Analiza sigurnosti operacijskog sustava Android

Štefanec, Mia

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ANALIZA SIGURNOSTI OPERACIJSKOG SUSTAVA ANDROID

Završni rad

Mia Štefanec

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 16.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Mia Štefanec
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3992, 24.09.2019.
OIB studenta:	66579753916
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Izv. prof. dr. sc. Krešimir Grgić
Sumentor iz tvrtke:	
Naslov završnog rada:	Analiza sigurnosti operacijskog sustava Android
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena 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 mentora:	16.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Mia Štefanec
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3992, 24.09.2019.
Turnitin podudaranje [%]:	7

Ovom izjavom izjavljujem da je rad pod nazivom**: Analiza sigurnosti operacijskog sustava Android**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Izv. prof. dr. sc. Krešimir Grgić

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

Broj mobilnih aplikacija razvijenih za rad na operacijskom sustavu Android svake godine je sve veći, a time je i mogućnost narušavanja sigurnosti i otjecanja podataka sve veća zato što osobe koje mogu neovlašteno pristupiti podatcima postaju sve kreativnije kako bi ostvarile svoj cilj. Korisnici nisu ni svjesni da bi njihovi podatci mogli biti zloupotrijebljeni ako nesvjesno instaliraju zlonamjernu aplikaciju ili aplikaciju koja koristi nisku razinu šifriranja podataka koja bi mogla zaobići ili probiti sve sigurnosne mehanizme koje operacijski sustav Android ima implementirane. U operacijskom sustavu Android stalno se pokušava poboljšati sigurnosna okolina kako bi se spriječilo moguće otjecanje podataka ili bilo kakav drugačiji pokušaj narušavanja sigurnosti.

U ovom završnom radu naglasak je stavljen na dva sigurnosna problema sigurnosne okoline operacijskog sustava Android, a to su problem otjecanja podataka do kojeg može doći instaliranjem zlonamjerne aplikacije kojoj korisnik daje dopuštenje u uvid podataka kojoj ti podatci nisu nužni, te problemi koji nastaju ako aplikacija koristi nisku razinu šifriranja podataka ili uopće ne koristi šifriranje.

Sigurnosna okolina operacijskog sustava Android objašnjena je u drugom poglavlju, u kojem će također biti prikazano trenutnog stanja sigurnosti na mobilnim uređajima i pregled postojećih rješenja. U trećem poglavlju ovoga rada detaljno je opisana kriptografija i metode šifriranja koje se najčešće koriste za zaštitu podataka u Androidu, a u četvrtom poglavlju definirane su i opisane vrste prijetnji u Android sustava. Peto i šesto poglavlje definiraju probleme otjecanja podataka i loših kriptografskih rješenja te se za svaki problem opisuju rješenja. U sedmom poglavlju također analiziraju problem iz prethodna dva poglavlja te ih prikazuje i implementira najbolje rješenje.

1.1. Zadatak završnog rada

U završnom radu treba analizirati i opisati sigurnosnu okolinu operacijskog sustava Android, uključujući sigurnosne prijetnje i mehanizme zaštite. Posebno treba istaknuti probleme otjecanja podataka uzrokovano zlonamjernim mobilnim aplikacijama, te probleme uzrokovane niskom razinom primijenjenih postupaka šifriranja podataka. Nadalje, treba razraditi model sigurnosnih prijetnji i mehanizama zaštite za navedene probleme i predložiti način njihovog korištenja u Android okolini. U praktičnom dijelu rada potrebno je za primjere zlonamjernih mobilnih aplikacija primijeniti predložene mehanizme zaštite, za različite sigurnosne prijetnje primijeniti odgovarajuće postupke šifriranja podataka, te analizirati sposobnost sigurnosnih alata i alata za testiranje za zaštitu od navedenih napada. Na temelju dobivenih rezultata treba analizirati sigurnost Android okoline.

2. SIGURNOST OPERACIJSKOG SUSTAVA ANDROID

Operacijski sustav Android koriste stotine milijuna korisnika diljem svijeta zbog čega je Android uspio pretećи Windows operacijski sustav na tržištu [1]. Ovaj operacijski sustav ne koristi se samo u mobilnim uređajima već i u tabletima, pametnim televizorima, satovima te igraćim konzolama i mnogim drugim uređajima koje čovjek može razviti. Iako je postao najzastupljeniji operacijski sustav na tržištu, velika većina korisnika ovog OS-a nije ni svjesno da su zapravo postali jedni od najugroženijih mobilnih korisnika i da njihovi podatci mogu završiti u krivim rukama.

2.1. Problemi sigurnosti na mobilnim uređajima

Činjenica da je Android sustav otvorenog koda i da svatko može doprinijeti razvoju i poboljšanju bilo kojih dijelova sustava postala je zapravo i loša strana jer upravo ona privlači napadače da iskoriste poznate nedostatke na brojnim uređajima te ponove metode napada kako bi ostvarili svoj cilj što brže i uspješnije. Poznato je da *Google* često ažurira svoj sustav kako bi što prije popravili moguće nedostatke te ohrabruje proizvođače uređaja, koji koriste njihov operacijski sustav, da tijekom razvoja uređaja koriste najnoviju inačicu sustava te da isti ažuriraju što češće, ali neki proizvođači to ignoriraju te tako riskiraju stvaranje problema za svoje korisnike. U sljedećim odlomcima bit će objašnjeni jedni od čestih problema sigurnosti mobilnih uređaja koji koriste Android [2], a nisu nužno povezani sa zlonamjernim aplikacijama već su dio operacijskog sustava.

Korisnik kao administrator. Iako neki ne vide problem u tome da korisnik ima potpunu kontrolu nad svojim uređajem, to se može pokazati opasnim ako korisnik ne zna što radi, dopusti aplikaciji pogled i korištenje nekih podataka koji su toj aplikaciji nepotrebni, povezivanje na nesigurnu mrežu i slično.

Tržište aplikacija. Ovdje nije nužno riječ o *Googleovoj* službenoj trgovini aplikacija već o cijelom tržištu aplikacija za Android sustava. U nekoliko slučajeva službeni proces ovjeravanja aplikacija koje se stavljaju na tržište pokazao se nedovoljno razvijenim i sigurnim jer su aplikacije I igre koje sadrže zlonamjeran kod bile lako dostupne korisnicima.

Rooting. Kako Android koristi Linuxovu jezgru, ovim procesom mnogi korisnici dobivaju dodatnu kontrolu nad svojim uređajem na razini podsustava kako bi zaobišli zapreke koje je proizvođač postavio što je slično superkorisniku u Linux operacijskom sustavu. U ovom procesu u uređaju se

gase sigurnosni mehanizmi što omogućuje da se zlonamjerna aplikacija instalira u neki od podsustava jer se tada mogu izbjeći detekcije kako bi se neometano pomoću šifriranja unijeli štetni kodovi što za rezultat može imati katastrofalne posljedice.

Privatnost. Neki proizvođači na svojim uređajima automatski postavljaju geolokaciju na fotografije ili objave na društvenim mrežama čime se u pitanje dovodi privatnost i sigurnosti korisnika koji možda nije svjestan da je otkrio i svoju lokaciju.

Dopuštenja. Kada korisnik odluči instalirati željenu aplikaciju, vrlo često je upitan dopušta li aplikaciji korištenje nekakvih podataka i funkcija uređaja koje su potrebne za normalan rad aplikacije. Ovo ne bi predstavljalo problem kada neke aplikacije ne bi tražile dopuštenja za stvari koje im nisu potrebne i već na prvi pogled bi trebale odvratiti korisnika od instalacije. Na primjer, nekakav običan kalkulator traži dopuštenje da koristi korisnikov imenik ili kameru.

2.2. Analiza trenutnog stanja sigurnosti na mobilnim uređajima i pregled postojećih rješenja

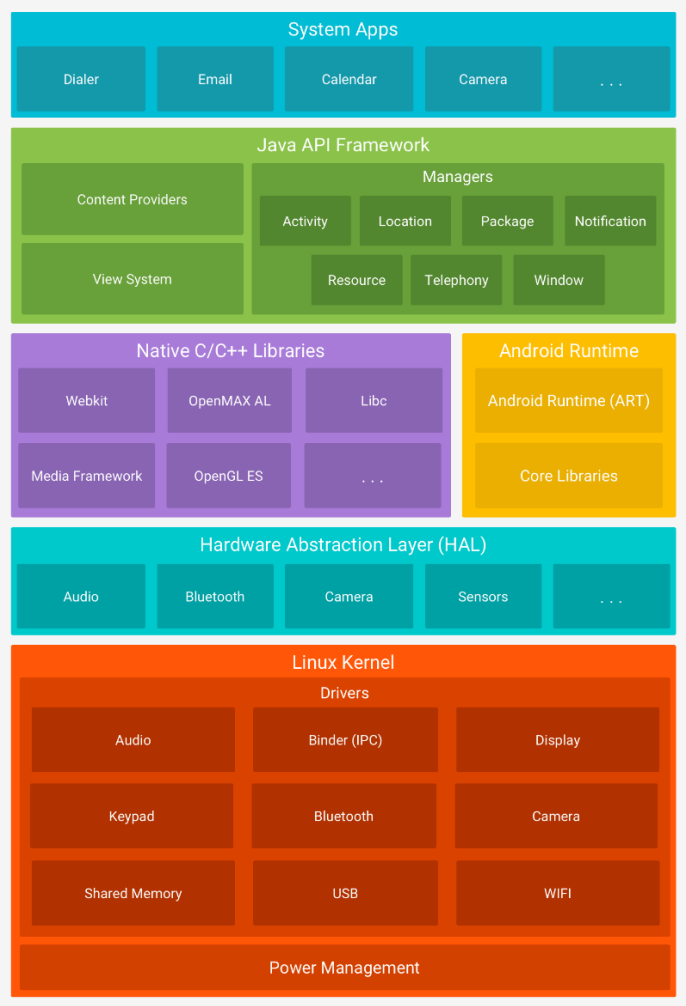
Iako se na prvi pogled može pomisliti kako nedostataka i prijetnji koji mogu naštetiti uređajima koji koriste operacijski sustav Android ima previše i da se ne mogu kontrolirati i spriječiti njihov utjecaj na sustav, programeri su pokazali da se jako dobro suprotstavljaju izazovima koji im se nameću kako bi korisnici bili što sigurniji. Kako je već spomenuto u prethodnom potpoglavlju, *Googleovi* programeri pokušavaju u što kraćem roku popraviti nedostatke i omogućiti noviju i sigurniju verziju sustava svojim korisnicima.

Kako je Android sustav temeljen na otvorenom kodu, svaki mjesec se zabilježi dvadesetak i više nedostataka i mjesta na kojima je sustav ranjiv. Većina njih nije previše opasna za korisnike, no znaju se otkriti i oni koji su kritični za korisnika. Zadnji poznati slučaj bio je u svibnju 2020.godine [3] kada je otkrivena greška *CVE-2020-0096* odnosno *StrandHogg 2.0* čija je prva inačica bila otkrivena u prosincu 2019. te u međuvremenu popravljena. *StrandHogg 2.0* zahvatio je uređaje koji rade na verzijama 8.0, 8.1, 9.0 te su samo korisnici Androida 10 zaštićeni. Ova ranjivost je opasna zato što se može iskoristiti i na uređajima na kojima nije izvršen rooting te ju je jako teško otkriti jer napadač ne ostavlja nikakve tragove iza sebe. Ako napadač uspije iskoristiti ovu ranjivost, tada može pristupiti bankovnim računima, kameri, porukama, fotografijama te podatcima za prijavu. *Google* je u

međuvremenu popravio ranjivost i otkrio da ranjivost, na sreću korisnika, nije iskorištena jer bi korisnik prvo morao instalirati zlonamjernu aplikaciju, svjesno ili nesvjesno, kako bi ta ranjivost došla do izražaja.

2.3. Sigurnost na razini operacijskog sustava

Android se sastoji iz nekoliko slojeva koji rade jedan iznad drugog. Ako se želi razumjeti okolina ovog sustava, potrebno je znati što svaki sloj predstavlja. Slojevi izvršavaju nekoliko funkcija koje omogućuju rad operacijskog sustava te svaki sloj pruža usluge sloju iznad sebe[4]. Na slici 2.1 preuzetoj iz [4], prikazan je izgled arhitekture sustava Android te njegovi slojevi.



Slika 2.1. *Arhitektura operacijskog sustava Android [4]*

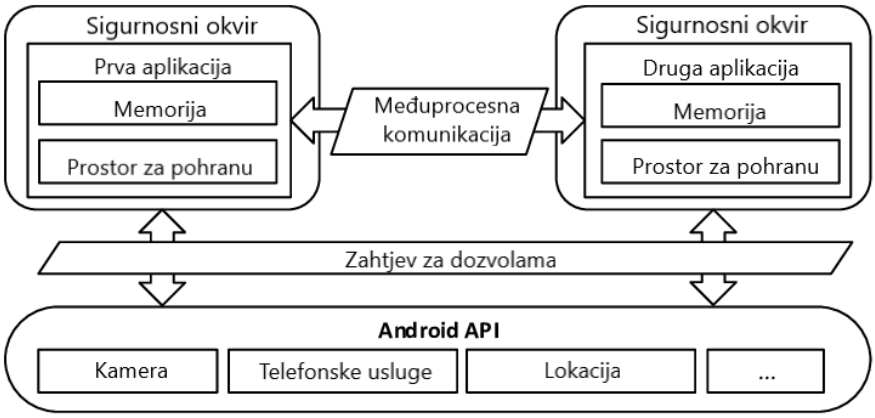
2.3.1. Jezgra Linuxa

Operacijski sustav Android temelji se na jezgri Linux operacijskog sustava, uz manje promjene kako bi se što bolje prilagodio potrebama Androida [4]. Linux se koristi u brojnim uređajima čija je sigurnosna okolina osjetljiva pa tako i u Androidu [5]. Razlog zbog kojega je Linux toliko korišten u uređajima raznih proizvođača te među stručnjacima iz grane kibernetičke sigurnosti je taj što je on provjeren sustav koji je dorađivan, popravljan i usavršavan tijekom godina. Android je korištenjem Linux jezgre sebi osigurao korisnički temeljen model za dopuštanje aplikacijama pristup potrebnim podatcima, izolirani rad procesa, prošireni mehanizam za sigurnosti IPC-a (eng. *Interprocess communication)* te mogućnost brisanja nepotrebnih te moguće nesigurnih dijelova jezgre. Svaka nova inačica Androida sadrži i novu inačicu Linux jezgre koja je izmijenjena prema potrebama odnosno poboljšana kako bi se popravili prethodni nedostatci i ranjivosti.

Unutar sigurnosnog modela Androida koristi se i Linux poboljšane sigurnosti[6] (eng. *Security-Enhanced Linux, SELinux)* koji radi prema principu podrazumijevanog odbijanja, odnosno ono što nije eksplicitno potvrđeno od strane korisnika, neće se ni izvršiti. Postoje dva modela *SELinuxa,* a to su način dopuštanja (eng. *Permissive mode)* koji bilježi odbijanje, ali ga ne provodi te izvršni način rada (eng. *Enforcing)* koji bilježi i provodi zabranu.

2.3.2. Sigurnosni okvir

Prema Linuxovom primjeru davanja jedinstvenog korisničkog ID-a (UID) svakom korisniku, tako Android svakoj aplikaciji daje UID te se svaka aplikacija izvršava kao zaseban proces. Sigurnosni okvir (eng. *Sandbox)* [7] onemogućava aplikacijama međusobnu komuniciranje kako bi se spriječilo moguće otjecanje podataka iz jedne u drugu ako je jedna od njih zlonamjerna. Rad dviju aplikacija koje koriste sigurnosni okvir prikazan je na slici 2.2 napravljenoj po uzoru na [8]. Sigurnosni okvir temeljen je na UNIX-ovom korisničkom razdvajanju procesa i dozvola za korištenje datoteka.



Slika 2.2. *Prikaz rada dviju aplikacija koje koriste sigurnosni okvir*

Ako bi se željelo izaći iz sigurnosnog okvira, tada bi se morala narušiti čitava sigurnosna okolina koju pruža Linux jezgra. Potrebna je dubinska obrana kako bi se sačuvao integritet sigurnosne okoline jer ni ona nije otporna na ranjivosti i nedostatke te se Android oslanja na brojne zaštite i sigurnosna ažuriranja kako bi korisnicima pružio potrebnu zaštitu.

2.4. Sigurnost na razini aplikacija

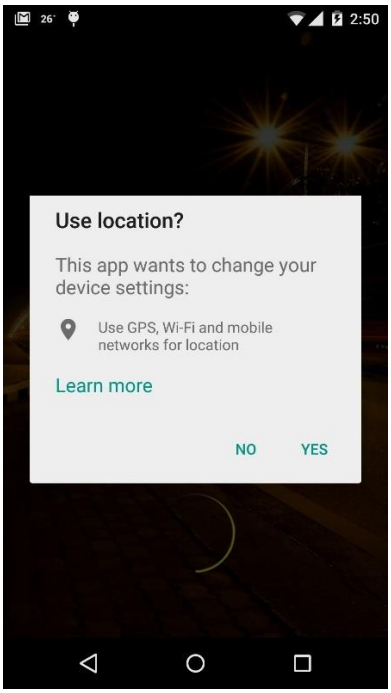
Android aplikacije [8] sastoje se od datoteke AndroidManifest.xml, aktivnosti (eng. *activities*), usluga (eng. *services*) i prijenosnika (eng. *broadcast receiver*). AndroidManifest.xml je kontrolna datoteka koja sugerira sustava što treba napraviti s komponentama poput aktivnosti, usluga, prijenosnika i svim dijelovima koji nude sadržaj. Aktivnost sadrži programski kod za samo jedan zadatak, a to je obično prikaz korisničkog sučelja te je jedna od aktivnosti ulazna točka svake aplikacije. Usluge su također programski kod koji se pokreće u pozadini aplikacija te može raditi kao nezavisan proces ili kao dio neke aplikacije. Usluge omogućuju da se ostale aplikacije povežu s njima te da pokrenu rad metoda preko udaljenih proceduralnih poziva. Prijenosnik je objekt koji se aktivira kada sam operacijski sustav ili aplikacija svojim radom stvore namjeru. Namjera je pasivna podatkovna struktura koja sadrži apstraktni opis događaja koji će se izvesti te predstavlja poveznicu među aktivnostima.

2.4.1. Model dozvola operacijskog sustava Android

Već je opisano kako svaka aplikacija radi unutar sigurnosnih okvira, odnosno *Sandboxa* zbog kojeg aplikacije mogu pristupiti ograničenom opsegu resursa unutar sustava. Ograničenja pristupa [7] su implementirana na različite načine jer su neki resursi ograničeni zbog namjernog izostavljanja

API-ja (*Application Programming Interface),* druge namjere su razdvojene prema ulogama što pruža jedinstvene sigurnosne mjere, a kod trećih se osjetljivo aplikacijsko sučelje koristi za rad povjerljivih

aplikacija te je zaštićen mehanizmom koji se zove dozvola. Neki od tako zaštićenih aplikacijskih sustava su funkcije kamere, *GPS, Bluetooth* i SMS/MMS funkcije te mrežne i podatkovne veze te se njima može pristupiti preko operacijskog sustava. Sve novije inačice Androida koriste model dozvola za vrijeme izvođenja što znači ako korisnik zahtjeva određenu funkcionalnost aplikacije koja zahtjeva korištenje nekog od zaštićenih aplikacijskih sučelja, sustava prikazuje dijalog kao što je prikazano slikom 2.3 gdje postavlja upit korisniku hoće li dopustiti ili odbiti takav zahtjev.



Slika 2.3. *Upit za korisnika želi li dopustiti korištenje zaštićenih API-ja*

Ako neki događaj unutar aplikacije pokuša pristupiti zaštićenom dijelu sustava za koji nije traženo dopuštenje tijekom instalacije same aplikacije doći će do pozivanja sigurnosne iznimke koji će aplikaciji reći da nema dopuštenje za izvršavanje takvog događaja. Model dozvola izuzetno je važan pri instalaciji aplikacija koje nisu preuzete sa službene trgovine Android aplikacija jer se u njima najčešće skrivaju zlonamjerni kodovi koji mogu dobiti pristup osjetljivim i osobnim podatcima korisnika. No nedostatak ovog modela je taj što korisnici automatizmom daju dozvole raznim aplikacijama te ni ne primijete da su nekoj aplikaciji dopustili korištenje nekih sučelja koji za tu aplikaciju nisu potrebni te se takvo doveli do nepotrebnog rizika otjecanja podataka. Na slici 2.4 (napravljena po uzoru na [8]), prikazano je kako aplikacije mogu dobiti pristup osobnim i osjetljivim podatcima jedino kroz sigurnosne provjere preko modela dozvola.

Slika 2.4. *Pristup korisničkim podatcima omogućen jedino nakon provjere dopuštenja*

2.4.2. Potpisivanje aplikacija

Kako bi razvijena aplikacija bila odobrena za postavljanje na *Google Play* uslugu, aplikacija mora biti potpisana s obzirom na to da je to prvi korak u implementiranju aplikacijskog sigurnosnog okvira. Ako aplikacija nije potpisana, neće se moći postaviti na službenu trgovini jer će biti odbijena zbog sigurnosnih rizika, a ako korisnik takvu aplikaciju želi instalirati od treće strane, Android sustav obavijestit će korisnika o riziku i predložiti da prekine s procesom instalacije. Obavezno je da aplikacije budu digitalno potpisane jer razvojni programer pomoću privatnog ključa koji je povezan s aplikacijom može ažurirati aplikaciju, a i potpisivanje aplikacije potvrđuje povjerenje između programera i *Google Play* trgovine da postavljena aplikacija neće naštetiti korisnicima, a ako dođe do takvog scenarija, ključ kojim je potpisana aplikacija će usmjeriti na programera koji će tada morati odgovarati zbog mogućih posljedica.

3. KRIPTOGRAFIJA I METODE ŠIFRIRANJA U OPERACIJSKOM SUSTAVU ANDROID

Android je postao najkorišteniji operacijski sustava, a time je privukao i moguće napadače koji će na ovaj ili onaj način doći do željenih podataka koje korisnik ostavlja za sobom korištenjem raznih aplikacija. Zato je šifriranje podataka izuzetno važno kako bi se očuvao integritet i sigurnosti podataka u aplikacijama.

3.1. Android *Keystore* sustav

Google se potrudio svojim korisnicima omogućiti što veću zaštitu [9] podataka te je implementirao *Keystore* sustav koji sprema kriptografske ključeve u poseban spremnik kako bi napadačima otežali pronalazak i uzimanje istih iz uređaja. Također nudi funkcionalnosti kako bi se ograničilo kako i kada se ključevi koriste. *Keystore* štiti ključeve od neovlaštenog korištenja tako da smanjuje neovlašteno korištenje ključa izvan Android uređaja sprječavanjem izvoženja ključeva iz aplikacijskog procesa i cijele sigurnosne okoline Androida. Nakon toga, aplikacija mora eksplicitno autorizirati korištenje svojih ključeva i prenijeti ograničenja izvan aplikacijskog procesa.

3.2. Metode šifriranja korištene u operacijskom sustavu Android

Izbor jedne od metoda šifriranja ovisi o tome što aplikacija ili projekt koji se razvija treba i koja metoda može pružiti najvišu razinu zaštite podataka. Neke metode nude bržu i lakšu implementaciju, ali nisu nužno efikasniji od metoda koje su teže za implementirati [10].

3.2.1. Asimetrično šifriranje

Ova metoda još je poznata kao i šifriranje javnog ključa te koristi par ključeva, gdje je jedan javni, a drugi privatni (slika 3.1 napravljena po uzoru na [10]). Javni je najčešće poznat svima koji koriste istu mrežu, a privatni je poznat samo drugoj strani, odnosno spremljen je na poslužitelju i ne smije se nikome otkriti. Ovaj način dopušta da se javni ključ dijeli kroz mrežu, a da se ne riskira otkrivanje šifriranih podataka jer bez privatnog ključa ne može se dešifrirati sadržaj. Nedostatak ove metode je što njena implementacije oduzima previše vremena te ako se izgubi ili zaboravi privatni ključ, nemoguće je dešifrirati podatke. Asimetrično šifriranje koristi se u *blockchain* transakcijama, a koristi ju i *Facebook* aplikacija za Android uređaje.

Slika 3.1. *Asimetrično šifriranje*

3.2.2. Simetrično šifriranje

Simetrično šifriranje koristi AES algoritam (*Advanced Encryption Standard*) koji predstavlja ključ za šifriranje i dešifriranje podataka i koristi, za razliku od asimetričnog šifriranja, samo jedan tajni ključ za šifriranje i dešifriranje podataka (slika 3.2 napravljena po uzoru na [10]). Iako je zbog ovoga brži i lakši za implementirati, ne pruža dovoljnu zaštitu podatcima jer ako se otkrije ključ šifriranja, također se razotkrio i ključ dešifriranja. Zato se ključ treba što bolje zaštititi i spremiti na takvo mjesto gdje ga se neće moći pronaći. Ovu metodu šifriranja koriste brojne aplikacije, a među njima su *WhatsApp* i *Firefox.*

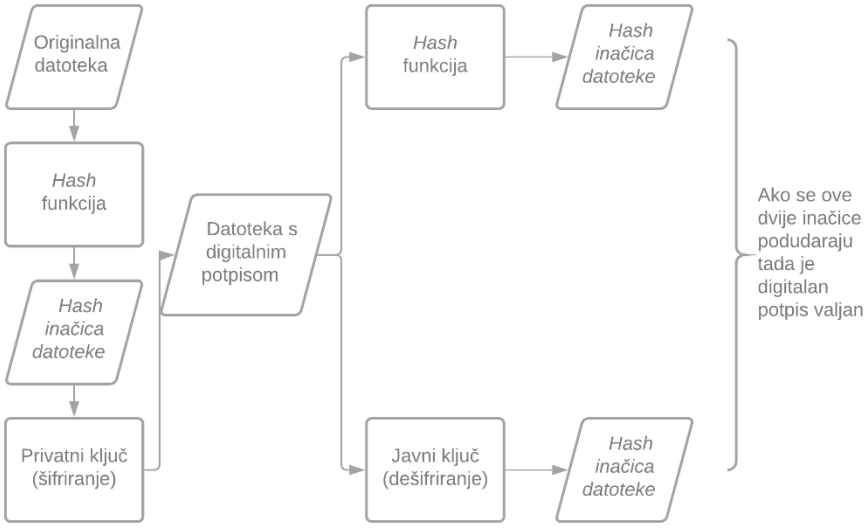
Slika 3.2. *Simetrično šifriranje*

3.2.3. *Hashing*

Metoda *hashing* koristi matematički algoritam kako bi se provjerio integritet podataka tako da uzme dio informacija i prikaže nasumični niz simbola [10]. Ovaj niz uvijek je iste duljine, a idealna *hash* funkcija stvara jedinstvene vrijednosti za svaki novi unos. U Androidu se ova metoda koristi kako bi se šifrirale lozinke s obzirom na to da je gotovo nemoguće otkriti lozinku koja je zaštićena *hashingom.* Nedostatak ove metode je što su se tijekom godina stvorili rječnici koji sadrže *hash* inačice nekih jednostavnih i često korištenih lozinki. Također, kako se koristi jedinstveni *hash* za određenu riječ, tako jednom otkrivena lozinka na jednom računu daje pristup napadaču svim drugim računima koji koriste tu lozinku.

3.2.4. Digitalni potpis

Algoritam digitalnog potpisivanja jedan je od najboljih algoritama šifriranja za Android aplikacije jer kombinira dvije gore spomenute metode: asimetrično šifriranje i *hashing*. Prvo poruka dobiva svoju *hash* inačicu, a zatim se provodi asimetrična metoda. Primatelj poruke koristi javni ključ dobiven asimetričnom metodom kako bi dobio *hash* inačicu potpisa, a zatim se ponovno koristi *hash* kako bi se usporedili primljeni i originalni *hash.* Ako se ova dva *hasha* podudaraju, primatelj može biti siguran da se poruka nije mijenjala tijekom prolaska kroz komunikacijski kanal. Nedostatak je taj što prva implementacija oduzima dosta vremena te ako korisnik promjeni privatni ključ neće se moći dešifrirati već prije dobiveni podatci. Proces je prikazan slikom 3.3 (napravljena po uzoru na [10]).



Slika 3.3. *Digitalni potpis*

3.2.5. *End-to-end* šifriranje

Prema ovom šifriranju pretpostavlja se da je komunikacija između dvije krajnje točke šifrirana i zaštićena. Radi po sličnom principu kao i asimetrično šifriranje, a glavno obilježje ove metode je da poslužitelj prenosi šifriranje poruke bez ikakve šanse da bi netko mogao narušiti integritet poruke mijenjanjem ili čitanjem iste. Manje poznata društvena mreža *Telegram* koristi upravu ovu metodu. Slanje kriptografskog ključa temelji se na Diffie-Hellmanovoj metodi [10] koja omogućuje slanje javnih ključeva kroz javnu i otvorenu mrežu bez da se kompromitiraju. U slučaju da se naruši sigurnosti jedne krajnje točke komunikacije, tada napadač može vidjeti i podatke druge točke.

3.2.6. Kriptografija eliptične krivulje

Javni ključevi koriste se u brojnim metodama pa tako i u ovoj. Ova metoda opisana u [10] koristi javne ključeve stvorene prema algebarskoj strukturi eliptičnih krivulja konačnih polja. To znači da koristi parametre koji će osigurati veću brzinu šifriranja. Nedostatak metode je što samo kriptografski jake eliptične krivulje mogu pružiti dovoljnu razinu sigurnosti.

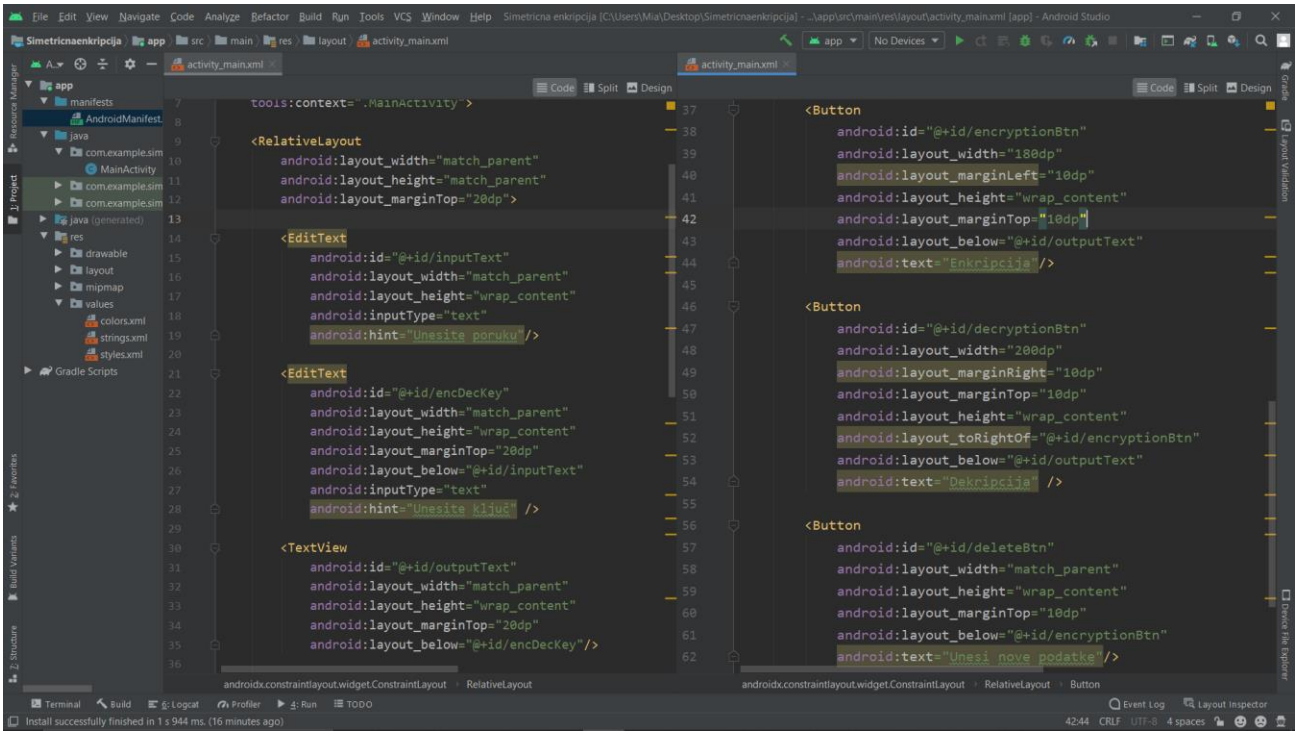
3.2.7. *HMAC*

Kod za autentifikaciju poruke temeljen na *hash* metodi kombinira *hashing* i privatni kriptografski ključ koji je poznat samo poslužitelj i krajnjem korisniku. Poslužitelj dešifrira poruku koristeći privatni ključ i uspoređuje podatke kriptirane *hashom* s originalnom porukom. Metoda se koristi i za šifriranje podataka i autentifikaciju korisnika, a aplikacije koje koriste ovu metodu za sigurnu komunikaciju su *Signal* i *WhatsApp*. Kako se temelji na *hashing* metodi, i kod ove metode napadači mogu iskoristiti rječnik *hash* lozinki kako bi uspješno došli do neovlaštenog pristupa računu.

3.3. Primjer kriptografske implementacije pomoću simetrične metode

Simetrična metoda šifriranja jedna je od lakših metoda za implementirati te će u ovom potpoglavlju biti prikazan i objašnjen jedan jednostavan primjer Android aplikacije koja šifrira i dešifrira kratku poruku koju korisnik unosi. Algoritam koji se koristi za šifriranje i dešifriranje je AES algoritam [11] koji ima tri vrste blok ključa, a to su 128-bitni, 192-bitni i 256-bitni (koji je korišten u primjeru) blok ključevi. Šifre nastale AES-256 algoritmom je teže za razotkriti napadima uzastopnim pokušavanjem nego što je to slučaj kod šifri nastalih AES-128 algoritmom. AES algoritam prvo mijenja podatke koristeći supstitucijsku tablicu [11], a zatim pomiče redove podataka i miješa stupce te na kraju svaki stupac transformira koristeći različite dijelove ključa za šifriranje. Što je ključ veći, to je potrebnije više ponavljanja kako bi se transformacija izvršila. Zbog jednostavnosti primjera, korisniku je omogućeno da sam izabere ključ za šifriranje i dešifriranje poruke, inače se ovakva praksa ne preporučuje.

U razvojnom okruženju Android Studio prvo je napravljeno jednostavno korisničko sučelje u datoteci *activity_main.xml* koje je prikazano na slici 3.4 te se sastoji od dva *EditText* elementa, *TextView* elementa te od tri gumba. U prvi *EditText* element korisnik unosi poruku koju želi šifrirati, a u drugi unosi ključ koji će se koristiti za šifriranje i dešifriranje. *TextView* korisniku pokazuje šifriranu, odnosno dešifriranu poruku. Korisničko sučelje prikazano je slici 3.5.

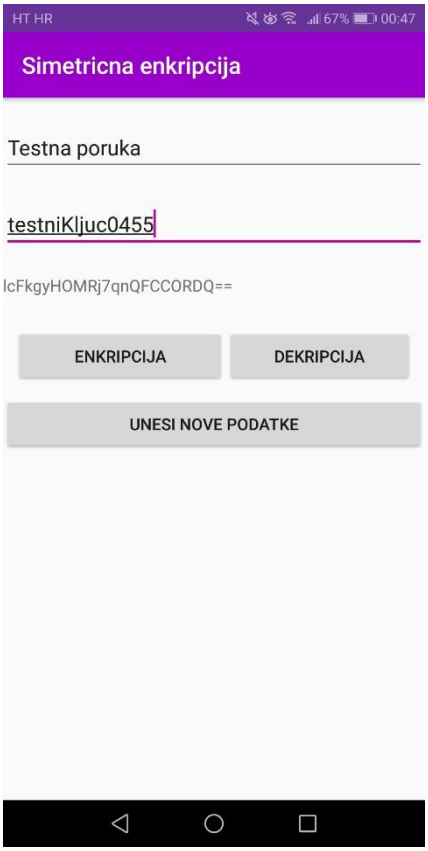


Slika 3.4. *Datoteka activity_main.xml za implementiranje korisničkog sučelja*

Slika 3.5. *Korisničko sučelje*

Korisnik unosi željenu poruku i ključ u odgovarajuće elemente te se pritiskom na gumb „Enkripcija“ generira tajni ključ prema ključu koji je unio korisnik koji pripada unesenoj poruci kao što je prikazano

slikom 3.6, a na slici 3.7 prikazan je programski kod koji prikazuje funkcije *encrypt* i *generateKey* koji omogućuju izvršavanje šifriranja poruke.



Slika 3.6. *Unos poruke i njezin šifrirani oblik*

```
private String encrypt(String data, String key) throws Exception{
    SecretKeySpec Key = generateKey(key);
    Cipher cipher= Cipher.getInstance(AES);
    cipher.init(Cipher.ENCRYPT_MODE,Key);
    byte[] encryptionValue=cipher.doFinal(data.getBytes());
    String encryptedValue= Base64.encodeToString(encryptionValue,Base64.DEFAULT);
    return encryptedValue;
}

private SecretKeySpec generateKey(String key) throws Exception{
    final MessageDigest digest= MessageDigest.getInstance("SHA-256");
    byte[] bytes = key.getBytes( charsetName: "UTF-8");
    digest.update(bytes,  offset: 0,bytes.length);
    byte[] Key= digest.digest();
    SecretKeySpec secretKeySpec= new SecretKeySpec(Key,  algorithm: "AES");
    return secretKeySpec;
}
```

Slika 3.7. *Funkcije za šifriranje poruke*

Klikom na gumb „Dekripcija“, u *TextView-u* prikazuje se originalna poruka (sl. 3.8) koju je korisnik unio na početku. Funkcija *Decrypt* prikazana je slikom 3.9. Želi li korisnik šifrirati novu poruku,

potrebno je pritisnuti gumb „Unesi nove podatke“ pomoću kojeg se brišu uneseni podatci iz *EditView* te *TextView* elemenata te korisničko sučelje izgleda ponovno kao na slici 3.5.

Slika 3.8. *Klikom na gumb „Dekripcija“ prikazuje se originalna poruka*

```
MainActivity.java
74  @    private String decrypt(String outputString, String key) throws Exception{
75           SecretKeySpec Key = generateKey(key);
76           Cipher cipher= Cipher.getInstance(AES);
77           cipher.init(Cipher.DECRYPT_MODE,Key);
78           byte[] decodedValue =Base64.decode(outputString,Base64.DEFAULT);
79           byte[] decodeValue =cipher.doFinal(decodedValue);
80           String decryptedValue= new String(decodeValue);
81           return decryptedValue;
82       }
83
```

Slika 3.9. *Funkcija za dešifriranje poruke*

Iz ovog primjera vidi se kako je simetrična metoda šifriranja jednostavna za implementirati, ali nije najsigurnija. Važno je za naglasiti da je u ovom primjer ključ koji se koristi za generiranje novog ključa za šifriranje i dešifriranje stalno vidljiv što nije sigurna praksa te je vidljiv samo iz razloga kako bi korisniku bilo što jasnije kako simetrična metoda funkcionira.

4. SIGURNOSNE PRIJETNJE MODELA DOZVOLA ANDROID SUSTAVA

Kao što je objašnjeno u prethodnim poglavljima, Android uređaji koriste sigurnosni okvir prema modelu dozvola koji ograničava aplikacijama pristup podatcima i zaštićenim funkcionalnostima koji aplikaciji nisu potrebni, bilo da su aplikacije preuzete sa službene trgovine ili treće strane. Jednom preuzeta i instalirana zlonamjerna aplikacija kojoj je, najčešće nesvjesno, dozvoljen pristup zaštićenim sučeljima i funkcionalnostima može uzrokovati otjecanje podataka, trošenje mobilnih podataka za pristup internetu te financijsku štetu korisniku bez njegovog znanja.

U dokumentu *AndroidManifest.xml* svake Android aplikacije nalaze sve dozvole koje aplikacija zahtjeva kako bi radila neometano, a u tom istom dokumentu se definiraju dozvole koje štite samu aplikaciju i njene komponente i resurse. Dozvole imaju četiri sigurnosne razine: normalnu, opasnu, potpis te potpis-ili-sustav (eng. *Signature-or-system*) razinu. Dozvole normalne sigurnosne razine su dozvole najmanjeg rizika što znači da aplikaciji, ako korisnik dopusti, daju pristup *API* pozivima bez da se korisniku nanese šteta. Opasne dozvole su dozvole visokog rizika čije prihvaćanje može imati štetne posljedice prilikom *API* poziva kao što je već spomenuto otjecanje podataka ili gubljenje kontrole nad uređajem. Dozvola potpisom omogućuje se ako je aplikacija koja ju zahtjeva potpisana istim certifikatom kao i aplikacija koja definira tu dozvolu, a dozvola potpis-ili-sustav se omogućuje samo ako aplikacija koja ju zahtjeva ima istu sustavsku sliku Androida ili se primjenjuje pravilo dozvole potpisa koja je već opisana.

4.1. Izravne prijetnje

Izravne prijetnje mogu, između ostalog, dovesti do otjecanja podataka ili financijskih gubitaka. U izravne prijetnje se podrazumijevaju[12]: prekomjerni zahtjevi za dozvolama, vrijeme provjere-vrijeme korištenja (eng. *time of check-time of use)* i napad eskalacijom dozvola.

Prekomjerni zahtjevi za dozvolama smatraju se jednim od najozbiljnijih prijetnji za operacijski sustav Android i njegove sigurnosne okvire jer izravno narušava načelo najmanje privilegije (eng. *the principle of least privilege*) te se time korisnikove informacije i financije dovode u rizik od iskorištavanja i gubitaka. Prema istraživanjima iz 2011.godine [12], 56% prekomjerno privilegiranih aplikacija je imalo samo jedan zahtjev za dozvolom više od potrebnoga, a 94% aplikacija je imalo četiri ili manje prekomjernih zahtjeva. Iako bi se na prvi pogled reklo da se zahtjeva previše dozvola, ovo je zapravo pokazatelj da razvojni programeri pokušavaju u *AndroidManifest.xml* dokumentu

definirati zahtjeve za dozvolama koji su stvarno potrebni za rad aplikacije nego da dodaju veći broj nepotrebnih zahtjeva. Do prekomjernog broja zahtjeva može doći i nenamjerno jer razvojni programeri nekad definiraju zahtjeve koji su naizgled potrebni za neku funkcionalnost koju su dizajnirali iako nisu potrebni ili pogreškom prilikom kopiranja koda koji nisu pregledali i uklonili neodobrene dozvole ili prilikom testiranja. Zbog toga se prekomjerni zahtjevi za dozvolama kategoriziraju u zlonamjerne i nesvjesne prijetnje.

Vrijeme provjere - vrijeme korištenja. Dozvole su u operacijskom sustavu Android predstavljene kao niz znakova i bilo koje dozvole istog imena su tretirane ekvivalentno iako pripadaju drugim aplikacijama koje međusobno nisu povezane ni na jedan način. Programeri koji razvijaju zlonamjerne aplikacije ovu manu iskorištavaju tako da, nakon što prvo navedu korisnika da instalira jednu takvu aplikaciju, zahtjeva dozvolu A koja ima isti naziv kao i dozvola B koja štiti pristup kritičnom resursu. Nakon toga, korisnik može instalirati najobičniju aplikaciju koja u sebi ima dozvolu B. Na ovaj način zlonamjerna aplikacija može pristupiti kritičnim resursima koji su inače zaštićeni dozvolom B.

Napad eskalacijom dozvola. Ovaj napad omogućuje zlonamjernoj aplikaciji da pristupi i čita željene informacije drugih aplikacija bez da izravno traži potvrdu dozvole. Napadi se dijele na napad zbunjenog pomoćnika (eng. *Confused deputy attack*) i napad tajne suradnje (eng. *Collusion attack)*. Napad zbunjenog pomoćnika iskorištava ranjivosti nezaštićenih sučelja privilegiranih dobroćudnih, odnosno korisniku normalnih, aplikacija. Na primjer, aplikacija A nema odobrenu dozvolu D te njena komponenta K_1 ne može pristupiti resursu R jer je zaštićen dozvolom D. Iako komponenta K_1 ne može direktno pristupiti resursu R, to ipak može učiniti preko komponente K_2 koja pripada aplikaciji B ako ta aplikacija ima odobrenu dozvolu D ili ne zahtijeva dozvolu uopće kako bi joj se pristupilo.

4.2. Neizravne prijetnje

Neizravne prijetnje mogu poslužiti kao pomoć za izazivanje napada na Android uređaje, a u te prijetnje se podrazumijevaju preciznost dozvola, nestručno upravljanje dozvolama i nedovoljno dokumentiranje dozvola.

Preciznost dozvola. Operacijski sustav Android definira preko 200 dozvola, a taj broj s pojavom svake nove inačice postaje veći. Većina dozvola se smatra preciznim dozvolama jer aplikaciji pružaju proizvoljan pristup određenim resursima, a među njima su *INTERNET, READ_PHONE_STATE* i *WRITE_SETTINGS* dozvole,odnosno dozvole za korištenje Interneta, čitanja stanja uređaja i

korištenje postavki. Dozvola za Internet omogućuje aplikaciji slanje HTTPS zahtjeva svim domenama te povezivanje na željene lokacije i portove, ali može doći do neučinkovitosti pri provođenju kontrole pristupa Internetu na aplikaciji. Internet dozvola je potrebna za rad velikom broju aplikacija, ali korisnik ne može kontrolirati ni ograničavati korištenje dozvole. Iz tog razloga, zlonamjerna aplikacija se može lako predstaviti kao legitimna koja stvarno treba pristup Internetu, ali će isti taj pristup zloupotrijebiti bez znanja korisnika.

Nestručno upravljanje dozvolama. Kako je već ranije spomenuto, Android sustava definira preko 200 dozvola, a u cijelom procesu stvaranja i odobravanja tih dozvola sudjeluju i razvojni programeri te krajnji korisnici. Programeri definiraju zahtjeve za dozvolama unutar manifest dokumenta aplikacije, a na korisniku je da prihvati ili odbije zahtjev za dozvolom prilikom korištenja uređaja. Za vrijeme definiranja zahtjeva za dozvolama, programeri nekad ne znaju pod kakvim rizikom će se korisnik naći kada i ako prihvati zahtjev zato što će jedan programer odvojiti vrijeme kako bi shvatio što svaka od 200 dozvola radi i kako ih koristili, a drugi će jednostavno prekomjerno zahtijevati dozvole kako bi njihova aplikacija radila što dovodi do vrste izravne prijetnje.

Nedovoljno dokumentiranje dozvola. Iako se na službenoj stranici operacijskog sustava Android može naći veliki broj dokumentacije kako bi razvojni programeri što lakše našli potrebne odgovore, u dokumentaciji koja opisuje dozvole nekad nije jasno koje metode klasa zahtijevaju dozvolu te je otkriveno šest pogrešaka u ovoj dokumentaciji. Ovakve pogreške mogu zbuniti programera koji će, umjesto da bude siguran gdje će i što definirati, jednostavno nagađati i pretpostaviti što treba napraviti i nesvjesno dovesti korisnika u sigurnosni rizik. Krajnji korisnici uređaja ne razumiju što sve dozvole znače te ih olako prihvaćaju i dovode se u sigurnosni rizik.

5. OTJECANJE PODATAKA

Otjecanje podataka se još može opisati i kao spora krađa podataka[13] te je ovo jedan od najvećih problema podatkovne sigurnosti na bilo kojem operacijskom sustavu. Ovi podatci uključuju prema [14], ime, korisničko ime, elektroničke adrese, broj uređaja, broj računa i ostale detalje uređaja koje napadač može iskoristiti. Do otjecanja podataka najčešće dolazi na dva načina [15]: preko *Google-ovih* usluga i pretraživanjem interneta.

Google-ove usluge predstavljaju okvir funkcionalnosti koji služi kao osnova za rad brojnim aplikacijama i njihovim značajkama. Primjerice, aplikacija želi pristupiti sučelju u kojem se nalaze podatci o korištenju karata te usluga *Google* karte (eng. *Google Maps*) pruža aplikaciji sve prikupljene podatke vezane uz korištenje karti koje aplikaciji trebaju. U pitanju može biti najlegitimnija aplikacija kao *Relive* koja će pratiti korisnikovo kretanje tijekom aktivnosti, a može biti i zlonamjerna aplikacija koja informacije šalje napadaču.

Internet tražilice su uglavnom najkorištenije aplikacije na Android uređajima te korisnici na njima, između ostalog, obavljaju Internet trgovinu i koriste Internet bankarstvo te pretraživači spremaju lozinke potrebne za korištenje ovih usluga. Tako svi podatci se nalaze na jednom mjestu i olakšavaju napadačima da ostvare svoj cilj. Upravo zato polovina podataka otječe iz Internet preglednika.

Google je 2014. godine preuzeo platformu *Firebase* koja se koristi za razvijanje aplikacija te pruža visoku razinu sigurnosti kako bi razvojni programer mogao što bolje zaštiti svoju aplikaciju i korisnike od brojnih prijetnji s kojima se operacijski sustav Android susreće. Na *Play Store-u*, službenoj trgovini Android aplikacija, nalazi se preko 150 000 aplikacija koje koriste *Firebase* [16], a od toga oko 12 000 aplikacija susreće se s problemom otjecanja podataka. Na sreću korisnika, samo je 4 200 aplikacija toliko slabo zaštićeno da omogući štetnije hakerske napade na bazu podataka i preuzmu sve željene informacije. No, loša vijest za korisnike je činjenica [16] da je prilikom otjecanja podataka iz tih 4 200 aplikacija otkriveno preko 7 milijuna elektroničkih adresa, preko 4 milijuna korisničkih imena, milijun lozinki i 5 milijuna telefonskih brojeva.

Kada korisnik prilikom korištenja aplikacije unese svoje podatke, može se dogoditi da aplikacija te podatke spremi na nesigurna mjesta u memoriji uređaja koja mogu postati dostupna zlonamjernim aplikacijama koje se nalaze na uređaju. Tada i kod postaje podložan napadima jer ga napadač može izmijeniti i pristupiti nesigurnim lokacijama na uređaju. Naravno nije svako otjecanje podataka

namjerno izazvano, postoje nenamjerne situacije otjecanja podataka[17], ali nažalost, bilo otjecanje namjerno ili nenamjerno, napadači uvijek nađu način kako to iskoristiti. Do nenamjernog otjecanja podataka može doći pri [17] kopiranju i korištenju međuspremnika (eng. *Copy/paste buffer caching*), zapisivanja u dnevnik (eng. *Logging*) te slanjem analitičkih podataka trećoj strani (eng. *Analytics data sent to third parties*). Kod kopiranja i korištenja stvari iz međuspremnika, kao što je broj kartice ili broj osobne, napadač ih lako može pročitati ako u izvorni kod uređaja doda jednostavni kod koji je prikazan slikom 5.1. Napadač može čitati informacije i s nekog drugog mjesta ako prikupljene podatke pošalje na udaljeni poslužitelj kojeg on kontrolira.

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

Slika 5.1. *Programski kod koji omogućuje napadaču čitanje iz međuspremnika*

Do slanja analitičkih podataka trećoj strani dolazi korištenjem *API-ja* treće strane unutar aplikacije i tako aplikacija čita bitne podatke poput broja uređaja i njegove lokacije. Jedno od „boljih“ mjesta na kojemu se može potražiti postoji li otjecanje podataka je Android dnevnik (eng. *Logs*). Razvojni programeri uglavnom koriste dnevnik za otklanjanje poteškoća dok razvijaju aplikaciju. Jedan od načina na koji napadači mogu čitati iz dnevnika je pomoću zlonamjerne aplikacije koja primjerice može sadržavati kod prikazan slikom 5.2. Iako je dozvola *READ_LOGS* onemogućena aplikacijama treće strane od *Jellybean* Android inačice (*Android 4.1. API level 16*), ovakva vrsta koda može se iskoristiti na svim uređajima na kojima korisnici imaju puni pristup operacijskom sustavu (eng. *rooting*)

```
Process process = Runtime.getRuntime().exec("su -c logcat -d");
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(process.getInputStream()));
StringBuilder log=new StringBuilder();
String line = "";
while ((line = bufferedReader.readLine()) != null)
{
log.append(line);
log.toString();
}
```

Slika 5.2. *Programski kod koji omogućuje čitanje iz dnevnika*

5.1. Problemi izazvani otjecanjem podataka

Istraživanje je pokazalo [18] da samo 2.3% podataka koji se izgube otjecanjem podataka čine brojevi kreditnih kartica korisnika, a čak 90% elektroničke adrese te 85.5% lozinke. Iz ovoga se vidi kako razina sigurnosti raste što je informacija osjetljivija, ali napadačima mogu biti sasvim dovoljni [15] elektronička adresa i lozinka kako bi ostvarili svoj cilj. Računi elektroničke pošte imaju listu kontakata

s kojima korisnik komunicira i razmjenjuje poruke što napadači mogu iskoristiti za širenje zlonamjernih aplikacija i programa putem elektroničke pošte. Zatim, mogu pronaći i koristiti vjerodajnice i certifikate za druge usluge i račune jer ako napadač dobije pristup jednom računu koji je povezan s ostalima, lako će doći i do podataka za ostale, a podatke o elektroničkoj adresi i lozinki mogu iskoristiti i za pokušaj ucjene korisnika. Podatci kućne adrese, broja osobne iskaznice, datuma rođenja, broja bankovnog računa i osobnog identifikacijskog broja mogu poslužiti napadaču da ukrade identitet i otvori nove račune na ime korisnika i stvori mu financijske gubitke. Otjecanje podataka također može pomoći napadačima da poboljšaju svoje zlonamjerne aplikacije jer pristupom u dnevnik sustava mogu vidjeti kako je sustava reagirao i obranio se od određenih napada i iskoristiti to znanje kako bi stvorili bolju aplikaciju.

5.2. Zlonamjerne aplikacije

Zlonamjerne aplikacije mogu izazvati razne probleme jednom kada dobiju pristup uređaju. One mogu izazvati [19] napade eskalacijom dozvola, već spominjano otjecanje podataka, prisluškivanje i praćenje razgovora i poruka, snimanje zvuka ili videa bez znanja korisnika te pregled bankovnog tokena ako ga korisnik dobije putem SMS-a. Također, takva aplikacija može izazvati napad odbijanja usluge (eng. *Denial of Service, DoS*) tako da se procesor, memorija i baterija koriste van svojih granica i mogućnosti te se korisniku onemogući korištenje uređaja. Zlonamjerne Android aplikacije i programi dijele se na [19] trojance, *botnet*, špijunske programe, stražnja vrata, agresivne oglasne aplikacije, crve, i ucjenjivačke aplikacije.

Trojanci (eng. *trojans*) koji se predstavljaju kao dobronamjerna, obična, aplikacija, zapravo u pozadini izvode štetne aktivnosti i tako uzrokuju otjecanje povjerljivih podataka, a mogu navesti korisnika da sam oda svoje podatke. Do 2012.godine Trojanci su se najviše slali putem SMS poruka te su korisnicima izazvali financijske gubitke. Također, podatke poput kontakata, poruka i drugih brojeva su slali nepoznatim domenama. Primjer zlonamjerne aplikacije je lažna aplikacija popularnog američkog pružatelja *streaming* usluga, *Netflixa*. Ova zlonamjerna aplikacija[20] navede korisnika da unese svoje korisničke podatke za prijavu, a korisnik nakon uspješne prijave umjesto usluge za gledanje filmova i serija dobije trojanca koji ima udaljeni pristup uređaju te može slušati razgovor korisnika tako da upali mikrofon, izvršavati razne naredbe, slati datoteke na udaljeni poslužitelj, snimati zaslon, vidjeti kontakte i slati SMS poruke.

Botnet aplikacije [19] stvaraju botove te uređaj može biti kontroliran naredbama s udaljenog poslužitelja koji se naziva *Bot-master*. Mreža ovakvih botova se naziva *Botnet*, a njihove naredbe se kreću od jednostavnijih poput izazivanja otjecanja podataka i njihovog slanja *Bot-masteru* do kompliciranijih i kompleksnijih kao što je izazivanja napada odbijanja usluge.

Špijunski programi (eng. *spyware*) tajno prate kontakte, poruke, lokaciju, bankovne kodove i slično kako bi ih zloupotrijebili. Također postoji velika vjerojatnost da će sve te podatke poslati udaljenom poslužitelju.

Stražnja vrata (eng. *backdoor*) dopuštaju ulaz u sustav tako da se obiđu svi sigurnosni protokoli i zaštite te se nakon toga mogu instalirati druge zlonamjerne aplikacije. Stražnja vrata koriste privilegije superkorisnika na *root* razini koje pomažu zlonamjernim aplikacijama da se prikriju od sigurnosnih aplikacija, a neke zlonamjerne aplikacije ih čak mogu i onemogućiti.

Agresivne oglasne aplikacije (eng. *aggressive adware*) koriste podatke o lokaciji koje Android prikuplja, a zatim sumnjive oglašivačke tvrtke podatke o lokaciji koriste kako bi stvorili prečac na početnom zaslonu, zabilješkama te mijenjaju zadane postavke pretraživača i aktiviraju obavijesti kojima će gušiti uređaj korisnika.

Crv (eng. *worm*) može stvoriti istu ili sličnu inačicu sebe i tako se proširiti mrežom ili nekom drugom vrstom prijenosa podataka kao što je primjerice *Bluetooth.*

Ucjenjivačke aplikacije (eng. *ransomware*) zaključavaju uređaj čineći ga nepristupačnim sve dok korisnik ne uplati zatraženi iznos novca napadaču koristeći mrežne naplatne usluge.

Prema istraživanju [21] skoro svaka zlonamjerna aplikacija promatra i sluša prijemnike unutar sustava kao što su *BOOT_COMPLETE, SEND_MESSAGE* te *SMS_RECEIVED, OUTGOING_CALL,* a kod običnih aplikacija nije uočeno promatranje ovih prijemnika. Na ovaj način se zlonamjerne aplikacije koriste kako bi pratile poruke te ih preusmjerile na druge brojeve koji mogu izazvati veći financijski trošak korisniku te uzrokovati već ranije spomenuto otjecanje podataka.

U svibnju 2020. godine, *Google* je [22] sa svoje službene trgovine uklonio 25 zlonamjernih aplikacija, prikazanih slikom 5.3 preuzete sa stranice *Hot For Security* [22], koje su uzrokovale otjecanje podataka koji su bili dio *Facebook* aplikacije. Svih 25 aplikacija pripadalo je istim razvojnim programerima tvrtke *Rio Reader LLC* i skinute su i instalirane više od 2 milijuna puta prije negoli li ih je *Google* otkrio i uklonio.

Application name	Package	Installs	Created date
Super Wallpapers Flashlight	com.wallpaper.flashlight.compass	500000	2019-07-23
Padenatef	com.sun.newjbq.beijing.ten	500000	2020-03-24
Wallpaper Level	com.liapp.level	100000	2019-04-22
Contour level wallpaper	com.communication.walllevel	100000	2019-04-28
iPlayer & iWallpaper	com.ldl.videoedit.iwallpapers	100000	2020-03-20
Video Maker	com.androidapp.videosedit.v	100000	2020-04-03
Color Wallpapers	com.play.ljj.wallpapercomapss	100000	2019-09-26
Pedometer	com.baidu.news.pedometer	100000	2020-01-18
Powerful Flashlight	com.meituan.ybw.flash	100000	2019-12-25
Super Bright Flashlight	com.tqyapp.sb.flashlight	100000	2019-01-18
Super Flashlight	com.superapp.xincheng	100000	2020-03-03
Solitaire Game	com.game.tqsolitaire	100000	2019-04-24
Accurate scanning of QR code	com.tqyapp.qr	50000	2019-02-20
Classic card game	com.card.solitairenew	50000	2019-05-09
Junk file cleaning	com.xdapp.cleaning	50000	2019-03-22
Synthetic Z	com.tqygame.synthetic	50000	2019-04-06
File Manager	com.smt.filemanager	50000	2017-12-27
Composite Z	com.game.hcz	50000	2019-04-22
Screenshot Capture	com.tianqiyang.lww.screenedit	10000	2019-07-03
Daily Horoscope Wallpapers	com.tianqiyang.lww.constellation	10000	2019-07-12
Wuxia Reader	com.wuxia.reader	10000	2017-05-14
Plus Weather	com.plus.android.weather	10000	2018-07-18
Anime Live Wallpaper	com.tqyapp.chuangtai	100	2019-01-10
iHealth Step Counter	com.tiantian.lang.tencent	N/A	2019-11-18
com.tqyapp.fiction	com.tqyapp.fiction	N/A	1970-01-01

Slika 5.3. *Zlonamjerne aplikacije uklonjene u svibnju 2020.godine [22]*

Ove zlonamjerne aplikacije su se predstavljale kao aplikacije za brojanje koraka, aplikacije za uređivanje fotografija i videa, mobilne igrice i sve su imale isti zlonamjerni kod te su tako dolazile do podataka bilo kojeg *Facebook* korisnika koji se prijavljivao u ovu aplikaciju, a prilikom toga imao instaliranu jednu od zlonamjernih aplikacija koja se prilikom pokretanja *Facebook-a* pokrenula u pozadini.

5.3. Moguća rješenja problema otjecanja podataka

Prilikom razvoja Android aplikacija dostupni su različiti alati i mehanizmi za zaštitu podataka, ali ne sjete se svi razvojni programeri zaštititi i manje očite elemente koji ulaze u proces rada s podatcima čije zanemarivanje može uzrokovati otjecanje podataka. Razvojni programeri mogu spriječiti otjecanje podataka [23] pravilnim i sigurnim spremanjem ključeva za šifriranje pomoću *Googleovog* alata *KeyStore* koji nude najsigurnije algoritme za spremanje i korištenje ključeva. Također,

programeri ne bi trebali ostavljati dnevnik (eng. *logs*) u inačici aplikacije koja se postavlja na službenu trgovinu aplikacija s obzirom na to da se u dnevniku mogu pronaći informacije koje bi napadačima dobro došle.

No, u većini slučajeva korisnici sami ugrožavaju svoju sigurnost, a zapravo se na vrlo jednostavne načine [24] mogu zaštiti od otjecanja podataka i zlonamjernih aplikacija. Korisnici ne bi trebali instalirati nepotrebne aplikacije, a pod njih se podrazumijevaju aplikacije koje predstavljaju daljinske upravljače koje, za razliku od daljinskog upravljača koji ste dobili uz pripadajući uređaj, one skupljaju korisnikove podatke o lokaciji, popis kontakata te ostalim podatcima kojima mogu pristupiti. Prilikom preuzimanja aplikacije korisnik bi trebao biti siguran da ju preuzima s legitimne stranice, a najsigurnije je aplikacije preuzeti sa službene *Google Play* trgovine. Iako se i na njoj mogu pronaći zlonamjerne aplikacije, kako je i opisano u ovom poglavlju, *Google* ih može ukloniti, a čim se uklone sa službene trgovine te aplikacije postaju nedostupne na korisnikovom uređaju ako ih ima instalirane te obavijesti korisnika o problemu preko svoje usluge *Play Protect*. Zatim, prije nego li instalira aplikaciju korisnik bi trebao jako dobro pogledati kakve zahtjeve za dozvolama ima aplikacija. U ovom poglavlju se opisivala lažna *Netflix* aplikacija koja je zapravo predstavljala trojanca te je čitala unesene podatke u aplikaciju, a iako je trojanac predstavljao vjerodostojnu inačicu poznate aplikacije, razlika je bila u nekoliko zahtjeva za dozvolama koje na očigled ne bi trebale biti dopuštene. Radi se o dozvoli za prikupljanju podataka drugih aplikacija koje su pokrenute u pozadini, dozvoli za prikupljanu podataka o identitetu odnosno svih računa koji se koriste na uređaju, dozvoli za čitanje liste kontakata te dozvolama za pokretanje aplikacije prilikom pokretanja uređaja i upravljanjem Wi-Fi prijema.

Preko 10 milijuna korisnika je instaliralo lažnu *Netflix* aplikaciju i ugrozilo svoje podatke zbog neznanja nečitanja zahtjeva za dozvolama. Na sreću korisnika, od Android inačice 6, može se ručno nakon instalacije upravljati dozvolama i naknadno omogućiti ili onemogućiti dozvole. S vremena na vrijeme korisnik bi trebao u upravitelju aplikacija pogledati koje se sve aplikacije nalaze na uređaju u slučaju da postoji aplikacija koju nije instalirao te bi trebao imati antivirusnu aplikaciju.

6. LOŠA KRIPTOGRAFSKA RJEŠENJA

Prema internetskoj zajednici [25], *The Open Web Application Security Project* (OWASP) loša kriptografska rješenja zauzela su peto mjesto na listi najčešće iskorištenih mana i slabosti u mobilnim aplikacijama. Loša i slaba kriptografska rješenja ili nesigurno korištenje kriptografije koje dovodi do nedostataka u šifriranju su dva najčešća problema kada se govori o slabostima kriptografije mobilnih aplikacija. Ako napadač otkrije ove slabosti, naći će način kako da sve šifrirane podatke ili kodove programa vrati u originalni tekst. Iskorištavanje loše kriptografije, osim što se dovodi do otjecanja podataka, također se korisniku, bilo privatnom ili poslovnom, krši privatnost, dolazi do krađe informacija i kodova, šteti ugledu korisnika te može doći do krađe intelektualnog vlasništva. Tvrtka koja se bavi kibernetičkom sigurnošću, *FireEye*, 2015. godine provela je analizu [26] najpopularnijih besplatnih aplikacija po broju preuzimanja na *Googleovoj* službenoj trgovini aplikacija, *Google Play* te su otkrili kako popriličan broj tih aplikacija nije šifriralo osjetljive podatke koristeći snažnije šifriranje. Zatim su analizirali sve aplikacije koje su prešle milijun preuzimanja do studenog 2014. godine te je otkriveno kako 5147 od 8261 analiziranih aplikacija (62%) ima barem jednu kriptografsku slabost. Operacijski sustav Android nudi razne algoritme i funkcije za šifriranje osjetljivih podataka, neki od njih su snažni i osiguravaju sigurnost podataka, dok neki nisu toliko sigurni i imaju svojih mana. Razvojni programeri trebali bi koristiti sve što im Android sustav nudi kako bi zaštitili svoje korisnike.

Šifriranje prema [26] otežava čitanje i korištenje podataka koje aplikacija pohranjuje na uređaju te dodaje posebni sigurnosni sloj podatcima koji se kreću između aplikacije i udaljenih poslužitelja, ali velik broj aplikacija ne provjerava certifikate poslužitelja što može dovesti do krađe podataka. Prema tvrtki *FireEye* [26], poželjna svojstva u šifriranju podataka su visoka razina entropije, algoritmi sa statusom te šifriranje na temelju lozinke. Algoritme za šifriranje je teže probiti ako je razina šifriranja veća, odnosno ako postoji veća nepredvidljivost prilikom nasumičnog generiranja brojeva koji će biti korišteni u procesu šifriranja. Visoka razina šifriranja se u Androidu postiže korištenjem *SecureRandom* klase. Algoritmi sa statusom koriste nasumično generirane vektore kako bi šifrirali podatke. Kako je inicijalni vektor nasumično odabran, ulazni tekst neće imati identičan rezultat šifriranja kao i izlazni, što je slučaj kada se koriste algoritmi bez statusa. Ovo znatno otežava napadaču stvaranje obrnutog rječnika kojim bi pokušao dešifrirati tekst. Šifriranje temeljeno na lozinkama je bitna funkcija kriptografije koja omogućuje korisniku zaštitu podataka unošenjem lozinke koja

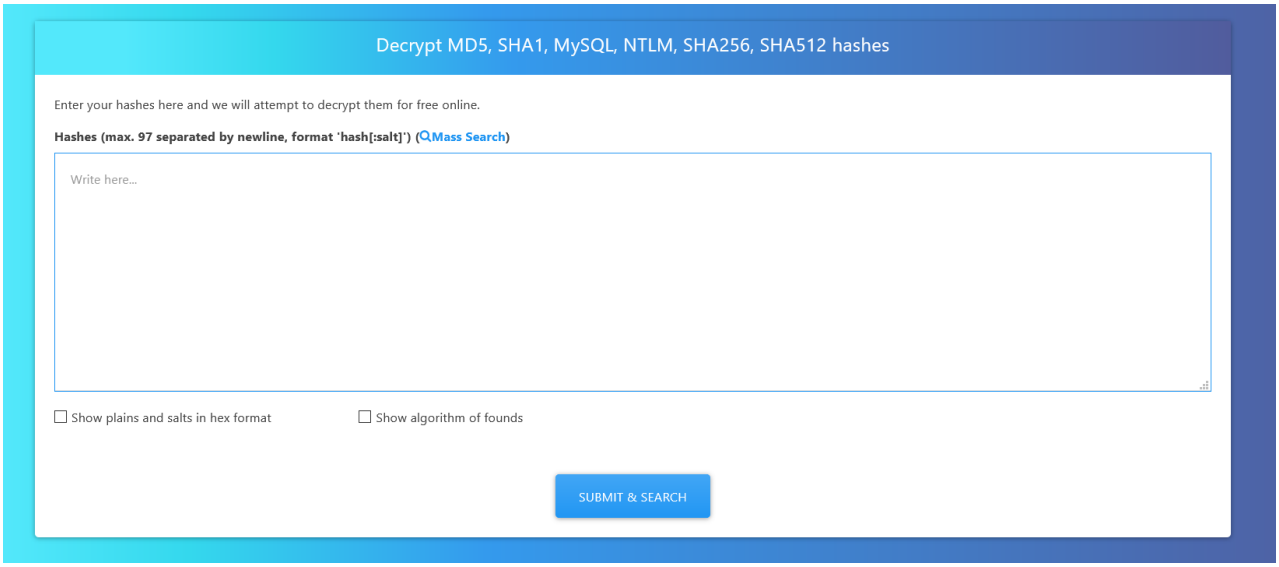
omogućuje da se podatci sigurno spreme i šalju preko interneta. Poteškoća ovog svojstva je što korisnici biraju lako pamtljive lozinke koje napadači mogu lagano probiti, a to se može riješiti generiranjem ključa iz dane lozinke te se taj ključ koristi za šifriranje podataka umjesto unesene lozinke.

Prema OWASP-u [27], postoje dva glavna razloga zbog kojih dolazi do sigurnosnih problema uzrokovanih lošim kriptografskim rješenjima, a to je mogućnost da se u aplikaciji implementiralo kriptografsko rješenje koje koristi već poznato slabi algoritam za šifriranje i dešifriranje te ga napadač vrlo lako može dešifrirati. Drugi razlog su nedostatci nastali prilikom postupka šifriranja iako je korišten snažan algoritam, ali je moguće da je proces šifriranja u nekom ili više koraka izveden krivo što napadačima otvara prostor za krađu podataka.

6.1. Korištenje slabih algoritama prilikom šifriranja

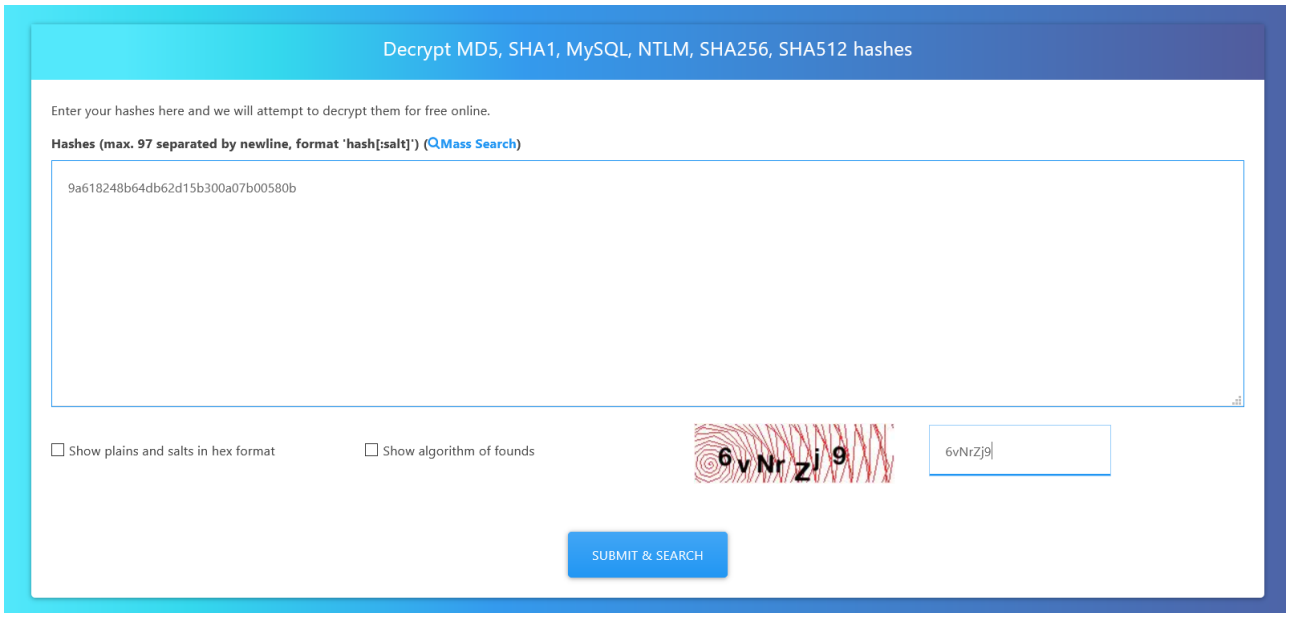
Neke aplikacije koriste nesigurne, slabe ili zastarjele algoritme [28] za koje je dokazano kako su podložni napadima zbog svojih slabosti, a to su *RC2*, *MD4*, *MD5* i *SHA1* algoritmi i protokoli. Ovi algoritmi ne zadovoljavaju potrebne sigurnosne zahtjeve te ih se zato ne bi trebalo implementirati u aplikacije. Ako aplikacija koristi ovako slabi algoritam, napadač bi mogao lako sebi omogućiti pristup podatcima ako dođe do ključeva stvorenih ovim algoritmima jer ih je lako dešifrirati.

Na jednostavnom primjeru [29] bit će prikazano korištenje ključa koji je nastao u procesu šifriranja korištenjem spomenutog algoritma *MD5*. U ovom primjeru se pretpostavlja da je aplikacija koristila *MD5* algoritam čiji je ključ sljedeći tekst: 9a618248b64db62d15b300a07b00580b. Na nesreću korisnika koji je instalirao ovakvu aplikaciju, ili neku drugu koja koristi jedan od navedenih slabih algoritama, na internetu je dostupno nekoliko platformi, alata i programa koje sadrže liste ključeva nastalih iz ovih algoritama koje su razni napadači sakupljali, dešifrirali i unosili u njihove baze podataka te se, uz malo sreće za napadača, lako može dešifrirati uneseni ključ. Slikom 6.1 prikazano je korisničko sučelje jednog od alata za dešifriranje.

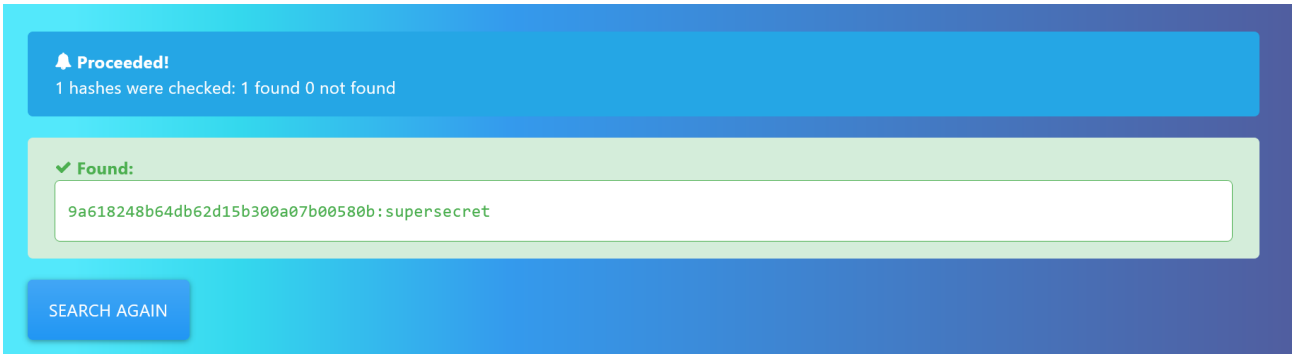


Slika 6.1. *Korisničko sučelje alata za dešifriranje ključeva [30]*

U prikazano korisničko sučelje unosi se gore spomenuti ključ te se pritiskom na gumb *Submit & Search* pokreće traženje dešifriranog teksta u bazi podataka (slika 6.2) te se dobiva rezultat *supersecret* kao što je prikazano slikom 6.3.



Slika 6.2. *Proces dešifriranje unesenog ključa [30]*



Slika 6.3. *Rezultat dešifriranja [30]*

Ovim primjerom je prikazano kako se lako može doći do originalne poruke ili podatka iz aplikacije koja se koristi algoritmom *MD5* s obzirom na to da postoje velike liste već dešifriranih ključeva dobivenih ovim i njemu sličnim slabim i zastarjelim algoritama. Korištenje slabih algoritama je riskantan potez prilikom razvoja aplikacije jer se time u pitanje nepotrebno dovodi sigurnosti korisnika te bi razvojni programer trebao pripaziti i pokušati implementirati što bolje i sigurnije rješenje, što nije problem ako se prati dokumentacija i savjeti za šifriranje koju Android nudi.

6.2. Nedostatci nastali u postupku šifriranja

Ako se snažan algoritam za šifriranje implementira [29] na nesiguran način, može doći do novih problema i nedostataka. Primjer je simetrični ključ *AES* [29], koji je, iako je snažan algoritam, podložan napadima u operacijskom sustavu Android ako ga se krivo implementira jer rukovanje ključevima tada može postati problem. Ni najbolji algoritmi [25] neće pružiti zaštitu korisniku ako su krivo implementirani i ako je rukovanje ključevima loše odrađeno, a pod time se podrazumijeva spremanje ključeva u isti direktorij u kojem se nalazi šifrirani sadržaj, korištenje ručno kodiranih (eng. *hardcoded*) ključeva unutar datoteka te omogućavanje napadaču jednostavni pristup ključevima. Ako je implementacija simetričnog šifriranja nužna, tada se preporučuje [29] korištenje šifriranja koje se temelji na lozinkama gdje se ključevi generiraju i dijele ovisno o unosu korisnika.

Prema OWASP-u [27] najlakši način do kojeg dolazi do lošeg rukovanja ključevima je kada se stvore i koriste vlastiti algoritmi i protokoli za šifriranje zato što osoba koja ih stvara ne mora pratiti svaku uputu koju je Android preporučio. Razvojnim programerima se savjetuje da koriste moderne i snažne algoritme koji su prihvaćeni od strane stručnjaka te budu oprezni pri implementaciji istih. Također treba izbjegavati spremanje osjetljivih i bitnih podataka na uređaj.

7. PRIKAZ STVARNOG PROBLEMA I ANALIZA RJEŠENJA ZA SPRJEČAVANJE OTJECANJA PODATAKA I LOŠEG ŠIFRIRANJA PODATAKA

U ovom poglavlju bit će prikazan stvaran problem otjecanja podataka i lošeg šifriranja podataka koji su se pojavili kod korisnika Android uređaja. Bit će prikazano najbolje rješenje za oba problema te analiza istoga. Slikom 7.1 prikazan je model pristupa prikazu problema i rješenja.

Slika 7.1. *Pristup za prikaz problema i rješenja*

7.1. Prikaz problema i analiza rješenja za sprječavanje otjecanja podataka uzrokovanog špijunskom aplikacijom

Kako je već ranije opisano u ovom radu, špijunski programi ili aplikacije su zlonamjerne aplikacije koje tajno prate rad operacijskog sustava Android te promatraju podatke kojima se korisnik koristi uz veliku vjerojatnost od slanja tih podataka udaljenom poslužitelju. U nastavku ovog rada bit će opisan *Tizi* špijunski program i način na koji ona radi, hoće li antivirusne aplikacije na uređaju primijetiti da je u pitanju zlonamjerna aplikacija te kako ga ukloniti s uređaja.

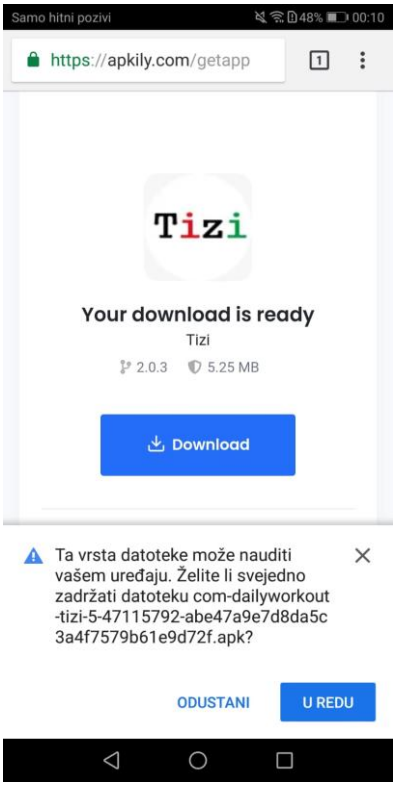
7.1.1. Prikaz i objašnjenje problema *Tizi* špijunske aplikacije

Tizi Android špijunska aplikacija [31] kreirana je kako bi pratila i koristila podatke prikupljene iz aplikacija poput društvenih mreža, snimala pozive, koristila kameru zaraženog uređaja, čitala i slala *SMS* poruke i *GPS* lokaciju udaljenom poslužitelju i na kraju dobila potpuni pristup Android uređaju i obavljala razne druge opasne radnje. Ovaj tip špijunske aplikacije *Google* [32] je otkrio u rujnu 2017., a daljnje analize su pokazale da je dostupna još od 2015.godine te je preko 1300 Android

uređaja bilo pod napadom *Tizi* virusa, odnosno zlonamjerne aplikacije. Najviše uređaja pogođeno je na teritoriju Afrike, a zatim Azije, Europe i Sjeverne Amerike.

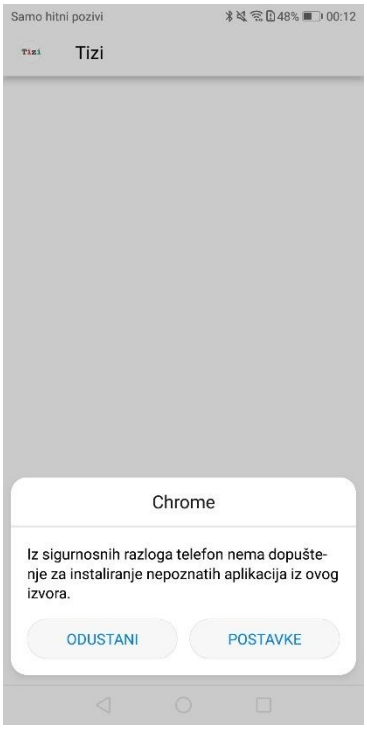
Tizi osim što je špijunska aplikacija, ima i elemente stražnjih vrata koji omogućuju ovoj aplikaciji da dobije potpuni pristup aplikaciji na razini operacijskog sustava te tako instalira špijunske aplikacije. Kreiran je kako bi sakupljao podatke dobivene od *Facebooka, Twittera, WhatsAppa, Skypea, Telegrama, LinkedIna i Vibera.* Nakon što korisnik instalira aplikaciju, ona dobiva potpuni pristup uređaju i krade informacija iz navedenih aplikacija te izvodi mnoge druge opasne napade, odnosno, dolazi do otjecanja podataka. Zlonamjerna aplikacija za vrijeme svog napada iskorištava prisutne manjkavosti uređaja koji imaju starije i neažurirane inačice operacijskog sustava Android, ali to ne znači da su korisnici novijih inačica operacijskog sustava Android sigurni. U novijim inačicama aplikacija mora tražiti zahtjev za dozvolom pristupa razini operacijskog sustava, ali dosta korisnika ne čita što zahtjevi traže te ih prihvate neznajući da su zlonamjernoj aplikaciji omogućili napad.

Inačice ove aplikacije bile su dostupne i na službenoj *Googleovoj* trgovini aplikacija, a neke su bile dostupne od treće strane. Trenutno su poznata [31] tri paketa koja su imala *Tizi* virus u sebi, a to su: com.press.nasa.com.tanofresh, com.dailyworkout.tizi i com.system.update.systemupdate paketi. Prema [33], pretpostavlja se da je *Tizi* aplikacija za vježbanje dobila svoj naziv prema istoimenom sportskom brendu u Keniji. Aplikacija više nije dostupna na *Google* trgovini, ali prilikom pokušaja preuzimanja od treće strane, korisniku se pojavljuje obavijest kao što je prikazano slikom 7.2. *Napomena: primjer je prikazan na uređaju s novijom inačicom operacijskog sustava Android.*



Slika 7.2. *Pokušaj preuzimanja zlonamjerne aplikacije Tizi*

Ako korisnik ignorira upozorenje i odluči zadržati datoteku pritiskom gumba *U redu*, pojavljuje se upozorenje (slika 7.3) koje obavještava korisnika da je zbog sigurnosnih razloga instalacija aplikacije prekinuta, a korisnik ju može nastaviti ako u postavkama uređaja omogući instalaciju aplikacija preuzete od treće strane. Nakon što je korisnik omogućio instalaciju, aplikacija je pokušala započeti instalaciju, ali sam sustav je prepoznao kako je riječ o zlonamjernoj aplikaciji i blokirao pokušaj instalacije (slika 7.4)

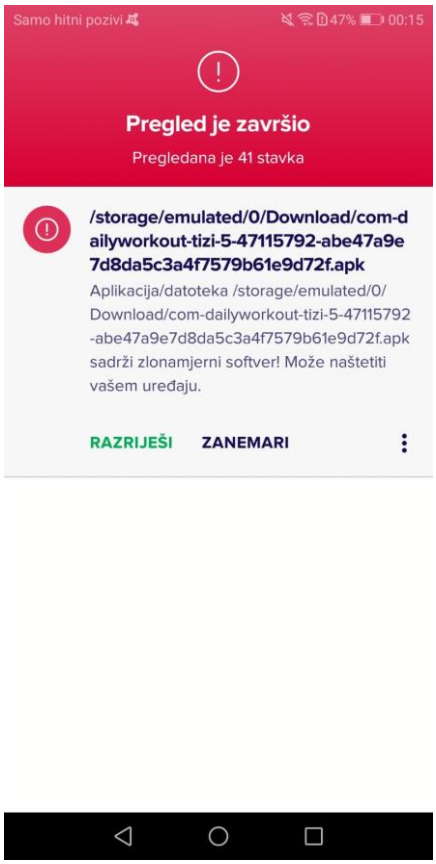


Slika 7.3. *Sigurnosno upozorenje*

Slika 7.4. *Obavijest o blokiranju instalacije*

7.1.2. Prijedlog i analiza rješenja

U prikazanom primjeru, sigurnosni Android je pravodobno reagirao te je prekinuo instalaciju aplikacije, ali kod uređaja koji koriste starije inačice operacijskog sustava, moguće je da virus zaobišao sigurnosni sustava te se uspješno instalirao. Ako korisnik primijeti da mu se ekran ledi, sporije radi ili mu se pojavljuje prevelik broj reklama, potrebno je pokrenuti skeniranje uređaja pomoću jednog od antivirusni alata, poput *Avast Mobile Security* aplikacije koja je i korištena u ovom primjeru. Kada se završi skeniranje uređaja, najčešće će antivirusna aplikacija pronaći datoteku sa zlonamjernom aplikacijom i upozoriti korisnika te mu ponuditi uklanjanje zlonamjerne aplikacije kao što je prikazano slikom 7.5.



Slika 7.5. *Prikaz obavijesti o pronalasku zlonamjerne aplikacije*

Postoje slučajevi kada je zlonamjerna aplikacija već uspjela postići svoj cilj te je osim izazivanja otjecanja podataka također promijenila brojne postavke i onemogućila rad bilo kakvih sigurnosnih sustava i aplikacija unutar uređaja. U tom slučaju, najbolje što korisnik može napraviti je vratiti uređaj na tvorničke postavke jer mu nijedan drugi način ne garantira potpuno uklanjanje aplikacije s uređaja,

a time mu ne nudi ni potpunu sigurnosti. Kao što je već navedeno u ovom radu, korisnik će se zasigurno zaštititi ako prilikom preuzimanja aplikacija pročita sve zahtjeve za dozvolama, preuzima aplikacije isključivo s *Google* službene trgovine jer veća je šansa [32] da će korisnik izgubiti uređaj nego što će preuzeti zlonamjernu aplikaciju preko službene trgovine. Korisniku se također preporučuje da redovno ažurira svoj uređaj kako bi se preuzele sigurnosne zakrpe te ažurirala lista prijetnji koja omogućava uređaju da otkrije zlonamjerne aplikacije.

7.2. Prikaz problema i analiza rješenja za sprječavanje lošeg šifriranja podataka

Zbog kompleksnosti izvedbe i implementacije kriptografskih rješenja, sljedeći primjer bit će opisan samo teoretski. Prikazat će se problem slabog šifriranja unutar aplikacije za praćenje zdravlja i aktivnosti korisnika, *HealthSuite*.

7.2.1. Prikaz i objašnjenje problema lošeg šifriranja podataka unutar aplikacije

U prosincu 2018. godine, Ministarstvo domovinske sigurnosti SAD-a (eng. *Department of Homeland Security, DHS*) objavilo je [34] kako je aplikacija tvrtke *Philips, HealthSuite*, podložna napadima hakera zbog otkrivenih slabosti unutar šifriranja podataka, ponajviše zbog nedostatka snažnog algoritma. Ovaj problem zahvatio je sve postojeće inačice aplikacije dostupnih za operacijski sustav Android. Oznaka slabosti je *CVE-2018-19001* te je poznato [35] da pripada *CWE – 326* skupini slabosti, a svojstvo ove skupine je spremanje i prenošenje osjetljivih podataka koristeći postupak šifriranja koji nije dovoljno snažan za zaštitu podataka koji se koriste u aplikaciji. Ukupan broj slabosti u ovoj skupini je 111, a otkrivena slabost unutar ove aplikacije nalazi se na 67. mjestu [36] s ocjenom ranjivosti 4.6 iz čega se može zaključiti da ova slabost nije toliko opasna u usporedbi s ostalim slabostima iz ove skupine. Prema [37], slabost šifriranja *CVE-2018-19001* ima djelomičan utjecaj na povjerljivost, dostupnost i integritet aplikacije.

Aplikacija omogućuje korisniku unos i praćenje zdravstvenih navika za čije prikupljanje podataka koriste mnoge druge *Phillipsove* uređaje koji se mogu upariti s aplikacijom. Ti uređaji mjere otkucaje srca, težinu, visinu, analizu tjelesne aktivnosti (koja može uključivati i lokaciju korisnika), tlak te kvalitetu sna. Ako napadač uspije probiti šifriranje, može promijeniti podatke unutar aplikacije i narušiti integritet aplikacije i privatnost korisnika, zbog toga američka vlada smatra da razvojni programeri ove aplikacije nisu koristili dovoljno jako šifriranje s obzirom s kakvim se sve podacima koristi aplikacija. Tvrtka Philips uvjerava korisnike [34] da do trenutka pronalaska manjkavosti unutar

šifriranja nije bilo slučaja iskorištavanja ovog problem, a iz tvrtke su najavili kako će problem riješiti tijekom 2019. godine. Uz to su zamolili korisnike da ne *rootaju* svoje uređaje jer tada je rizik od pronalaska šifriranih i dešifriranih podataka još veći te se napadaču povećavaju šanse za ostvarivanjem cilja, a to je krađa podataka.

Prema Benu Ransfordu [34], predsjedniku zdravstva kibernetičke tvrtke *Virta Labs*, šifriranje je znanost u kojoj i najveći stručnjaci svakodnevno griješe. Istaknuo je kako je *Philipsova* aplikacija tada koristila nekoliko slabih sustava šifriranja te da se pri dizajniranju i razvoju aplikacije trebalo dobro promisliti i pretpostaviti koliko će sustava biti slab i podložan napadima te tako ograničiti mogućnost eventualnih napada.

7.2.2. Prijedlog i analiza rješenja

Nije poznato na koji način je tvrtka *Philips* tijekom 2019. godine popravila problem, ali je Ben Ransford [34] preporučio korištenje različitih ključeva za šifriranje za svaku kopiju aplikacije ili svakog korisnika kako u slučaju napada na jednog korisnika i krađe njegovih podataka ne bi došlo do otkrivanja podataka ostalih korisnika. Kako je već spomenuto u radu, na razvojnim programerima je da koriste što modernije, snažnije i bolje funkcije šifriranja, algoritme i ključeve kako bi zaštitili svoje korisnike od mogućih napada, jer uz svaki napad, osim što se krši privatnost korisnika, narušava se i ugled tvrtke ili pojedinačne osobe koja je bila zadužena za šifriranje podataka. Ako razvojni programer nije sigurnosni stručnjak i ne zna kako točno implementirati šifriranje, ne bi trebao pokušavati osmisliti svoje algoritme za šifriranje jer se tada korisnike stavlja u još veći rizik, a za razliku od problema otjecanja podataka gdje korisnik može utjecati na svoju sigurnost, ovdje on nema utjecaja i velika većina problema i rješenja se tiče direktno razvojnih programera i ostalih stručnjaka.

8. ZAKLJUČAK

U ovom radu proučena je sigurnosna okolina operacijskog sustava Android, odnosno opisana sigurnosna okolina na razini operacijskog sustava. Ona uključuje jezgru Linux operacijskog sustava i sigurnosni okvir, te na razini samih aplikacija gdje je opisan model dozvola operacijskog sustava Android te potpisivanje aplikacija. Zatim je opisan rad sustava *Keystore* koji se u Androidu koristi za što lakšu i bolju implementaciju kriptografskih rješenja. Opisane su sve metode šifriranja koje se mogu koristiti u operacijskom sustavu Android zajedno s njihovim prednostima i nedostacima. Zatim su opisane najčešće sigurnosne prijetnje s njihovim vrstama. Objašnjen je problem otjecanja podataka koje može izazvati čak i službena *Googleova* usluga, a također su navedeni problemi do kojih može doći otjecanjem podataka kao što je krađa identiteta, poveći financijski gubitci te općenito kršenje privatnosti. Nadalje, opisane su zlonamjerne aplikacije i njezine vrste od trojanca pa do ucjenjivačkih aplikacija.

Na primjeru zlonamjerne aplikacije prikazano je da operacijski sustav Android i antivirusni alati vrlo dobro reagiraju i brzo otkrivaju postojanje zlonamjerne aplikacije na uređaju. Operacijski sustav Android je otkrio zlonamjernu aplikaciju, upozorio korisnika s nekoliko obavijesti te na kraju blokirao instalaciju aplikacije na uređaj, a antivirusni alat je pronašao datoteku u kojoj se nalazi zlonamjerna aplikacija te ju uklonio s uređaja. Može se zaključiti kako je svaka vrsta zlonamjerne aplikacije sama po sebi opasna te korisnik mora biti vrlo pažljiv kada instalira aplikaciju, a najvažnije je da čita što dopušta aplikaciji, da provjeri imali nepotrebnih zahtjeva za dozvolama te da provjerava popis instaliranih aplikacija kako bi bio siguran da se bez njegovog znanja instalirala zlonamjerna aplikacija. Ono što *Google* može učiniti je što prije otkriti i ukloniti ovakve aplikacije i njihove autore kako bi korisnici ne bi ni imali pristup takvim aplikacijama. Korisnik bi također trebao biti siguran da preuzima aplikacije sa službene trgovine, a ne od treće strane. Na kraju, opisan je problem do kojeg može doći lošim kriptiranjem i spremanjem ključeva šifriranja unutar aplikacije, a to je otjecanje podataka koje je aplikacija prikupila, a nije ih zaštitila dovoljno jakim postupkom šifriranja. Ovaj problem je složeniji i zahtjeva više vještine kako bi se sigurno i pravilo implementiralo šifriranje na uređaju te ovdje korisnik ne može učiniti puno kako bi se zaštitio, osim da se informira o mogućim problemima šifriranja aplikacije. Sigurnosna okolina operacijskog sustava Android načelno dobro djeluje i radi, stalno se nadograđuje i prilikom ažuriranja se instaliraju sigurnosne zakrpe, ali napadači uvijek nađu drugi način kako ostvariti svoj cilj, a to je naštetiti korisniku.

LITERATURA

[1] Smith, CSO, Android Now the World's Most Popular Operating System, CSO, 2017.godina, dostupno na: https://www.csoonline.com/article/3187011/android-is-now-the-worlds-most-popular-operating-system.html, pristupljeno 05.07.2020.

[2] A. Shibly, Android Operating System: Architecture, Security Challenges and Solutions, South Eastern University of Sri Lanka, 2016., dostupno na: https://www.researchgate.net/publication/299394606_Android_Operating_System_Architecture_Security_Challenges_and_Solutions, pristupljeno 05.07.2020.

[3] D. Winder, This Critical Android Security Threat Could Affect More Than 1 Billion Devices: What You Need To Know, svibanj 2020., dostupno na: https://www.forbes.com/sites/daveywinder/2020/05/26/critical-android-data-stealing-security-threat-confirmed-for-almost-all-android-versions-strandhogg-google-update-warning/#1d37da202c82, pristupljeno 06.07.2020.

[4] S. Lahoti, What Role Does Linux Play in Securing Android Devices?, listopad 2018., dostupno na: https://hub.packtpub.com/what-role-does-linux-play-in-securing-android-devices/, pristupljeno 07.07.2020.

[5] Android dokumentacija, System and Kernel Security, dostupno na: https://source.android.com/security/overview/kernel-security, pristupljeno 07.07.2020.

[6] Android dokumentacija, Security-Enhanced Linux in Android, dostupno na: https://source.android.com/security/selinux, pristupljeno 07.07.2020.

[7] Android dokumentacija, Application Sandbox, dostupno na: https://source.android.com/security/app-sandbox, pristupljeno 08.07.2020.

[8] Android dokumentacija, Application Security, dostupno na: https://source.android.com/security/overview/app-security, pristupljeno 08.07.2020.

[9] Android dokumentacija, Android keystore system, dostupno na: https://developer.android.com/training/articles/keystore, pristupljeno 09.07.2020.

[10] T. Kriuchkov, Top 7 Methods of Data Encryption in Android Applications, travanj 2019., dostupno na: https://www.apriorit.com/dev-blog/612-mobile-cybersecurity-encryption-in-android, pristupljeno 10.07.2020.

[11] M. Rouse, Advanced Encryption Standard (AES), studeni 2014., dostupno na: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard, pristupljeno 10.07.2020.

[12] Z. Fang, W. Han, Y. Li, Permission Based Android Security: Issues and Countermeasures, Computers & Security, Vol. 43, lipanj 2014., dostupno na: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=3531&context=sis_research, pristupljeno 17.08.2020.

[13] Forcepoint, What is Data Leakage?, travanj 2020.,dostupno na: https://www.forcepoint.com/cyber-edu/data-leakage, pristupljeno 19.08.2020.

[14] A. Freier, Business of Apps, Almost All Retail Android Apps Leak Personal User Data, siječanj 2020., dostupno na: https://www.businessofapps.com/news/almost-all-retail-android-apps-leak-personal-user-data/, pristupljeno 19.08.2020.

[15] S. Topuzov, Secure Group, How Data Leaks From Your Smartphone – and Can You Stop it?, veljača 2017., dostupno na: https://blog.securegroup.com/how-does-data-leak-from-your-smartphone-and-how-do-you-stop-it, pristupljeno 19.08.2020.

[16] Z. Muhammad, Digital Information World, 4200+ Android Apps Responsible for Massive Data Leak, svibanj 2020., dostupno na: https://www.digitalinformationworld.com/2020/05/4200-android-apps-responsible-for-massive-data-leak.html, pristupljeno 20.08.2020.

[17] Srinivas, Infosec, Exploiting Unintended Data Leakage (Side Channel Data Leakage), veljača 2015., dostupno na: https://resources.infosecinstitute.com/android-hacking-security-part-4-exploiting-unintended-data-leakage-side-channel-data-leakage/, pristupljeno 21.08.2020.

[18] E. Tuvey, IT Pro Portal, Mobile Data Leaks – the Hidden Dangers to Organisations, siječanj 2017., dostupno na: https://www.itproportal.com/features/mobile-data-leaks-the-hidden-dangers-to-organisations/, pristupljeno 22.08.2020.

[19] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M.S. Gaur, M. Conti, Android Security: A Survey of Issues, Malware Penetration and Defenses, veljača 2014., dostupno na: https://openaccess.city.ac.uk/id/eprint/12200/1/ , pristupljeno 23.08.2020.

[20] I. Arghire, SecurityWeek, Fake Netflix App Takes Control of Android Devices, siječanj 2017., dostupno na: https://www.securityweek.com/fake-netflix-app-takes-control-android-devices, pristupljeno 25.08.2020.

[21] M. Alazab, V. Moonsamy, L. Batten, R. Tian, School of Information Technology, Deakin University, Australia, Analysis of Malicious and Benign Android Applications, 2012., dostupno na: https://www.veelasha.org/pubs/c2012-1.pdf, pristupljeno 27.08.2020.

[22] A. Bizga, Hot For Security, Google Removes 25 Malicious Google Play Apps Stealing Facebook Login Credentials, srpanj 2020., dostupno na: https://hotforsecurity.bitdefender.com/blog/google-removes-25-malicious-google-play-apps-stealing-facebook-login-credentials-23629.html, pristupljeno 28.08.2020.

[23] S. Srivastava, App Invetiv, 4 Proven Ways to Avoid Data Leakage in An Android App, srpanj 2020., dostupno na: https://appinventiv.com/blog/4-proven-ways-to-avoid-data-leakage-in-an-android-app/, pristupljeno 28.08.2020.

[24] S. Topuzov, Secure Group, Malicious Apps Spy on You and Steal Your Data. How Can You Stop Them?, prosinac 2016., dostupno na: https://blog.securegroup.com/malicious-apps-spy-on-you-and-steal-your-data.-how-can-you-stop-them, pristupljeno 29.08.2020.

[25] B. Chauhan, Astra, All You Need to Know About Android App Vulnerability: Insufficient Cryptography, ožujak 2020., dostupno na: https://www.getastra.com/blog/app-security/all-you-need-to-know-about-android-app-vulnerability-insufficient-cryptography/, pristupljeno 07.09.2020.

[26] V. Raman, A. Mettler, Y. Zhang, M. Isberner, FireEye, Cryptographic Vulnerabilities in Android Applications, siječanj 2015., dostupno na: https://www.fireeye.com/blog/threat-research/2015/01/cryptographic_vulner.html, pristupljeno 07.09.2020.

[27] OWASP, M5: Insufficient Cryptography, dostupno na: https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography, pristupljeno 07.09.2020.

[28] A. Agrawal, Android Application Security Part 13 – Broken Cryptography, listopad 2015., dostupno na: https://manifestsecurity.com/android-application-security-part-13/, pristupljeno 09.09.2020.

[29] Srinivas, Infosec, Broken Cryptography, veljača 2015., dostupno na: https://resources.infosecinstitute.com/android-hacking-security-part-16-broken-cryptography/, pristupljeno 09.09.2020.

[30] Hashes, dostupno na: https://hashes.com/en/decrypt/hash, pristupljeno 09.09.2020.

[31] U. Kiguolis, Remove Tizi Android virus (Simple Removal Guide) – Tutorial, studeni 2017., dostupno na: https://www.2-spyware.com/remove-tizi-android-virus.html, pristupljeno 10.09.2020.

[32] Google Play Protect security engineers, Tizi: Detecting and blocking socially engineered spyware on Android, studeni 2017., dostupno na: https://security.googleblog.com/2017/11/tizi-detecting-and-blocking-socially.html, pristupljeno 10.09.2020.

[33] Z. Zorz, Tizi backdoor rooted Android devices by exploiting old vulnerabilities, studeni 2017., dostupno na: https://www.helpnetsecurity.com/2017/11/28/tizi-backdoor-rooted-android/, 10.09.2020

[34] M.K. McGee, Weak Encryption Leaves Mobile Health App at Risk for Hacking, prosinac 2018., dostupno na: https://www.careersinfosecurity.com/weak-encryption-leaves-mobile-health-app-at-risk-for-hacking-a-11833, pristupljeno 10.09.2020.

[35] CVE Details, dostupno na: https://www.cvedetails.com/cwe-details/326/cwe.html, pristupljeno 15.09.2020.

[36] CVE Details, dostupno na: https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=2&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&month=0&cweid=326&order=3&trc=111&sha=ad0f12443707ab86de50d718072dddbfb7142fbb, pristupljeno 15.09.2020.

[37] CVE Details, dostupno na: https://www.cvedetails.com/cve/CVE-2018-19001/, pristupljeno 15.09.2020.

SAŽETAK

Cilj ovog rada je opisati probleme otjecanja podataka uzrokovanog zlonamjernom aplikacijom i loših kriptografskih rješenja te na primjeru pokazati najbolja rješenja. Za problem otjecanja podataka opisana je jedna zlonamjerna aplikacija koja se između ostalog pojavila u obliku aplikacije za vježbanja, a zapravo se radilo o špijunskom programu koji je uzrokovao otjecanje podataka iz aplikacija za potporu društvenim mrežama. Prilikom pokušaja instaliranja te aplikacije od treće strane na uređaj, Android sustav izdao je nekoliko upozorenja te na kraju blokirao instalaciju, a korišteni antivirusni alat također je prepoznao opasnost koju je i uklonio s uređaja. Za primjer lošeg kriptografskog rješenja, odabrana je jedna aplikacija kojoj je američka vlada otkrila slabosti u šifriranju zbog korištenja nepouzdanog i slabog algoritma s obzirom koje podatke o korisniku ima. Rješenje ovog problema je nepoznato, jer se korištene metode šifriranja pokušavaju skriti od napadača, ali je kao rješenje predloženo korištenje različitih ključeva za šifriranje za svakog pojedinog korisnika kako bi se, u slučaju napada na jednog, zaštitili ostali korisnici.

Ključne riječi: Android, kriptografska rješenja, otjecanje podataka, sigurnost, zlonamjerne aplikacije.

ABSTRACT

The goal of this paper is to describe the problems of data leakage caused by malicious application and poor cryptographic solutions and to show the best solution of their examples. One malicious application was described for the problem of data leakage, which appeared, among other things, in the form of a fitness application, but it was actually a spyware that caused data leakage from social media applications. When trying to install this application from a third party on the device, the Android system issued several warnings and eventually blocked the installation, and the antivirus tool which was used also recognized the danger and removed it from the device. As an example of a poor cryptographic solution, the HealthSuite application was chosen, since the US government revealed weaknesses in encryption due to the use of an unreliable and weak algorithm given the user data it has. The solution to this problem is unknown because developers try to hide used encryption methods from attackers, but as a solution it is suggested to use different encryption keys for each individual user to protect other users in case of an attack on one.

Keyword: Android, cryptography, data leakage, malware, security.

ŽIVOTOPIS

Mia Štefanec rođena je 16. ožujka 1997. godine u Osijeku, Hrvatska. Osnovnu školu Dobriša Cesarić u Osijeku završava 2012. godine te upisuje I. gimnaziju Osijek. Godine 2016. upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.

POPIS SLIKA

PRILOZI

Prilog 1. Završni rad u datoteci docx.

Prilog 2. Završni rad u datoteci pdf.