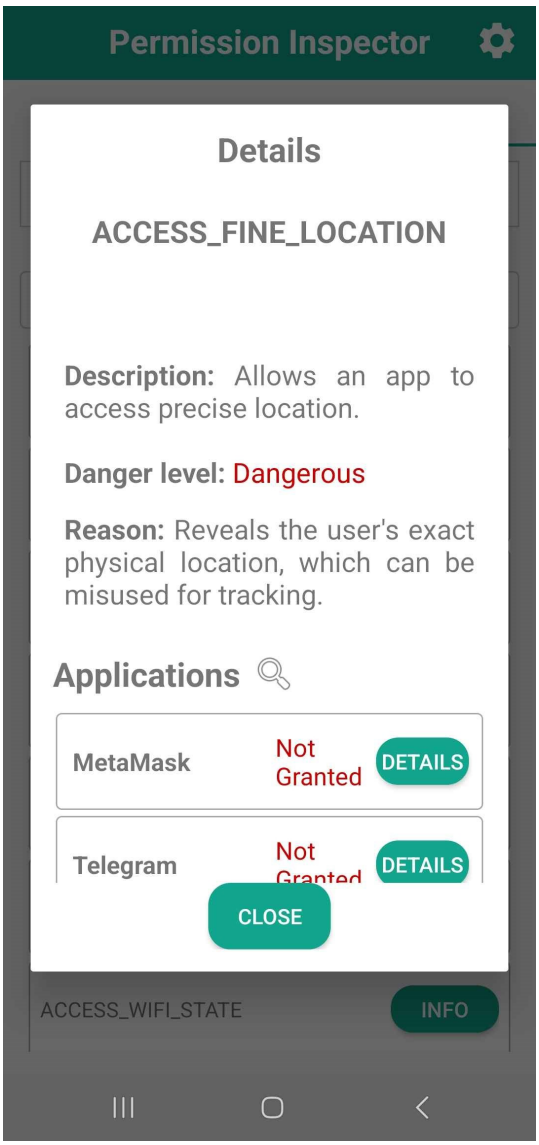
Zadnji isječak koda 3.22 prikazuje filtriranje prema grupi kojoj dopuštenje pripada.

```
boolean matchesGroup = isAllGroup ||
    (isOtherGroup && !allGroups.contains(permissionGroup)) ||
    permissionGroup.equals(selectedGroup);
```

Isječak koda 3.22 Filtriranje dopuštenja po pripadajućoj grupi

3.3.4 Detaljni prikaz dopuštenja

Detaljni prikaz dopuštenja sadrži informacije o pojedinom dopuštenju, a moguće im je pristupiti klikom na tipku "INFO" za promatrano dopuštenje. Informacije vezane uz promatrano dopuštenje prikazuju se u novom prozoru, kao što je prikazano na slici 3.16.



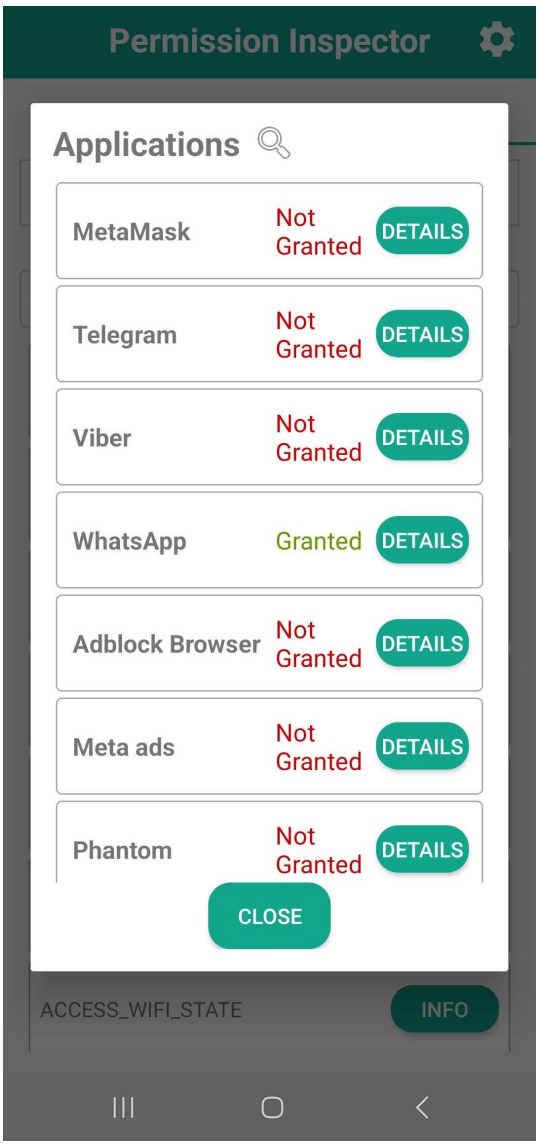
Slika 3.16 Detaljni prikaz informacija promatranog dopuštenja

Otvaranjem detaljnog prikaza, korisnik dobiva pregled informacija o promatranom dopuštenju, koje uključuju:

- **Ime dopuštenja** - Jedinstveni identifikator *Firestore* dokumenta.
- **Opis** - Kratak opis svrhe tog dopuštenja kako bi se korisnik mogao upoznati s funkcijom.

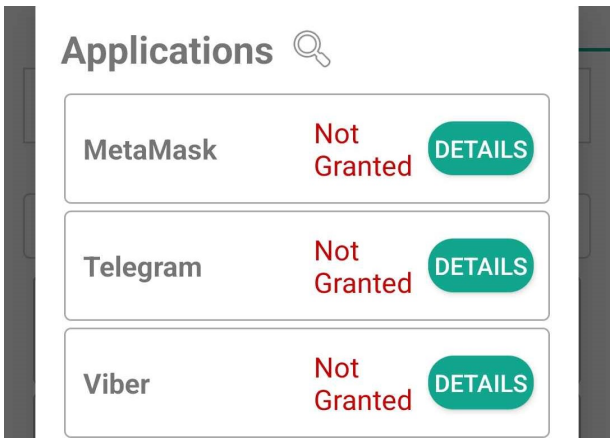
- **Razina opasnosti** - Opisuje razinu opasnosti kojoj dopuštenje pripada.
- **Razlog (eng. *Reason*)** - Ako je razina opasnosti promatranog dopuštenja označena s "Dangerous", opisuje se razlog zbog kojeg se promatrano dopuštenje smatra opasnim.
- **Aplikacije (eng. *Applications*)** - Predstavlja ispis svih aplikacija na uređaju korisnika koje također koriste ili su zatražile promatrano dopuštenje.

U listi aplikacija, jedan redak predstavlja jednu aplikaciju a sastoji se redom od imena aplikacije, statusa promatranog dopuštenja za tu aplikaciju te tipke "DETAILS" (hrv. *Detalji*). Status promatranog dopuštenja za aplikaciju poprima vrijednosti "Granted" ili "Not Granted" ovisno o tome ako je promatranoj aplikaciji odobreno korištenje promatranog dopuštenja. Slika 3.17 prikazuje listu aplikacija koje su zatražile promatrano dopuštenje. Klikom na tipku "DETAILS" otvara se detaljni prikaz aplikacije, kakav je opisan u poglavlju 3.2.3.

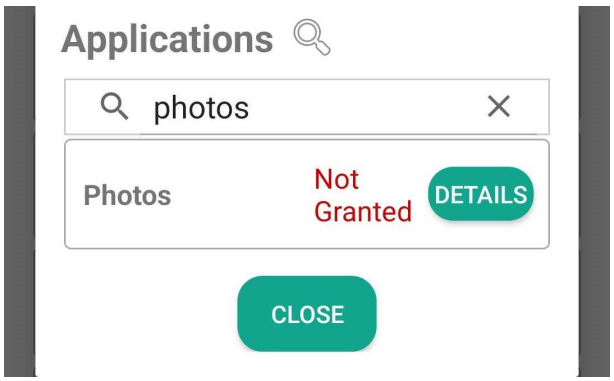


Slika 3.17 Lista aplikacija koje su zatražile promatrano dopuštenje

Klikom na ikonicu povećala pokraj naslova "Applications" otvara se mogućnost filtriranja aplikacija po imenu, kako je prikazano na slici 3.18. i 3.19.



Slika 3.18 Neaktivan filtar za aplikacije koje su zatražile promatrano dopuštenje



Slika 3.19 Pretraživanje aplikacija koje su zatražile promatrano dopuštenje sa tekstualnim uzorkom "photos"

Programski kod za filtriranje aplikacije prema imenu za promatrano dopuštenje je prikazan u kodnom isječku 3.23. Uneseni tekstualni uzorak prosljeđuje se u klasu *FilterUtils* koja sadrži metodu *filterAppsInDialog* s pomoću koje se izvršava filtriranje.

```
String lowerCaseQuery = query.toLowerCase();
for (PermissionInfoDialogFragment.AppInfo appInfo : appList) {
    String appName = appInfo.getAppName().toString().
   toLowerCase();
    if (appName.contains(lowerCaseQuery)) {
        if (!appInfo.isSystemOrHiddenApp() ||
   showSystemOrHiddenApps) {
            filteredAppList.add(appInfo);
        }
    }
}
```

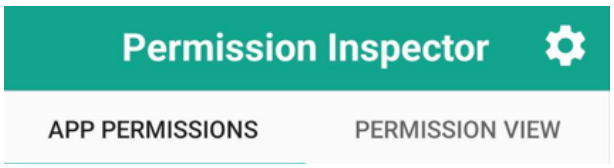
Isječak koda 3.23 Filtriranje aplikacija promatranog dopuštenja prema imenu aplikacija

3.4 Sustavne aplikacije

Sustavne (eng. *System*) aplikacije se mogu prikazati pomoću aplikacije za pretragu potencijalno špijunskog softvera na mobilnim uređajima.

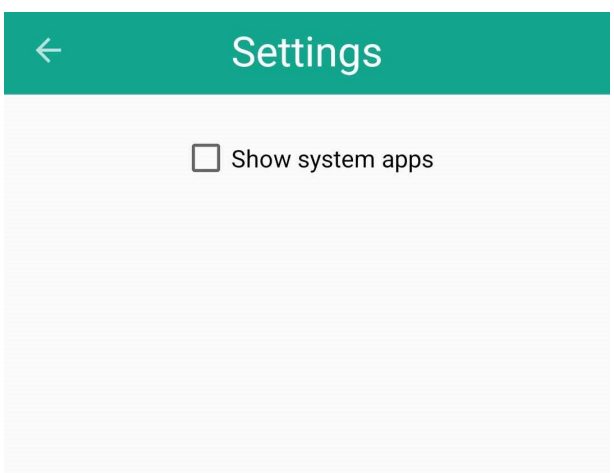
Sustavne aplikacije su aplikacije koje su instalirane uz operacijski sustav [11]. Ove aplikacije vrlo često izvršavaju kritične funkcije bez kojih uređaj ili operacijski sustav ne bi funkcionirao. Jednostavno ih je opisati kao aplikacije koje se nalaze u "/system/app" direktoriju na *Android* uređaju. Običan korisnik *Android* uređaja obično nema ovlaštenja za mijenjanje dopuštenja ovih aplikacija ili njihovo uklanjanje s uređaja. Primjer takvih aplikacija su standardne aplikacije poput kamere, postavka, poruka i mnoge druge.

Ako se žele pregledati sustavne aplikacije, potrebno je uključiti mogućnost prikazivanja. Ta se mogućnost nalazi u postavkama aplikacije koje se nalaze na vrhu prozora u gornjem desnom kutu. Klikom na ikonicu zupčanika kako je prikazano na slici 3.20 otvaraju se postavke aplikacije za pretragu potencijalno špijunskog softvera na mobilnim uređajima.



Slika 3.20 Prikaz ikonice zupčanika za otvaranje postavki

Pri otvaranju postavki potrebno je odobriti prikaz sustavnih aplikacija klikom na okvir za izbor (eng. *Checkbox*) kako je prikazano na slici 3.21. Promjene u postavkama se automatski spremaju te pri ponovnom učitavanju aplikacija prikazat će se i sustavne aplikacije.



Slika 3.21 Postavke aplikacije

Kada se u postavkama odobri prikazivanje sustavnih aplikacija, koristeći *Android SharedPreferences* API postavke se pohranjuju lokalno na uređaj korisnika. Isječak koda 3.24 dohvaća vrijednost pohranjenu u *checkbox_ all_ apps_ status* koja označava ako je potrebno učitati sustavne aplikacije.

```
private static final String CHECKBOX_KEY =
        "checkbox_all_apps_status";
    ...
    preferences = getActivity().getSharedPreferences(PREFS_NAME
   , Context.MODE_PRIVATE);
    boolean isChecked = preferences.getBoolean(CHECKBOX_KEY,
   false);
```

Isječak koda 3.24 Dohvaćanje vrijednosti varijable *checkbox_all_apps_status*

Kada je potrebno učitati aplikacije, neovisno da li se aplikacije učitavaju u glavnom prikazu svih aplikacija ili u detaljnom prikazu dopuštenja, uvijek se koristi programski kod prikazan u isječku koda 3.25.

```
for (PackageInfo packageInfo : packages) {
    ApplicationInfo appInfo =
        packageInfo.applicationInfo;
    if (!isChecked) {
        if ((appInfo.flags
            & ApplicationInfo.FLAG_SYSTEM) != 0 ||
                (appInfo.flags &
                    ApplicationInfo.FLAG_UPDATED_SYSTEM_APP) !=
    0) {
            continue;
        }
    }
    appList.add(appInfo);
}
```

Isječak koda 3.25 Provjera vrste instaliranih aplikacija

Prikazani programski kod koristi integrirane zastavice (eng. *Flag*) koje označuju ako je određena aplikacija sustavna. *Android* definira mogućnosti korištenja raznih zastavica, a u ovom slučaju, zastavice koje označavaju sustavne aplikacije su:

- FLAG_SYSTEM - Označava da je aplikacija sustavna aplikacija. S obzirom na to da sustavne aplikacije imaju veće privilegije od običnih aplikacija, *Android*

sustav ih označava s ovom zastavicom [12].

- FLAG_UPDATED_SYSTEM_APP - Ova zastavica služi kako bi *Android* sustav mogao prepoznati ako je sustavna aplikacija ažurirana u nekom trenutku te više nije identična izvornoj inačici aplikacije [12].

Poglavlje 4

Zaključak

S rastućim brojem novih aplikacija za mobilne uređaje, dopuštenja koje mogu zatražiti su sve mnogobrojnija i svestranija. Dopuštenja mogu biti osnovna, predstavljajući minimalni rizik za korisnika, ili mogu biti takva da značajno ugrožavaju sigurnost uređaja i privatnih podataka korisnika.

Kako bi se najgore izbjeglo, potrebno je informirati korisnika o potencijalnim opasnostima pojedinih dopuštenja te mu također ponuditi alat za provjeru dopuštenja već instaliranih aplikacija na njegovom uređaju. Aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima uspješno omogućava te funkcionalnosti. Korisnik može pregledati sve instalirane aplikacije i detaljno analizirati dopuštenja koja one zahtijevaju. Uz pomoć unaprijed definirane *Firestore* baze podataka, također mu je omogućen pristup korisnim informacijama o raznovrsnim dopuštenjima. Ako korisnik smatra određenu aplikaciju opasnom, aplikacija nudi mogućnost označavanja pregledanih aplikacija te jednostavan pristup "Upravitelju dopuštenja" za oduzimanje određenih dopuštenja.

Međutim, aplikacija je ograničena na oduzimanje samo onih dopuštenja koja se mogu povući putem *Androidovog* "Upravitelja dopuštenja". Također, potrebno je održavati i periodično ažurirati *Firestore* bazu podataka kako bi korisnik uvijek imao pristup najaktualnijim informacijama o raznovrsnim dopuštenjima.

Unatoč nedostacima, aplikacija za pretragu potencijalno špijunskog softvera na mobilnim uređajima uspješno i na jednostavan način omogućava prosječnom korisniku *Android* mobilnog uređaja detaljni uvid u instalirane aplikacije na njegovom uređaju.

Bibliografija

[1] A. Khatoon i P. Corcoran, "Android permission system and user privacy — A review of concept and approaches", IEEE 7th International Conference on Consumer Electronics - Berlin, 2017.

[2] R Fedler, C. Banse, C. Krauss i V. Fusenig, "Android OS Security, Risks and Limitations A Practical Evaluation", Fraunhofer AISEC, 2012.

[3] A. Developers, Meet Android Studio, kolovoz 2024., url adresa: https://developer. android.com/studio/intro.

[4] Google, Cloud Firestore, kolovoz 2024., url adresa: https://firebase.google.com/products/firestore.

[5] Google, Firestore, kolovoz 2024., url adresa: https://cloud.google.com/firestore.

[6] A. Developers, Manifest.permission, kolovoz 2024., url adresa: https://developer.android.com/reference/android/Manifest.permission# QUERY_ALL_PACKAGES.

[7] Google, Change app permissions on your Android phone, kolovoz 2024., url adresa: https://support.google.com/android/answer/9431959?hl=en.

[8] A. Developers, Save simple data with SharedPreferences, kolovoz 2024., url adresa: https://developer.android.com/training/data-storage/shared-preferences.

[9] A. Developers, <permission>, kolovoz 2024., url adresa: https://developer.android.com/ guide/topics/manifest/permission-element.

[10] A. Developers, Manifest.permission_groups, kolovoz 2024., url adresa: https://developer.android.com/reference/android/Manifest.permission_group.

[11] B. Sharma, System App In Android, kolovoz 2024., url adresa: https://medium.com/android-news/system-app-in-android-f003d418b4cc.

[12] A. Developers, ApplicationInfo, kolovoz 2024., url adresa: https://developer.android.com/reference/android/content/pm/ApplicationInfo.

Sažetak

Zahtjevi mobilnih aplikacija za dopuštenjima postaju sve raznovrsniji i složeniji. Ta dopuštenja mogu varirati od onih s minimalnim rizikom do onih koji značajno ugrožavaju sigurnost uređaja ili privatnih podataka korisnika. Kako bi se smanjili rizici, važno je informirati korisnike o potencijalnim opasnostima pojedinih dopuštenja i pružiti im alat za pregled dopuštenja aplikacija koje su već instalirane na njihovom uređaju. Razvijena aplikacija omogućava korisnicima da analiziraju sve instalirane aplikacije i pregledaju dopuštenja svake od njih. Korištenjem unaprijed definirane *Firestore* baze podataka, aplikacija pruža pristup korisnim informacijama o raznim dopuštenjima. Korisnici mogu označiti aplikacije koje smatraju opasnima i pristupiti "Upravitelju dopuštenja" za oduzimanje dopuštenja. Aplikacija nudi jednostavan način za detaljan uvid u dopuštenja instaliranih aplikacija na Android uređajima.

Ključne riječi — **dopuštenja, aplikacije, pregled, opasnost**

Abstract

The range of application permission requests has become more diverse and complex. These permissions can vary from those posing minimal risk to those that can significantly compromise the security of the device or the user's private data. To mitigate these risks, it is crucial to inform users about the potential dangers of specific permissions and provide a tool for checking the permissions of apps already installed on their device. The developed application allows users to analyze all installed apps and review the permissions for each app. Utilizing a pre-defined *Firestore* database, the application provides access to useful information about various permissions. Users can mark apps they consider dangerous and access the "Permission Manager" to revoke permissions. The application offers a straightforward way to gain detailed insights into the permissions of installed apps on Android devices.

Keywords **— permissions, applications, review, danger**