Sigurnosni aspekti mobilnih aplikacija

Dumančić, Ilija

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:518740

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-20**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

SIGURNOSNI ASPEKTI MOBILNIH APLIKACIJA

Završni rad

Ilija Dumančić

Osijek, 2016.

Obrazac Z1 - Obrazac za ocjenu završnog rada

Osijek,

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Ilija Dumančić
Smjer:	Komunikacije i Informatika
Mat. br. studenta, godina upisa:	3731, 2013.
Mentor:	Doc.dr.sc. Krešimir Grgić, dipl.ing.
Sumentor:	
Naslov završnog rada:	**Sigurnosni aspekti mobilnih aplikacija**
Primarna znanstvena grana rada:	**Telekomunikacije i informatika**
Sekundarna znanstvena grana (ili polje) rada:	
Predložena ocjena završnog rada:	
Kratko obrazloženje predložene ocjene:	

Potpis sumentora:	Potpis mentora:

Dostaviti:

1. Studentska služba

Korekcija ocjene Odbora za završne i diplomske ispite

Odbor za završne i diplomske ispite Elektrotehničkog fakulteta Osijek, temeljem članka 11. Pravilnika o završnim ispitima na preddiplomskim i stručnim studijima Elektrotehničkog fakulteta Osijek donio je odluku o korekciji konačne ocjene završnog rada na ____ sjednici Odbora održanoj dana _____ godine.

Korigirana konačna ocjena završnog rada:	_____

Potpis predsjednika Odbora:

_____

Dostaviti:

1. Studentska služba

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

IZJAVA O ORIGINALNOSTI RADA

Osijek,

Ime i prezime studenta:	Ilija Dumančić
Studij :	Preddiplomski studij elektrotehnike
Mat. br. studenta, godina upisa:	3731,2013.

Ovom izjavom izjavljujem da je rad pod nazivom**:**

Sigurnosni aspekti mobilnih aplikacija

Izrađen pod vodstvom mentora

Doc.dr.sc. Krešimir Grgić, dipl.ing.

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

Sa mnoštvom aplikacija koje su danas dostupne na tržištu većina korisnika nije svjesna opasnosti koje one donose. Strategije za suočavanje sa različitostima ovih prijetnji potrebno je bolje sagledati. Otkad je prvi veliki crv (virus) napao glavni sustav u 80-ima, sigurnost je bila veliki problem za kompjuterske sustave [1]. Kako se računalstvo sve više bavilo mobilnim platformama, tako su napadi i virusi pronašli svoje nove ciljeve u njima. Sve prisutna uporaba mobilnih uređaja stavlja veliki pritisak na sigurnost pojedinih aplikacija. Zbog toga što ovi uređaji sadrže velike količine osobnih informacija one su jako zanimljive mete za napadače koji traže financijsku dobit. Istraživači također tvrde da oko pola napadača je u mogućnosti ukrasti osobne podatke i pratiti korisnike. Daljnje stvari komplicira činjenica da su mobilni uređaji ograničene računalne snage i korisničkog sučelja, što omogućuje napadačima lakši pristup podacima i skrivanje same radnje. U radu će biti opisani sustavi fokusirani na analizu, detekciju i razvoj štetnih aplikacija. Kao tradicionalni pristupi primijenjeni sigurnosnom sustavu tako i analize mogu biti izvršene sa statičkom ili dinamičkom tehnikom, ili sa obje.

Kako bilo, sigurnost mobilnih aplikacija ne smije biti samo fokusirana na podacima i aplikacijama. Mobilne platforme koriste raznovrsne nove mogućnosti koje nisu mogle biti primijenjene na osobnim računalima. Napadač može kompromitirati sustav spojen na mobilni uređaj putem ranjivosti na nekoj od točaka. Sve više razloga za brigu daju uređaji kao što su pametni satovi, naočale itd. Iako može ugroziti sigurnost uređaja, mnogi korisnici pokušavaju nadograditi svoje mobilne uređaje i održati prava na pristup za potpunu kontrolu i prilagođavanje, što će u radu biti detaljno opisano. Mobilni uređaji ne mogu se samo osloniti na sigurno okruženje. Cijeli sigurnosni sustav i uporaba mora se smatrati prikladnom. U daljnjem radu biti će opisan slučaj istraživanja na aplikacijama za pametne telefone i druge pametne uređaje (kao npr. pametni TV). Pokazat će se kako njihov uređajno-agnostički pristup može biti primijenjen u kontekstu projekta. Nadalje, demonstrirat će se kako se ciljevi upotrebljivosti, prijateljstva i sigurnosti mogu razmatrati istovremeno. Njihova trgovina aplikacijama je zanimljiv sustav sa značajkama koje uključuju naprednu sigurnost, biometrijsku autorizaciju, više stupanjsku autorizaciju, pokretnu navigaciju, ocjenu aplikacijske reputacije i upravljanje identitetom.

1.1. Zadatak završnog rada

Problematika sigurnosti od velike je važnosti u svim vrstama komunikacijskih mreža i na svim vrstama računalnih platformi. Budući da su u posljednje vrijeme mobilne računalne platforme (u svim svojim oblicima) doživjele značajnu ekspanziju i povećanje broja korisnika, sve su izraženiji problem njihovi sigurnosni aspekti, budući da na tržištu postoji ogroman broj mobilnih aplikacija koje korisnici masovno koriste, uglavnom nesvjesni njihovih sigurnosnih ranjivosti i problema do kojih može doći njihovom zlouporabom. Potrebno je detaljno i sustavno analizirati problematiku sigurnosti mobilnih platformi i aplikacija. Istražiti i objasniti sigurnosne prijetnje i ranjivosti koje postoje u ovakvom okruženju, kao i odgovarajuće metode i tehnike za detekciju zlonamjernih aktivnosti i poduzimanje odgovarajućih protumjera. Analizirati sigurnosne algoritme, protokole i okvire koji se primjenjuju na suvremenim mobilnim platformama.

2.SIGURNOSNI ALATI I DIFERENCIJALNA ANALIZA

Otkrivanje virusa u mobilnim aplikacijama je postalo kompleksno jer su kreatori sve vještiji u skrivanju istih. Alterdroid je alat koji koristi dinamičku analizu koja uspoređuje i traži razliku između originalne verzije aplikacije i mogućih zaraženih.

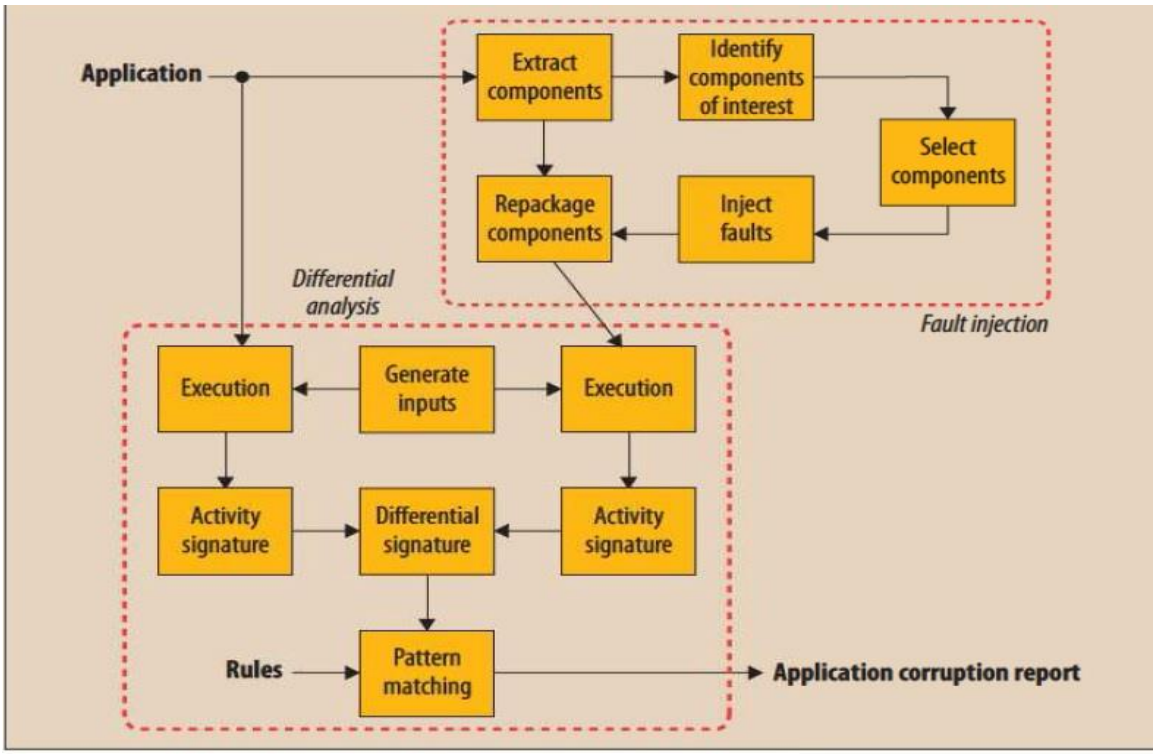
Pametni telefoni rapidno rastu u svojoj popularnosti sa svojim iznimno jakim računarstvom, komunikacijom i memorijom. Zadnja istraživanja su pokazala da korisnici sve više provode vremena na mobilnim aplikacijama skoro kao i na osobnim računalima prilikom korištenja Web pretraživača. Ovom naravno prethodi to što je sve više aplikacija na tržištu koje korisnici mogu preuzeti iz trgovina sa svoga pametnog uređaja. Pametni telefoni također predstavljaju značajan broj sigurnosnih i osobnih prijetnji koji su u mnogim aspektima alarmantniji i opasniji nego u okruženju gdje se koriste osobna računala. Većina pametnih telefona je opremljeno sa mnogim senzorima koji otkrivaju našu lokaciju, geste, kretnje i fizičke aktivnosti. Virus koji boravi u pametnom telefonu mogao bi naštetiti privatnosti uređaja. Većina današnjih platformi izgrađena je na modificiranoj verziji desktop operativnog sustava i sadrži neke od tradicionalnih sigurnosnih postavki, one također sadrže više izrađene sigurnosne modele koje najbolje pristaju ovim uređajima i njihovoj arhitekturi. Jedna od značajnijih postavki je i ta da ograničava pristup aplikaciji vašim podacima i druge servise. Ograničenje također provodi kroz sigurnosne ograde i druge dizajnirane mjere koje reguliraju među-aplikacijsku komunikaciju. Otkad je glavni izvor sigurnosnih problema postao kako ograničiti dolaznost aplikacija iz ‘’treće razine’’, sigurnosne razine su se povećale te je primarni cilj obrana od njih. Neke trgovine provode revizijski proces preko aplikacija koje uključuju neka sigurnosna testiranja. Ovu prijetnju dodatno komplicira činjenica da se ne zna koja aplikacija je prijetnja a koja nije, što predstavlja još veći izazov, specijalno za one aplikacije koje su mogu nazvati ‘’siva zona’’, to su one aplikacije koje nisu skroz zaražene ali njihova konstitucija predstavlja opasnost po korisnika i njegovu sigurnost.

2.1. Razvoj virusa u operacijskim sustavima

Rapidna prodaja pametnih telefona usko je povezana i sa sve većim rastom virusa na tim platformama. Virus pametnih telefona je postao prilično unosan posao zbog postojanja velikog broja potencijalnih ciljeva i dostupnosti ponovnoj uporabi virusne metodologije koja ga čini veoma lakim za izradu novih uzoraka. Istraživanja su uspješna ali sa mnogo neriješenih problema. Ovo je ubrzalo potrebu za pametnim analizama i tehnikama za sprječavanje virusa u njihovoj dnevnoj funkciji. Tomu više što je virus pametnih telefona postao nečujan, sa najnovijim primjerima baziranim na određenom kodu prekrivenih tehnika za izbjegavanje sigurnosnih zamki. Primjerice DroidKungFu virus koji će biti detaljno opisan kasnije, Android virus koji se prvi put pojavio u lipnju 2011.g. i iznjedrio 6 verzija, najviše je distribuiran kroz službene trgovine ili alternativnim trgovinama tako što se sam nadodavao u različite legitimne aplikacije. Takav koristan teret je kodiran u mapu aplikacije i dekodiran za vrijeme izvođenja koristeći ključ koji je smješten u lokalnoj varijabli pripadajući specifičnoj klasi veličina. Više sofisticirane tehnike skrivanja određenih tehnika, osobito za kodove, u nastojanju su da stvore dodatnu prepreku u virusnoj analizi, koje se na kraju moraju oslanjati na kontroliranu dinamičku analizu za otkrivanje potencijalnih dijelova koda.

2.2. Alterdroid

Alterdroid je alat koji se koristi za otkrivanje zamaskiranih virusa koji su skriveni u dijelovima aplikacijskog paketa. Takve komponente su često skrivene izvan aplikacijskog glavnog koda. Ključna ideja Alterdroid-a sastoji se od analiziranja ponašanja razlike između originalne aplikacije i izmijenjene verzije sa pažljivo uvedenim modifikacijama ili greškama. Takve modifikacije su napravljene da nemaju vidljive efekte na aplikacijska izvršavanja, osigurava da izmijenjena verzija je zapravo ono što bi trebala biti i nema skrivenih funkcionalnosti. Kao npr. mijenjanje vrijednosti piksela u fotoaparatu ili nekoliko znakova u dekodiranju poruke, na njih zapravo nema utjecaja. Ako se nakon takvih dinamičkih akcija sruši ili mrežna veza ne pronađe, mogao bi biti slučaj da se fotografija sastoji od dijelova koda ili se niz sastoji od mrežne adresa ili URL-a. Na sl.2.1. je prikazana visoko stupanjska arhitektura Alterdroida, koja se sastoji od dva dijela: unosa grešaka i diferencijalne analize.



Sl. 2.1. Arhitektura Alterdroida [3]

Prvi dio uzima aplikaciju kao kandidata za unos i generiranje greške. Najprije vadi sve dijelove aplikacije i identificira one koji su sumnjivi zbog skrivenih virusa. Takva identifikacija temelji se na detekciji anomalija baziranih pri uspoređivanju određenih statističkih obilježja za dijelove sastavljene od različito definiranih modela kao što su: kod, slike, video, tekstovi, baza podataka, itd. Zatim unosi grešku u dio komponenti kandidata i raspakirava zajedno sa izmijenjenom verzijom u novu aplikaciju. Ovaj proces podržava istodobno unošenje različitih grešaka u različite dijelove i pokreću se u algoritmu koji je zadužen za traženje skrivenih funkcionalnosti. Diferencijalna analiza prilikom izvršavanja uzima i originalni dio i dio gdje je unesena greška u aplikaciji u istim uvjetima te promatra i snima ponašanje, te bilježi oba *aktivna potpisa.* Takvi potpisi su sekvencijalni tragovi aktivnosti prilikom izvršavanja aplikacije, kao otvaranje mrežne konekcije, slanje ili primanje podataka, preuzimanja komponenata, slanje SMS poruke i surađivanje sa cijelim sustavom podataka. Tada, kao u operaciji ispravljanja niza, izračunava se Levenstheinova udaljenost (udaljenost između dva znaka) da bi se vratio diferencijalni potpis, to bi bio popis vidljivih razlika u pogledu umetanja, brisanja i zamjene [2]. Konačno, analizira se diferencijalni potpis kroz proces uzimanja uzoraka i uspoređivanja razlika dvaju uzoraka, gdje se mogu naći skrivene funkcionalnosti.

2.2.1. Identifikacija značajnih komponenti

Aplikaciju možemo prikazati kao skup komponenti, svaka ima svoju klasu (kod), kao i kod drugih sredstava kojima se pristupa kod izvođenja, kao što su osobni podaci. Komponente imaju tip, uključujući kod, sliku, video, bazu podataka, itd. Definiramo značajnu komponentu (*engl. CoI-component of interest)* kao onu koja ne pristaje modelu definiranu za sve komponente toga tipa [3]. U postojećoj verziji Alterdroid, model mjeri statističke značajke, kao očekivanu entropiju, distribuciju bajta ili prosječne veličine. Također se primjenjuje nekoliko istraživanja na temelju vrste datoteke i njegovih dodataka. Alterdroid izračunava te značajke iz skupa podataka komponenata istog tipa. Za svaki model tu je funkcija koja provjerava je li komponenta usklađena sa svojim modelom. Npr. ako je model bajt raspodjela, onda funkcija može biti test ocjene prilagodbe između modela i komponente bajt raspodjele. Takvi jednostavni modeli su dovoljni da se uoči najčešći i vrlo jednostavan način skrivanja određenih poteza uočenih u mobilnim virusima, uključujući zamaskirani kod kao dopunsku multimedijsku datoteku, podatkovna veza skrivena u varijablama teksta itd. Stanje neispravnosti može biti ubačeno u aplikaciju izmjenom jedne ili više njenih komponenti. Operator prilikom unosa greške *(engl. FIO-fault-injection operator*) kreira modificiranu aplikaciju tako što zamijeni komponentu sa njezinom malom izmjenom. Ovo može ali i ne mora biti primjetno u razlici prilikom izvršavanja aplikacije. Operator (*engl. FIO-fault-injection operator*) taj promijenjeni dio samo djelomično pokazuje, kao što je zamaskirani virus teško pronaći. Operator se također može ubaciti u ne štetni dio za bolje razumijevanje njegove funkcije u aplikaciji. Neki operatori tretiraju komponente kao dijelove niza i primjenjuju promjene kao što je nasumce preklapanje bitova ili zamjena dijela komponente sa nekom drugom iz dijela komponenti aplikacije. Osim toga, specifični operator za podatke može biti koristan za izmjenu podataka prilikom očuvanja ispravne sintakse kada je fokus na promjeni sadržaja bez vraćanja objekta nekoristan. Alterdroid unosi značajnu komponentu (*engl. CoIs-components of interest*) koja je identificirana sa greškom te ju ponovno sastavlja, zajedno sa ostalim komponentama kako bi generirali mutiranu aplikaciju [4]. Ovaj proces također može generirati nekoliko takvih aplikacija, kao što postoji više načina primjene operatora za različite komponente u setu značajnih komponenti. Alterdroid generira aplikacije koje su greškom ubrizgane jednu po jednu te primjenjuje diferencijalnu analizu. Sve dok ne pronađe štetno ponašanje ponavljanje radnja se ponavlja cijelo vrijeme.

Svi operatori u Alterdroidu se ne razlikuju, što znači da se greškom unesena aplikacija ponaša normalno kao izvorna, pod uvjetom da promijenjena komponenta ne uključuje bilo kakve skrivene

funkcije. To omogućava učinkovitije procese greškom unesenih aplikacija temeljene na sastavu da je operator neprimjetan i nerazlučiv za druge. Prema tome, ako se taj isti operator primjeni na više dijelova i ako je samo u jednom skrivena štetna funkcija, nastala aplikacija će se ponašati kao štetna komponenta koja je bila ubrizgana na početku. Cjelokupni greškom unesen proces, koji je prepleten s procesom diferencijalne analize, je u biti algoritam za pretraživanje koji identificira sve moguće prijeteće komponente.

2.3. Primjena diferencijalne analize

Aplikacija zahtjeva usluge kroz niz slobodnih poziva sustava. Ponašanje aplikacije može se opisati kroz djelatnosti koje provodi. U nekim slučajevima bilo bi jedan na jedan dopisivanje između djelatnosti i poziva sustava, dok u drugim djelatnost obuhvaća niz poziva sustava provedenih danim redoslijedom. Aplikacijsko izvršenje može ići različitim smjerovima, ovisno o ulazima koje zada korisnik i stanja okruženja u kojem se izvršava radnja. Alterdroid sažima to vidljivo ponašanje prilikom izvršenja u neki oblik aktivnog potpisa, što predstavlja vrijeme koje je bilo zadano za izvršenje naredbe. Ključni zadatak u Alterdroidu je analiziranje ponašanja između originalne aplikacije i modificirane verzije nakon unošenja krivog dijela. Iznosi podatke nakon snimanja i analiziranja između različitog ponašanja obje verzije. Rješava problem tako što koristi metodu niz po niz (*engl. string-to-string*), u kojem razliku predstavlja minimalan broj izmijenjenih operacija, unosa, brisanja, i zamjena, tada prenosi jedan potpis u drugi te minimalna razlika u potpisima postaje vidljiva [4]. Alterdroidova shema analize smanjuje svojstva aplikacije zbog prisutnosti ili odsutnosti određenih obrazaca u razlikama potpisa između originalne verzije aplikacije i izmijenjene. Koncept se može prikazati u tri primjera:

Primjer 1.

Pretpostavimo da aplikacija koristi sliku kao ikonu na svom korisničkom sučelju. Modificiranje nekih piksela takvih ikona, ili zamjena sa nekom drugom važećom ikonom ne bi trebalo utjecati na izvršenje aplikacije. Ako se modificirana aplikacija ponaša drugačije od originala pod istim uvjetima možemo zaključiti da je originalna verzija sadržavala neki dio neodgovarajuće funkcionalnosti, nešto kao dio zamaskiranog dijela koda.

Primjer 2.

Neka je *v* varijabla takva da njezin sadržaj ne utječe na izvršavanje programa. Npr. varijabla može biti sadržaj niza koda koja sadrži poruku o pogrešci koja će se pokazati u određenom trenutku. Takav niz će se koristiti u postojećem virusu da sakrije URL koji upućuje na poslužitelja odakle virus može preuzeti kod, dobivanje upute, slanje podataka, itd. Za izbjegavanje otkrivanja, niz je najčešće zamaskiran, uglavnom kroz jednostavne zamjenske sheme, i URL se otkriva na kraju izvršenog vremena. Bilo koja modifikacija niza rezultirat će oštećenim URL-om koji će spriječiti uspostavu veze.

Primjer 3.

Moguće je otkriti iz kojeg dijela cure informacije kroz različite odgovarajuće senzore, npr. GPS, itd., te nakon procesa ubrizgavanja greške diferencijalni potpisi nemaju pristup takvim senzorima i mrežnoj vezi.

2.3.1. Alterdroid u praksi: Tri slučaja studije

Za bolji prikaz kako trgovina operatora i sigurnosna analiza može koristiti Alterdroid za pomoć prilikom traženja zamaskiranih virusa, predstavljaju se tri slučaja studije [2]: DroidKungFu (DKF), AnserverBot (ASB) i GingerMaster(GM). Ova tri primjera sadrže zamaskirane tehnike određenih poteza u različitim stupnjevima složenosti, kao i zlonamjerne značajke u virusu za pametne uređaje kao što su komande za kontrolu kao funkcije i curenje informacija.

DroidKungFu

Glavni zadatak DKF-a je sakupljanje određenih podataka o zaraženom Android uređaju, uključujući IMEI broj (Međunarodni broj mobitela za identifikaciju), model mobitela i verziju operativnog sustava. Većinom je rasprostranjen u otvorenim ili alternativnim trgovinama prilikom raspakiravanja, te se nadodaje u zlonamjerne radnje u raznim legitimnim aplikacijama. Aplikacije zaražene sa DKF-om se distribuiraju zajedno s *root* verzijom skrivenom unutar aplikacije. To ometa statičku analizu, moguće ju je otkriti samo za vrijeme izvođenja. Ubačena je jedna varijanta DKF-a u Alterdroid, iz koje su prvo raspakirane sve varijante značajnih komponenti *(engl. CoI-components of interest*),

unesene su varijante greške u komponente i zatim primijenjena diferencijalna analiza prilikom izvođenja rezultirajuće aplikacije u usporedbi sa originalom. Sadržavao je oko 170 izvornih datoteka, 153 PNG (*engl. Portable network graphics*) datoteka, 6 MP3 datoteka, 2 XML datoteke, 1 DEX datoteka, i jedan RSA ključ (koristi se za osiguranje podataka prilikom slanja). Svaki od dijelova je bio sumnjiv da sadrži maskiranu funkcionalnost.

Slika 2.2.a. prikazuje različito ponašanje Alterdroida kroz period od 2 minute. Zelena polja predstavljaju pozitivno ponašanje bez štetnih djelovanja, dok crna polja predstavljaju DKF. Alterdroid je otkrio da je tekstualna datoteka modificirana. Kasnije je otkriveno da datoteka sadrži RAC (*engl. Real Application Clusters*) i utvrdili da je onemogućen maliciozni pristup, kroz mrežni pristup i curenje informacija, a kasnije i obavljanje nekih drugih funkcija. Ova se analiza podudara sa prethodnim objavljenim rezultatima DKF-a. U njegovom slučaju primjenjujući samostalne statičke metode detekcije nije dovoljno za prepoznavanje zlonamjernih radnji bez ljudskih dodatnih inspekcija. Specifično je da svaka varijanta koristi drugačiji ključ za kodiranje sakriven u dijelu koda. Čak i kada se koristi dinamička analiza, ova metoda samo iznosi grube činjenice o štetnom djelovanju virusa. Ustvari, ponašanje DKF-a je snažno isprepleten sa originalnom verzijom koda tako da je neki od njegovih ključeva aktivan, kao što je mrežna veza, što je lako vidljivo.

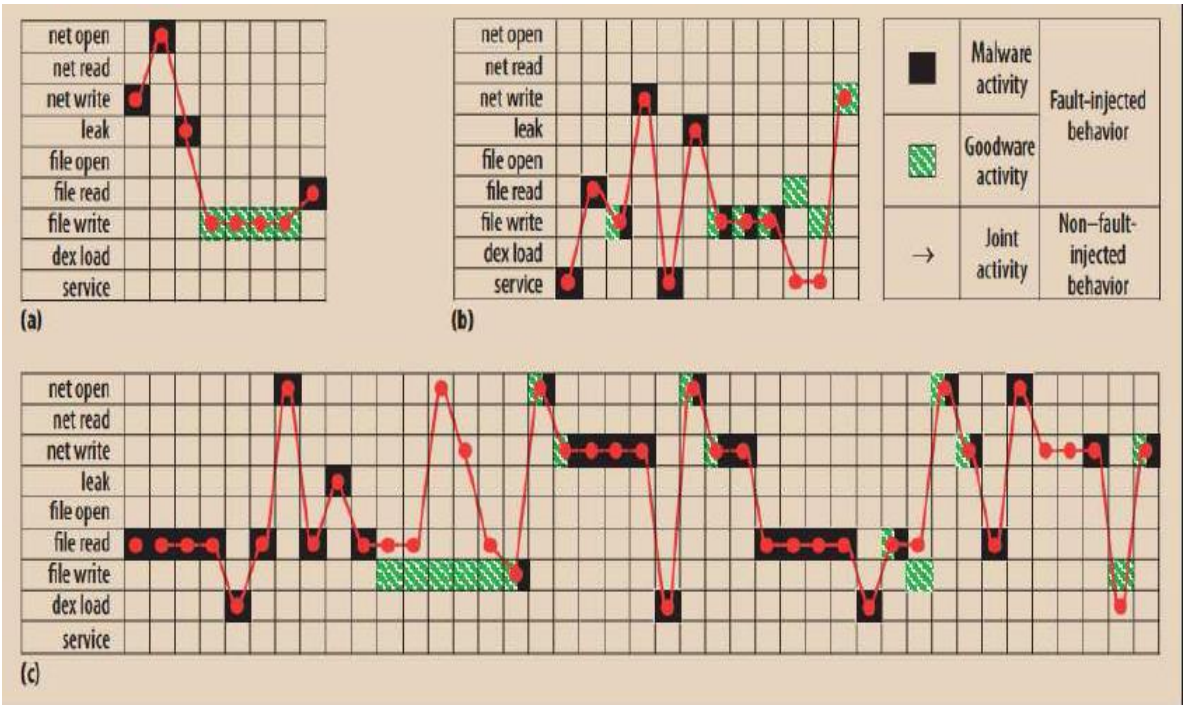
AnServerBot

ASB primjerak koji je analiziran je sličan prvoj verziji DKF-a u smislu sofisticiranosti i razvojnoj strategiji. ASB uvodi ažuriranje komponenata koje omogućuje dohvat za vrijeme izvođenja aplikacije. Također sadrži naprednu anti-analitičku metodu za izbjegavanje otkrivanja: s jedne strane ASB uvodi integritet komponenti za provjeru da li je aplikacija izmijenjena, dok na drugoj strani radi štetu u izvorni kod aplikacije. Osim toga, ASB prikriva svoje unutarnje metode i klase. Dok je jedan instaliran, a drugi se dinamički učitava bez da je instaliran. ASB krije jednu od tih komponenata u mapu pod imenima *anservera.db* ili *anserverb.db*. Umeće novu komponentu pod imenom *com.sec.android.provider.drm* koja izvršava *root*, poznat pod nazivom Asroot. Kao i u slučaju DKF-a,može se primjetiti da sve varijante ASB-a sadrže nezanemarive količine značajnih komponenti. Uzorak koji je ispitan sastojao se od 78 izvornih podataka, uključujući 54 slike, jednu bazu podataka, jednu DEX datoteku i ZIP datoteku. Nakon nekoliko iteracija postupka unošenja greške, Alterdroid je identificirao stvarni teret unutar baze podataka, kao i ponašanje u vezi sa datotekom. Točnije,

aktivira *CoI* nakon promatranja. Kada je greška unesena u bazu podataka, ASB-ova cjelovita provjera prekida njegovo izvršenje i daje rezultate slične koje je očekivao od originalne aplikacije. Na slici 2.2.c. prikazano je različito ponašanje preko dvije minute. ASB prvo uspostavlja mrežnu vezu nakon učitavanja poslane datoteke. Nakon toga nastavlja čitanje podataka. Legitimna aplikacija koristi mrežu kao što i ne dopušta propuštanje informacija.

GingerMaster

GM je prvi poznati virus koji koristi *root* proces za eskalaciju. Sa djelovanjem je krenuo još od Android verzije 2.3., te djeluje i na novim verzijama. Glavni cilj je da izfiltrira privatne podatke kao što su osobni podaci, IMEI broj, MSI broj, listu kontakata spremljenih u telefonu. GM se općenito pojavljuje u akciji *root-a*, pohranjen kao PNG ili JPEG datoteka. Odmah nakon zaraze uređaja, GM se spaja na server i pokreće nove radnje. Analizirani su GM uzorci sa oko 60 izvora, od kojih su 30 slike u različitim formatima. Od ovih slika, Alterdroid je detektirao 4 kao opasne. Kasnije detaljne analize su pokazale da su bile deformirane u PNG formatu koji se sastoji od nekoliko ASCII skripti. Alterdroid je također sposoban da otkrije takve deformirane slike koje igraju ključnu ulogu u pokretanju štete u originalnoj aplikaciji, uključujući ASCII skripte.
Na slici 2.2.b. je vidljivo drugačije ponašanje preko dvije minute nakon što je Alterdroid unio takvu sliku. GM je započeo izvršavanje usluga koje obavljaju neke operacije prije nego što krene curenje informacija putem mreže. Čak i kada su štetne komponente skrivene, Alterdroid je u mogućnosti razlikovati ih i pomoći identificirati temeljne štetne radnje.



Sl. 2.2. Različito ponašanje Alterdroida u sva tri slučaja: a)DroidKungFu, b)GingerMaster, c)AnserverBot, [3]

2.3.2. Analitički rezultati

Alterdroid je testiran na skupu podataka od oko 300 štetnih aplikacija, onih u ‘’sivoj zoni’’ i sigurnih aplikacija. Svaka aplikacija je izvršena tokom 2 minute, iz razloga koji su bili navedeni prethodno više od polovice ih je uspješno otklonjeno.

Tablica 2.1.prikazuje eksperimentalne rezultate s obzirom na obje izvedbe diferencijalne analize i performansi za otkrivanje uspješnosti. Diferencijalna analiza se mjeri kao vrijeme potrebno Alterdroidu po aplikaciji i ubrizganih greškom aplikacija, uključujući trajanje učinkovitosti analize. Efikasnost detekcije je dano u pozitivnoj stopi, to jest omjer između broja ispravljenih pozitivnih slučajeva i ukupnog broja ispitanih aplikacija. Što se tiče legitimnih aplikacija, prilikom analize nije primijećena nikakva sumnjiva radnja. Te su aplikacije označene kao dobre ali je označeno jedno ili više pravila koje idu uz aplikacije iz ‘’sive zone’’. Alterdroid je više alat za analizu nego otkrivanje, glavni cilj je identificirati zamaskirani dio i izvijestiti ih u analizi. Klasificiranje skrivenih funkcionalnosti kao štetne ili ne je sasvim drugačiji problem koji u većini slučajeva ovisi o korisnikovoj uporabi i skrivanju privatnih informacija. Jedan značajan aspekt Alterdroida je da se

proces dinamičke analize računa za oko pola vremena od ukupnog. Npr. analizirajući CoIs od jedne aplikacije i ubrizgavanje greške traje oko 10 sekundi, dok je pokretanje Android 2.3 verzije sa 512 Mbit-a memorije trajalo oko 160 sekundi. Brojke pokazuju da je izvedba cijelog procesa diferencijalne analize snažno utjecala na potrebne pripreme.

Vrijeme koje je potrebno za izdvajanje diferencijalnog potpisa ovisi o tipu virusa i njegovom mehanizmu za pokretanje štete, kao što se vidi za GM i GM+. Međutim, lako je primjetno da većina primjeraka to brzo izvršava, trenutak promatranja je generalno dovoljan za pouzdanu procjenu.
Točnost detekcije Alterdroida je bila vrlo dobra, osobito u odnosu sa maskiranim virusom. Prosječnost pada ispod 97% samo u odnosu sa GM-om ali i to je s vremenom ispravljeno za dinamičku analizu. Za primjer su rezultati za ASB koji pokazuju prosječan broj 3,90 podudaranja nekoliko različitih potpisa. Jedan izazov s kojim se suočava prilikom analiziranja aplikacija u ‘’sivoj zoni’’ je kako identificirati gdje je štetno ponašanje, a gdje nije. Mnoge legitimne aplikacije su skoro zaštićene i sigurne i nema naznaka štete ali iznosi mnoge aktivnosti koje donose korisnike u opasnost. Analize su pokazale da većina takvih sumnjivih ponašanja su povezani s aplikacijskom točkom povezivanja lokalnih podataka i prenošenje istih preko mreže. Alterdroid učinkovito nadopunjuje statički alat za analizu koji se fokusiraju na pregledu koda komponenata i na taj način mogu se zanemariti dijelovi koda skriveni u objektima ili koji su jednostavno maskirani. Alterdroid je zamišljen kao okvir za opće namjene sa svestranim nadopunama što se tiče arhitekture koja se može postići u mnogim dijelovima. Dokazano je da Alterdroidova efikasnost ne ovisi samo o trajanju dinamičke analize nego i o brzini djelovanja štetnih radnji.

Tablica 2.1. Prikaz rezultata diferencijalne analize Alterdroida u zaraženim aplikacijama [3]

App category	Apps	Components of interest (average per app)	Fault-injection operations (average per app)	Matches (average per app)	Overhead (average per fault-injection operation and app)	True positives	False negatives	Accuracy (%)
DroidKungFu	34	6.11	6.11	11.71	289.93 s	33	1	97.06
AnserverBot	187	1.35	1.35	3.90	246.22 s	186	1	99.47
GingerMaster	4	4.00	4.00	6.00	248.23 s	3	1	75.00
GingerMaster+	4	4.00	4.00	3.00	1,026.01 s	4	0	100.00
Grayware	16	2.88	2.88	4.19	248.24 s	18	0	100.00
Goodware	81	0.57	0.57	0.00	0.00 s	81	0	100.00

2.4. RootGuard

Iako je popularan za postizanje potpune funkcionalnosti, sami postupak *root-a* Androida dovodi uređaj do opasnosti. RootGuard nudi zaštitu od virusa sa raznim privilegijama koje pružaju korisniku fleksibilnost i kontrolu [5].

Cijenjen zbog svoje pristupačnosti i širokoj primjeni, Android je svjetski najpoznatiji operativni sustav. Još uvijek, njegov sigurnosni model sprječava korisnike i aplikacije da iskoriste u punoj funkcionalnosti sustav. Konkretno, sve privilegije koje sa sobom nosi operacija *root-a* su ograničene, te su privilegije dopuštene jedino nekima.

Ovaj nedostatak koji nosi sa sobom ograničava ljudima potpunu korisnost svojih uređaja i aplikacijama za razvoj. Npr. korisnik ne može ukloniti unaprijed instalirani i rijetko korišteni *bloatware* (aplikacije koje zauzimaju dosta prostora te usporavaju rad, crpi bateriju) i sigurnosni softvare koji nema mogućnost braniti se protiv virusa u pravo vrijeme [5]. Za prevladavanje takvih ograničenja korisnici moraju dobiti *root* pristupe. Dok je i dalje poželjan, *root* je značajno opasan za sigurnost uređaja. Nakon *root*-a aplikacija ima pristup cijelom sustavu i nižoj razini hardvera, što naravno zlonamjerne aplikacije mogu iskoristiti, zaobići sigurnost Androida te ugroziti sustav. Za rješavanje takvih problema preporuča se RootGuard, koji je lagan i praktičan alat dizajniran da zaštiti Android uređaje nakon izvršenja *root*-a.

RootGuard omogućuje korisnicima uzeti aplikaciju koje su mu potrebne za operativne zahvate u skladu sa svojim parametrima, dok zadržava pojedine police za sigurnost Androida. U projektiranju RootGuarda, adresirani su različiti izazovi koji otkrivaju koju funkciju je potrebno pratiti, koje zamijeniti, koristeći sitnu memoriju za spremanje tih polica i izlažući ih ostalim virtualnim putem. Sa daljnjim razvojem uključujući stvarni svijet i konceptualni dio, dokazano je da RootGuard uspješno smanjuje napade virusa prilikom *root*-a.

2.4.1. *Root* Android-a i upravljanje *root* privilegijama

U Linuxu SU naredbe odnose se na različite vrste korisnika što omogućuje operatoru za prebacivanje iz jednog korisnika u drugi. U Androidu, te naredbe omogućuju samo procese koji pripadaju *rootu* ili će postati *root*. Glavni cilj *root*-a Androida je instaliranje izmijenjene i prilagođene verzije koja omogućuje bilo koji proces aplikacije. Dok se mehanika razlikuje ovisno o modelu uređaja, postojeće metode mogu se podijeliti u dvije kategorije:

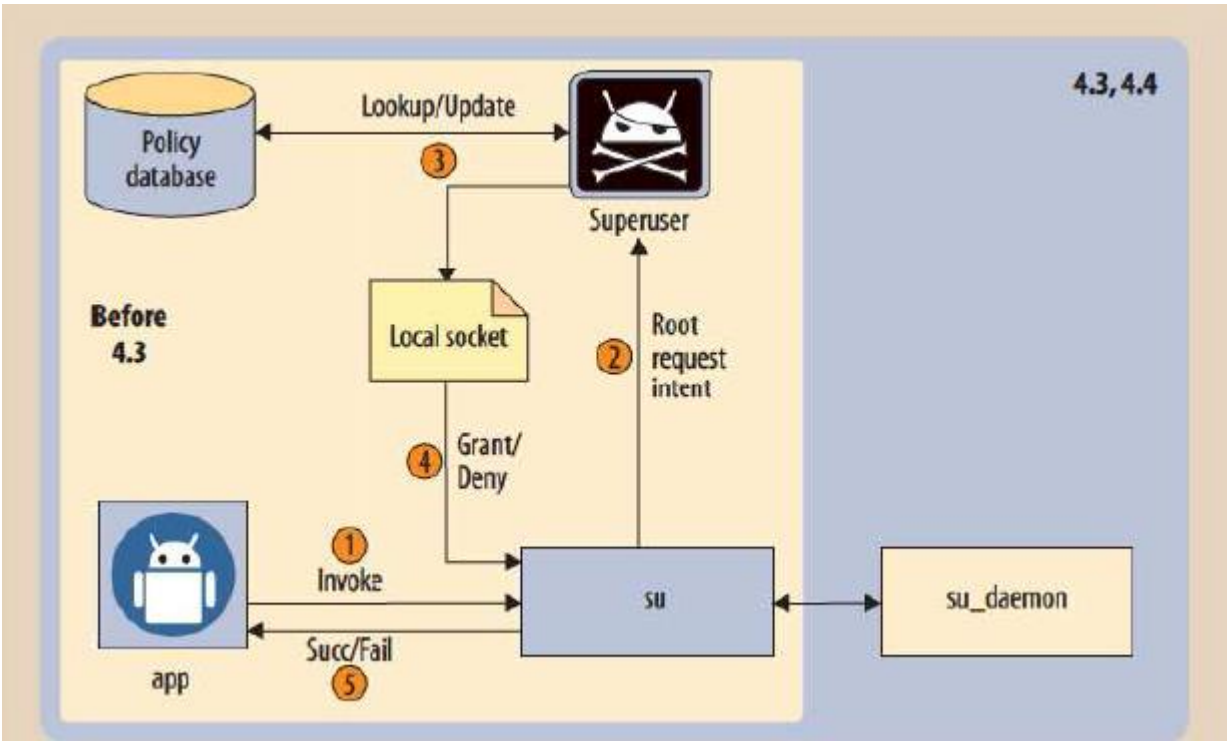
-Iskorištavanje ranjivosti sistema. Koristeći ovaj pristup potencijalno iskorištavanje programa raspoređenih na Android uređaju i zatim izvršavanje privilegija. Kada je uspješan proces, postupak otvara *root* proces, i zatim se sustav može prenamijeniti kao za čitanje/pisanje, dopuštajući izmijenjenim SU naredbama da budu kopirane u taj dio. Većina Android uređaja može biti nadograđeno na ovaj način.

-Treperenje slikom. Ovaj način dopušta korisnicima da izbaci cijeli skup datoteka ili, taj skup grupirati u jednu datoteku kao ‘’sliku’’ na različitim lokacijama u sustavu datoteka, to se može raditi koristeći SU naredbe pod imenom ‘’fastboot flash’’. Nekoliko uređaja podržava ovaj način *root*-a uređaja.

Za neiskusne korisnike, mnogo je automatiziranih alata koji su dostupni za lakše rukovanje. Kada se jednom kreiraju SU naredbe, slijedeći korak je instaliranje aplikacije za upravljanje *root*-a, tj. odobriti ili odbiti zahtjeve drugih aplikacija za pristup. Takve aplikacije najčešće određuju ‘*’Super Usera’’*. Iako postoje neke razlike među tim aplikacijama, svi oni djeluju pod istim uvjetima prilikom *root*-a što je vidljivo na slici 2.3. Originalni model, prikazan lijevo, jako malo je izmijenjen u Android 4.3, u novije vrijeme, tom modelu je potrebna modifikacija da bi se mogle smjestiti sve promjene vezane uz sigurnost samog Androida., prikazano desno.

Aplikacija koja zahtjeva *root* prvo poziva SU naredbe. Zatim naredbe šalju upit putem jednostavne poruke. Nakon toga ‘*’Super User’’* naredba konzultira svoju bazu podataka te pregledava da li je u mogućnosti poslati zahtjev. Nakon toga šalje povratnu poruku da li je u mogućnosti poslat zahtjev preko lokalne sabirnice.

Za ovaj osnovni tip *root*-a dodana je komponenta *su_daemon*, koja je ugrađena za lakše rukovanje Linuxom. On u dijelovima sprječava aplikaciju od *root*-a. *SU_daemon* je alat kojim lakše dolazi do ‘’podizanja’’ uređaja, bez ograničenja. Kada SU dopusti pokretanje *root*-a te naredbe bivaju proslijeđene na alat *su_daemon* za izvršavanje. Alat zahtjeva minimalno nadgledanje.



Slika 2.3. Menadžment modela s root privilegijama. [5]

2.4.2. Sigurnosni propusti u *root* alatima

Većina dostupnih alata za *root* zahtjeva veću sigurnosnu kontrolu. Prvo, u upitima za zahtjeve aplikacija moraju biti odobreni ili odbijeni. Štetne aplikacije mogu jednostavno zaobići taj proces ponašajući se kao legitimne aplikacije i dobivaju povjerenje korisnika prije izvršavanja štetne radnje. Kada je jednom odobren proces *roota*, postojeći alati neće prikazivati daljnje ponašanje. Oni pružaju samo grubu kontrolu, a korisnici koji su dozvolili *root* aplikacije možda u stvarnosti samo žele

određenu aplikaciju bez te opcije. Drugo, postojeći alati nisu u mogućnosti braniti se od štetnih napada, virus koji dobiva tu naredbu *root*-a može prouzrokovati beskorisnost tog sustava.

2.4.3. Dizajn i implementacija

Na slici 2.4. su opisane 3 glavne komponente RG-a [5].

1.''SuperuserEx''. Ova komponenta je izrađena na vrhu izvora ''Super User-a'' prilagođena od CyanogmenMod-a. Dodani su neki novu modeli koji nude korisniku GUI za nadograđivanje RG-a, ali je zadržan ostali dio dijelova originalnog SU-a.

2. Policy storage database. RG-ove police su trajno pohranjene u datotekama /*etc*/*rootguard*. Za nadogradnju istih koriste se kopije u drugoj memoriji, i vidljive su korisniku putem virtualne datoteke */dev/rootguard*. U driveru uređaja, dodana je funkcija Look Up() za sigurnost servera. U međuvremenu driver je definiran datotekama u *file_operations structure*. Kada korisnik doda novu policu u */etc/rootguard*, driver je tada obavješten od ''SuperuserEx-a'' kroz Netlink za sinkroniziranje nove police u memoriji. Također je dodana skripta za Android */init.rc* u mapu */etc/rootguard*. Za izbjegavanje natrpavanja memorije, dizajnirana je kompaktna polica koja minimalizira veličinu sveukupne police.

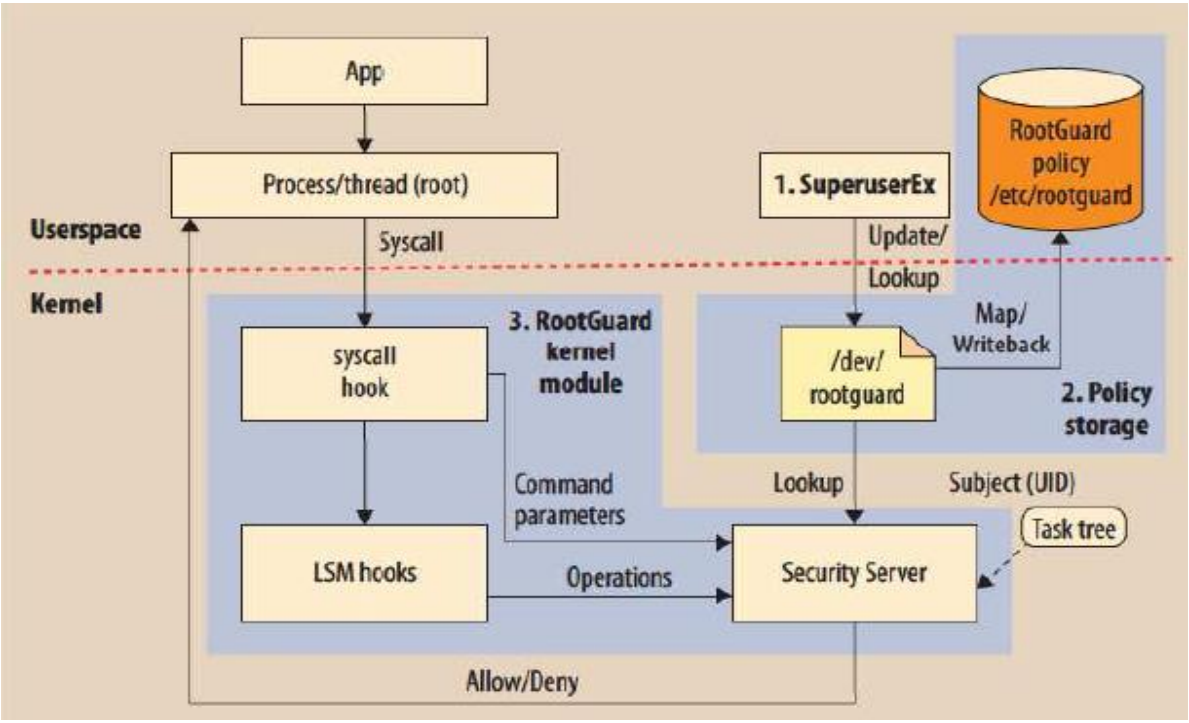
3. Kernel module. RG-ov model identificira operacija od aplikacija koje imaju privilegiju za *root* i odlučuje kada će koja biti izvršena.
Sastoji se od Linux Security Module (LSM) kuke ili kopče, sustava poziva i sigurnosnog servera. LSM dopušta korištenje API zbog pristupa objektima tako što postavlja kopču u kod neposredno prije procesa. Koriste se takve kopče za odlaganje procesa *roota* na aplikaciji, to je prvi upit RG police kroz sigurnosni server. Npr. kada neka aplikacija pokuša montirati sustav dijeljenja u obliku pisanja, sustav poziva naredbu *sys_mount*. Prije ugradnje u operaciju, putanja prolazi kroz LSM kopču, *rg_mount*, čiji upiti prolaze sigurnosni server kako bi se utvrdilo da li je operacija dozvoljena.

Sustav poziva pomoću kopče. Skupljanjem informacija u LSM kopčama ponekad je nedovoljno jer niže stupanjske operacije u objektima možda ne mogu osigurati dovoljno podataka o aplikacijskim

više stupanjskim ponašanjima. Da bi se ispravio ovaj problem, treba instalirati kopču *sys_execve* koja pomaže u operaciji za otkrivanje više parametara. Tradicionalni način za spajanje sustavnih poziva je zamjena odgovarajućih ulaza sa izmijenjenom adresom. Nova verzija sys_execve nije pohranjena u tablici poziva. On se poziva posredno preko ARM-a. Koristi se relativna instrukcija za skok, koja omogućuje RG-u da koristi tehniku spajanja da bi locirao ciljnu lokaciju.

Sigurnosni server. Dizajniran za posebnu provedbu RG politike, pružajući sučelje prethodno opisano, preko kojeg LSM kopče mogu slati upite serveru za RG odluke. Sigurnosni server koristi informaciju sakupljenu u *sys_execve* za identifikaciju aplikacijske operacije, subjekta i dozvolu ili odbijanje za *root*.

Koristi se jedinstveni identifikator (UID) za razlikovanje aplikacijske i neaplikacijske operacije jer svaki Android je prijavljen na jedinstveni UID broj veći od 10.000, dok su UID brojevi manji od toga broja dodijeljeni za sustavnu uporabu. Razlog tomu je što aplikacija izvršava jedinstvenu *root* naredbu svaki put, nakon toga može se utvrditi subjekt koji je praćen u procesu privilegiranja naredbom *task_struct* sve dok proces aplikacije nije pronađen [5].



Sl. 2.4. Tri glavne komponente RootGuarda, [5]

2.4.4. Zadana pravila

RootGuard ima cijeli set različitih pravila da bi zaštitio neiskusne korisnike i obranio ih od napada virusa sa privilegijama *roota*. Analizirani su najpopularnije aplikacije koje zahtijevaju *root* u trgovini Google Play i definirani su 4 glavne grupe [6]:

- Aplikacije za traženje datoteka i njihovo uređenje
- Aplikacije za back up
- Sigurnosne aplikacije sa punim vremenom zaštite i
- Aplikacije za pristup i uređenje postavki hardvera

Kada se instalira aplikacija RG zahtjeva od korisnika da točno specificira jednu od ovih podjela kojoj aplikacija pripada. Aplikacija se zatim provjerava prema pravilima koje ona zahtjeva. Može se pretpostaviti da će legitimna aplikacija koristiti privilegije na *root* samo kako bi ostvarili ono što njihov opis nalaže i ništa drugo. Aplikacija koja zatraži *root* pristup za pregledavanje datoteka neće promijeniti konfiguraciju hardvera. Za pomoć korisnicima za praćenje i izmjenu svojih aplikacija prilikom *roota*, RG snima svaki detalj procesa uključujući: vrijeme, metu, itd. Te ih predstavlja u ‘’SupersuserExu’’. Korisnik može provjeriti sve podatke ako želi. RG pokriva 7 subjekata na svim područjima a to su [6]: podjele, uređaji, sustav datoteka, privatne datoteke, procese, aplikacije i njihove komponente. Ove podjele ovise o zahtjevu aplikacijske grupe, i provodi 3 opcije:

- Dozvoljava operacijama korištenje bez korisnika
- Ne dopušta korištenje bez korisnika
- Pita korisnika za dozvolu

Neki primjeri, fokusirani na pretraživačke alate prikazuju kako ova pravila rade u stvarnosti:

- ***Ugradnja dijelova sustava:*** Sustav je napravljen samo za čitanje
- ***Pristup hardverskim uređajima:*** Uređaji su izloženi u korisničkom prostoru kao datoteke spremljene u direktorij */dev*
- ***Pristup podacima sustava ili drugim privatnim podacima aplikacije:*** Linuxov kontrolni pristup ne može zaštiti aplikaciju sa pravima na *root* od pristupa sustava datoteka i privatnim stvarima od druge aplikacije
- ***Upravljanje procesima memorije:*** Ponašanje procesa može kompletno biti promijenjeno ako je memorija izmijenjena

2.4.5. Prijetnje virusa sa pravima na *root*

Sandbox Android aplikacije ograničava mogućnost korištenja sustava podataka ili drugih izvora. Dok virus stvara probleme u korisničkom dijelu, šteta je uglavnom ograničena jer operacije nemaju dovoljno prava da naprave veću štetu. Virus odobren pravima za *root*, može zaobići sigurnosne mjere Androida, osim onih postojećih alata za *root*. Šest se prijetnja ističe najviše [6]:

Prijetnja 1. Tiha instalacija i deinstalacija. Prije instaliranja neke aplikacije, korisniku su prikazane sve dozvole koje ta aplikacija zahtjeva i može otkazati instalaciju ako su sumnjivi. Nakon stjecanja prava na *root*, virus može instalirati ili deinstalirati aplikaciju direktno pokretanjem *pm install* ili *pm deinstall* naredbe. Prema tome, novi virus može biti instaliran i aplikacija može biti deinstalirana bez znanja korisnika.

Prijetnja 2. Zaustavljanje antivirusnih alata. Po odabiru, aplikacije nemaju privilegije zaustavljati druge aplikacije. Svaki virus sa pravima *roota* može pokrenuti naredbu *kill* da bi zaustavio anti-virus i ostati nevidljiv. Manje više, virus može onesposobiti ključ antivirusne komponente, što je još gore jer će neke antivirusne aplikacije nastaviti raditi bez obzira što im je ključ onesposobljen, zbog toga što je glavna nit netaknuta, korisnici ne znaju da je zaštita isključena.

Prijetnja 3. Neuklonjivost. Sustav aplikacija u */system/app* direktoriju se ne mogu deinstalirati jer sustav nije predviđen za mogućnost *pisanja*. Nakon stjecanja prava na *root*, virus može privremeno montirati svoj sustav kao predviđen za *pisanje*, brisati te aplikacije te se sam ubaciti u direktorij */system/app*. Korisnik ne može ukloniti virus iz direktorija.

Prijetnja 4. Pristup podacima drugih aplikacija. Generalno privatni podaci neke aplikacije ne mogu biti čitani ili izmijenjeni od strane drugih aplikacija, ili je zabranjen pristup i dozvoljen samo onim aplikacijama koje imaju pristup. Npr. za pisanje ili čitanje imenika, aplikacija mora poslati zahtjev *READ_CONTACTS* ili *WRITE_CONTACTS* . Takvo ograničenje se ne odnosi na viruse sa pravima na *root*, što može primijeniti način pristupa bazi kontakata u ‘’globalno čitljivo’’ i ‘’globalno pisano’’ i tako ukrasti podatke od drugih aplikacija. U najtežim slučajevima, virus direktno mijenja način potpisa podataka iz sigurnosnog softvera da može zaobići skeniranje te učiniti sebe nevidljivim.

Prijetnja 5. Stražnja vrata. Virus sa pravima na *root* može kreirati stražnja vrata u sustavu koja dozvoljavaju zaobilaženje normalne ovjere. Kada se zahtijevaju prava za *root*, virus može utjecati na stražnja vrata kako bi dobio prava, bez potrebe SU naredbi.

Prijetnja 6. *Rootkits* i *Bootkits*. *Rootkit (*skup računalnih programa, većinom zlonamjernih*)* je posebno tajan virus dizajniran da bi zaobišao uobičajene metode detekcije i omogućio ovlašteni pristup bez obavijesti sustavu; *Bootkit* je neka vrsta *Rootkit*-a koji prvi preuzima kontrolu nad procesom *dizanja*. Oba modela mogu biti implementirana u Android platformu.

2.4.6. Korisnička iskustva

Pokrenuto je 5 popularnih Android aplikacija iz trgovine Google Play koje zahtijevaju prava *roota,* a to su: Titanium Backup, CPU Tuner, Root Explorer, LBE Privacy Guard i *Root* App Delete u uređaj koji sadrži RG. Rezultati pokazuju da kada korisnik odredi odgovarajuće grupe prilikom instaliranja, aplikacija će raditi normalno; korisnik ne mora dodavati nikakve dodatne konfiguracije. Za mjerenje vremena performansi RG-a, pokrenuto je naširoko poznata aplikacija AnTuTu na dva Google uređaja imena Nexus S, jedan koji ima osnovni instalirani AOSP a drugi RG. Za oba uređaja, potvrđeno je da se isti broj aplikacija učitava i pokreće u bilo kojem trenutku. Tablica 2.2. prikazuje rezultate za 50 pokrenutih puta. RG ima broj pregleda u operacijama *čitanja i pisanja,* gubici su razumni u skladištu ulaza/izlaza baze podataka. RG-ova RAM operacija i RAM brzina je nešto niža nego u AOSP-u, ali su prihvatljivi. CPU integer i CPU floating-point testovi su bili prihvatljivi, rezultati ne utječu previše na operacije RG-a. Multitask i Dalvik rezultati su mjerili korisnikovo iskustvo i vrlo malo utjecali na konačne rezultate, jer većinu promatra RG.

Tablica 2.2. AnTuTu rezultati prilikom mjerenja izvršenja RootGuard-ove izvedbe u odnosu na temeljni operativni sustav Androida. [5]

AnTuTu test case	AOSP		RootGuard	
	Average	Standard deviation	Average	Standard deviation
Multitask	525.05	7.04	519.28	17.21
Dalvik	228.47	2.76	223.33	8.18
CPU integer	346.84	2.36	349.38	1.62
CPU floating point	81.47	0.84	79.47	2.60
RAM operation	269.21	1.51	261.05	11.09
RAM speed	584.84	13.71	574.43	32.87
Storage I/O	765.05	74.44	742.48	70.92
Database I/O	420.84	50.50	399.52	40.71
Total	**3,221.79**	**76.74**	**3,148.95**	**143.51**

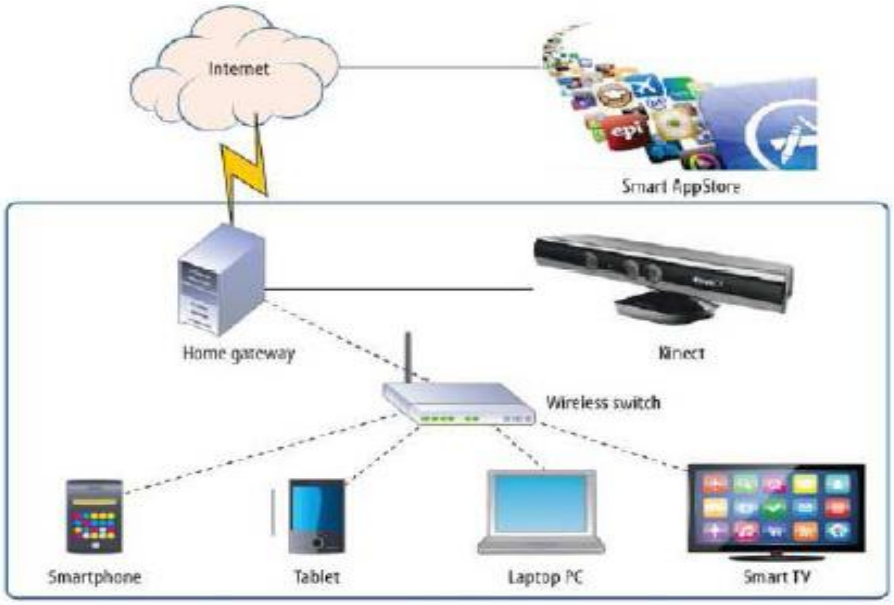
3.PAMETNA TRGOVINA APLIKACIJAMA I BYOD SUSTAV

Trgovine sa aplikacijama nude pet važnih značajki danas za pametne uređaje: biometrijske ovjere, više stupanjske autorizacije, prepoznavanje gesti i navigaciju, upravljanje identitetom i rezultate korisnika. Mobilni eko-sustavi sastoje se od pametnih telefona i tablet uređaja koji su doživjeli rast u posljednjem desetljeću, zbog njihovog jednostavnog korištenja, stalne internetske veze i pristupačne cijene. Nisu više samo uređaji za uspostavljanje poziva i slanja poruka: pojavile su se mnoštvo aplikacija, koje nude razne mogućnosti [7]. Trgovine aplikacijama u početku su se samo nudile mobitelima no s vremenom su postale dostupne i pametnim televizorima. Tradicionalne trgovine aplikacijama napravile su značajan napredak da bi spriječili napade virusa, ali neke prijetnje su se i dalje širile kao što su napadi na korisnikovu ovjeru ili aplikacijska prava na autorizaciju. Jednostavan primjer pokazuje iskustvo korisnika sa korištenjem pametne trgovine aplikacijama. U komforu svoga stana obitelj pristupa trgovini kako bi pronašli aplikaciju i uživali u njoj. Na slici 3.1. prikazan je eko-sustav mobilnih uređaja u kojemu je prikazana web kamera spojena na glavni pristup i u pozadini prepoznaje korisnike u sobi i kreira listu trenutno aktivnih korisnika ili njihove profile. Da bi instalirali željenu aplikaciju moraju se prvo ulogirat na trgovinu na jedan od načina [7]:

1. ***Pratiti tradicionalni proces korisničko ime/lozinka***. U ovom slučaju suočit će se sa zahtjevnim problemom, a to je da će morati tipkati korisničko ime i lozinku putem daljinskog upravljača što je dosta nezgodno. U najboljem slučaju koristit će tipkovnicu uparenu sa pametnim TV-om, ili eventualno tablet uređaj.

2. ***Koristeći biometrijsku ovjeru koju nudi trgovina***. Putem jednostavne geste, mahanja rukom, što web kamera prepoznaje. Svaki od korisnika može imati svoj način prijave.

Po odabiru profila Obitelj grupe, kombinacija glasa i 3D mehanizma za prepoznavanje lica pokrenut je paralelno u pozadini. Čim sustav prepozna svaku osobu sa svojim višerazinskim modelom, sustav dohvaća potrebne atribute profila za konačni proces prijave. Sustav dobiva sve prilagođene reputacijske rezultate dajući na značaju više onima koji dolaze sa drugih profila (korisnika iz grupa). Ako je aplikacija besplatna, instalacija se pokreće automatski. Nakon što je aplikacija kupljena i instalirana, korisnik može ocijeniti aplikaciju i dati preporuku. Trgovina zadržava preporuku i koristi za ažuriranje ugleda aplikacije.



Slika 3.1. Eko-sustav mobilnih uređaja, [7]

3.1.Tipovi prepoznavanja u sustavu

Biometrijska ovjera. Sustav za biometrijsko prepoznavanje prepoznaje više fizičkih osobina koje su jedinstvene i osebujne za svakog pojedinca. Očigledna je prednost u korištenju ovog sustava: nema potrebe za lozinkom, dovoljna je prisutnost osobe u blizini senzora [7]. Ovisno o značajkama, metoda je manje više zahtjevna, tražeći različite razine interakcije. Svaki pojedinac sadrži više fizičkih osobina, tako da svaki pokret mora biti jedinstven, stabilan, jednostavan za ponavljanje i ograničen. Današnji najčešći načini biometrijske metode su: slika irisa, 2D i 3D slike lica, otisak prstiju i govor.

Prepoznavanje glasa. Možemo podijeliti opći problem prepoznavanja govornika u nekoliko kategorija: prepoznavanje govornika protiv ovjere govornika, tekstualno ovisne i neovisne identifikacije, metoda klasifikacije. Trenutni sustavi obično postižu 90-95% točnosti za tekstualno neovisne i 94-98% za ovisne. U realnim uvjetima, govor je sklon varijacijama, a može utjecati i buka iz okruženja. Za prilagođavanje ovih faktora, istraživači koriste razne kompenzacije i normalizacije tehnike kao što je Cepstral srednje oduzimanje, snaga normalizacije, Gaussove metode, metoda savijanja. Povrh svih tih metoda najpopularnija je metoda K-susjedi (KNN), koja pretražuje sve uzorke za obuku i pronalazi najbolje; Gaussova mješavina modela (GMM), metoda koja opisuje

prostor u vjerojatnosni način, i vektor uređaja (SVM), razlikovni pristup koji koristi hyperplane metodu za optimalnu razliku.

Prepoznavanje lica. Postaje sve više popularnija metoda i koristi se u ovjeri, nadzoru i bazi podataka. Trenutna 2D metoda prepoznavanja lica je na 90% uspješnosti u kontroliranim uvjetima, ali ovaj broj značajno opada u realnim uvjetima. Trenutni sustav koristi nekoliko pristupa koje možemo podijeliti u 3 klase: lokalnu, holističku (cijelo lice) i hibridnu metodu (kombinacija ove prethodne dvije). Proces prepoznavanja počinje sa lokaliziranjem lica koji je uobičajeno praćen sa sustavom za eliminiranje nekih grešaka. 3D metoda koristi Z os kako bi dobio potpunu geometrijsku informaciju. Obično, dubina se stječe mjerenjem reflektirane infra-crvene zrake. Ponekad, snop može biti apsorbiran ili se reflektira, stoga algoritam za ‘’punjenje’’ rupa zaglađuje konačni skup.

Prepoznavanje irisa. Iris je jedna od najvažnijih osobina koja ispunjava dva ključna zahtjeva unikatnosti i stabilnosti. Identifikacija prepoznavanje irisa sastoji se od lokaliziranja irisa, ekstrakcije značajki i klasifikacije. Jedan od najuspješnijih sustava koristi filtere Gabor za ekstrakciju značajki, sa filtriranim signalima na dvije razine. Ovaj proces dovodi do mogućnosti binarnih znamenki; prepoznavanje se vrši jednostavnim podudaranjem najbližih uzoraka koristeći KNN metodu i Hammingovu udaljenost [7]. Ovaj pristup postiže 100% uspješnosti u kontroliranoj okolini, preostali rad je na lokaliziranju i scenama iz stvarnog svijeta.

Više-stupanjska autorizacija. Budući da više korisnika može pristupiti pametnoj trgovini aplikacijama s jednog ili više vrsta uređaja, sustav uključuje više-stupanjski modul autorizacije koji provodi adaptivna pravila za kontrolu s različitim razinama sigurnosti za različite korisnike. Više-stupanjska autorizacija se bavi različitim zahtjevima povjerljivosti kroz više biometrijskih modaliteta u kombinaciji sa lozinkama i PIN-ovima. Kad god postoji zahtjev za pristup pametnoj trgovini aplikacijama ili za instalaciju odabrane aplikacije, trgovina koristi glasovna prepoznavanja i prepoznavanja lica za identifikaciju korisnika. Ovisno o ishodu procesa, to bi moglo omogućiti pristup trgovini ili bi moglo zahtijevati korisnika za unos korisničkog imena i lozinke prilikom instalacije besplatne aplikacije ili PIN-a. Trgovina periodično provjerava da li je korisnik prisutan i da li je još netko prisutan u sobi. Ako se neka druga osoba približi ekranu pametnog TV-a, npr. autorizacijski modul će provjeriti i ponovo izračunati pristupnu kontrolu kako bi usporedila da li su osobe povezane.

Prepoznavanje gesti i navigacija. Metoda otiska prsta, koja koristi konveksnost prepoznavanja cijelog dlana. Tražeći koordinate ruke koristeći algoritam instaliran u Kinectu je relativno netočna s obzirom na probleme otkrivanja između dlana kada je otvoren i zatvoren. Razvila se nova metoda za pronalaženje srednje točke dlana. Lokaliziranje dlana pomaže odrediti centar kontura, koje su u konačnici središte dlana, bez obzira u kojem je položaju. To je važno otkrivanje kako bi mogli otkriti lokaciju prsta. Da bi se izračunali konveksnosti, koriste se standardne funkcije. Nalazeći polaznu točku, najdalju i krajnju točku, koja je bitna za lokaliziranje konveksnosti. Kod modificiranja ove metode s pragovima, možemo eliminirati lažne detekcije i dobiti samo prave detekcije sa konveksnostima. Za prepoznavanje gesti, uvedeni su tri seta s gestama, temeljeni na razvijenim izvornim algoritmima. Sva tri seta su integrirani u jedno okruženje, što je dovelo do lakšeg i boljeg prepoznavanja gesti.

3.1.1.Upravljanje identitetom

IdM (engl. IdM- identity management modul) je uglavnom odgovoran za upravljanje korisnicima pametne trgovine [8]. Modul podržava višekorisnički pristup uređajima i mogućnost upravljanja korisnicima, uređajima i servisima. Osobni podaci u postojećim sustavima su ograničeni na jednog pretplatnika, dok IdM sadrži osnovne podatke o svim korisnicima, sigurnosne vjerodajnice, i cijeli kontekst između korisnika i njegovog uređaja. IdM modul upravlja sa nekoliko sadržaja, specijalno u mobilnom okruženju, gdje korisnici žele adaptivno ponašanje, identifikaciju, i postavke temeljene na različitim parametrima za nekoliko konteksta. Iskoristivost prikazanog koncepta može se primijeniti u procjeni parametara koji opisuju ove kontekste, kao što su lokacija, buka u pozadini, itd. Ovi promjenjivi parametri mogu se odabrati od strane korisnika. IdM modul je također koristan za tumačenje korisničke trenutne okoline analizom i usporedbom parametara iz nekoliko uređaja (kao npr. ’’Pronaći sve korisnike na jednoj lokaciji, bez prethodnih veza između njih’’). Konteksti mogu postojati tijekom dugog vremenskog razdoblja, čak i bez aktivnosti korisnika. Pametna trgovina razlikuje definirane kontekste i kontekste temeljene na onom što se upravo dogodilo. Broj korisnika i uređaja nije ograničen [8]. Kontekst se ne sastoji samo od uređaja i korisnika, nego i o njihovoj međusobnoj povezanosti. Mobilnost uvodi scenarije gdje različiti konteksti imaju iste relacije između korisnika i uređaja, sa neprimjetnim razlikama u lokaciji, stvarajući potrebu da se prilagode podaci koji su potrebni IdM modulu za otkrivanje različitih grupa ljudi. IdM modul normalizira parametre multi-modalnih korisničkih sučelja kako bi se utvrdila korisnička stvarna dostupnost i tako definira

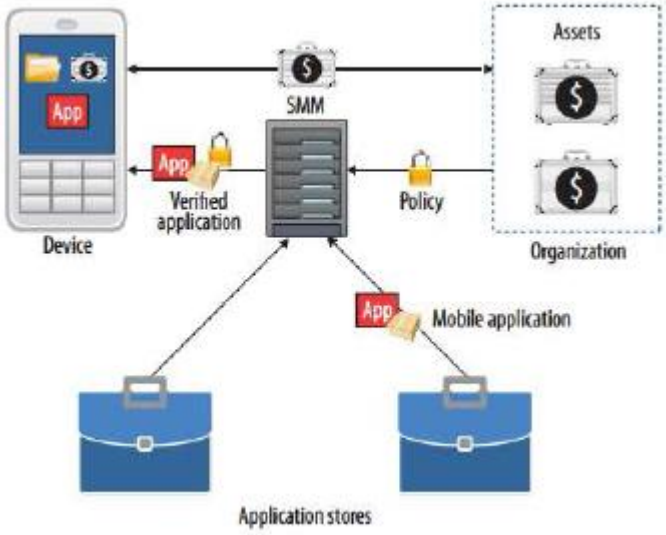
trenutni kontekst. Višerazinska autorizacija treba biti nadomještena za biometrijske metode i uzeti u obzir napredna rješenja kao što su kontrola korištenja za mobilne uređaje.

3.2. Osiguranje BYOD sustava

Sadašnji model distributera mobilnih aplikacija ne može se nositi sa kompleksnim sigurnosnim zahtjevima koji nastaju u BYOD (*engl.bring you own device*) sustavu [9]. Sigurna arhitektura podržava definiciju i provedbu BYOD politike i nudi obećavajuću provedbu implementiranog prototipa pod realnim uvjetima. Dva su faktora doprinijela rasprostranjivanju širenja mobilnih uređaja. Prvi, pojava modernog mobilnog operativnog sustava kao što su iOS, Android i Windows Mobile. Drugi faktor je širok spektar mobilnih aplikacija koji se nalazi u distribuciji koji nudi korisnicima da izaberu aplikaciju po svojoj potrebi. Tijekom instalacije aplikacija ili izvršenja, korisnik obično mora dozvoliti neke zahtjeve, kao što je dopuštanje lokacije prilikom korištenja aplikacije. Ovaj diskrecijski pristup kontrole resursa i usluga pati od brojnih sigurnosnih propusta, ali su ove ranjivosti nekad zanemarene sa raznim prednostima koje nude aplikacije, što im dozvoljava DAC (diskrecijski pristup) politika. U zadnje vrijeme, kompanije su pokazale zainteresiranost i razumijevanje za zaposlenike za korištenje svojih osobnih uređaja na poslu. Taj je trend vodio ka BYOD sustavu u kojem je zaposleniku dopušten pristup osjetljivim izvorima kroz svoje uređaje sve dok koristi i poštuje prava na sigurnost same kompanije. Mnoge aplikacije mogu dovesti kompaniju u opasnost. Komercijalno dostupna rješenja koja nude Apple, Samsung, Symantec, Blackberry i MobileIron među ostalima podržavaju BYOD sustav [9]. Većina ovih izbora pružaju uslugu mobilnog menadžmenta (*engl.MDM*) koji može blokirati ili resetirati uređaj pri kršenju pravila sigurnosti. Pravila su često navedena kroz primjenu aplikacijske crne ili bijele liste, ali bez biheviorističke analize, pravila su temeljena na neodređenom konceptu. Ovdje će se opisati novi pristup osiguranju BYOD sustava na temelju koncepta sigurnosnog metamarketa (e*ngl. Secure metamarket*). SMM posreduje pristupu tradicionalnim aplikacijskim trgovinama, vodi evidenciju o aplikacijama registriranim u uređajima, i glavni zadatak joj je provjeravati funkcionalnost uvažavaju li nove aplikacije sva sigurnosna pravila. U principu, ovaj zadnji zadatak je zapravo i najteži zadatak.

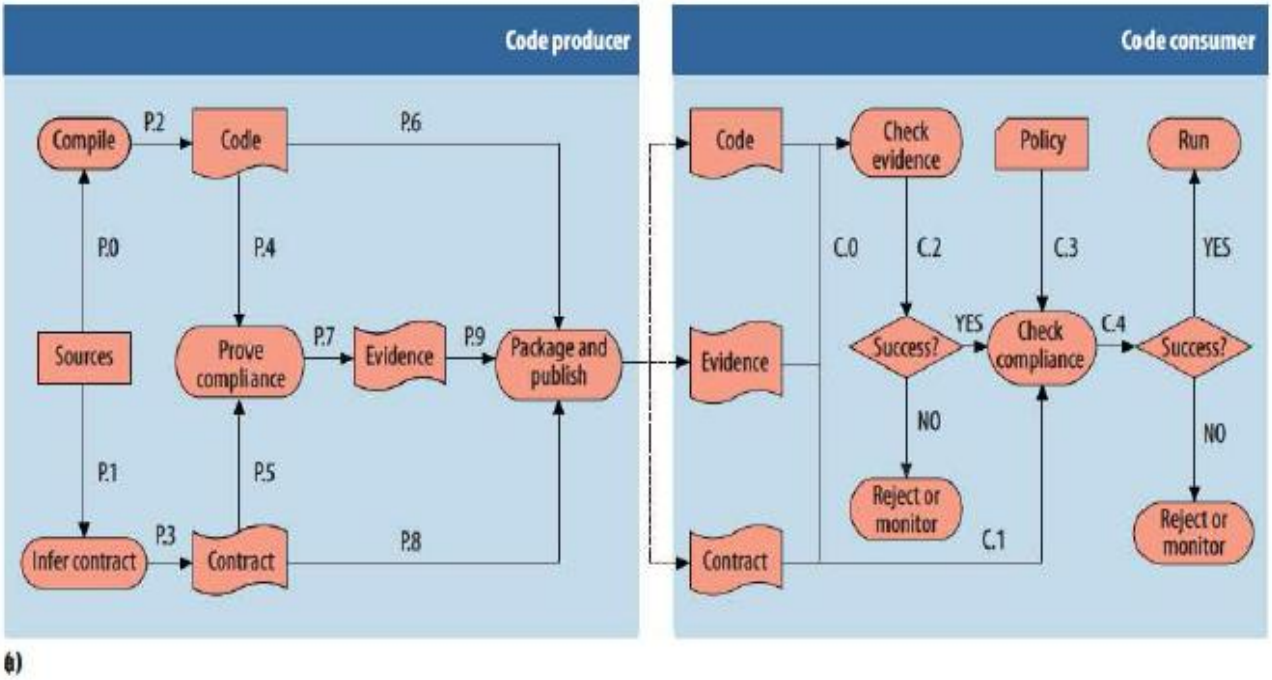
3.2.1. Sigurnost metamarketa

SMM (*engl. SMM- Secure metamarket*) ima za cilj da jamči za skup aplikacija koje su instalirani na mobilne uređaje da su u skladu sa određenim pravilima [9]. Slika 3.2. prikazuje očekivani scenarij. Koristeći već postojane ICT strukture, organizacija želi omogućiti korištenje osobnih uređaja. Definira sigurnosna pravila, ali su osobni uređaji izvan ICT strukture i na taj način se ne može kontrolirati tradicionalno provođenje mehanizma. Uređaj može pokrenuti i ‘’treće aplikacije’’ koji namjerno ili nenamjerno krše pravila. Idealna solucija bi trebala osiguravati da svi članovi organizacije mobilnih uređaja surađuju sa pravilima, izbjegavajući ostavljanje kritičnih odluka korisnicima, i očuvati upotrebljivost uređaja koliko je moguće.



Sl. 3.2. BYOD scenarij sa SMM-om koji provjerava je li aplikacija instalirana na uređaju u skladu sa sigurnosnom politikom, [9*]*

***Sigurnost na temelju ugovora* (*engl.SxC-security by contract*)**. Kao što slika 3.3. prikazuje, nakon stvaranja aplikacije, proizvođač koda generira ugovor (P.3) i pridaje ga softveru (P.8). Proizvođač koda također računa dokaze (P.7) koji ugovor (P.5) ispravno opisuje (P.4) i uključuje u aplikacijski paket (P.9). Prijemnik koda iskorištava dokaze da bi provjerio ispravnost ugovora (C.0) i zatim koristi ugovor (C.1) za provjeru aplikacije protiv vlastite sigurnosne politike (C.3). Ako bilo koji ugovor (C.2) ili politika (C.4) ne uspije na provjeri, aplikacija prima ograničen ili nijedan pristup, npr. preko sigurnosnog monitoringa.



Sl. 3.3. Sigurnost na temelju ugovora, [9]

Ako objje uspiju, aplikacija se može pokrenuti. SxC pruža elegantnu soluciju za sigurnost mobilnog koda, ali pri centralizaciji neke sigurnosne provjere pati od nekoliko problema:

- Provjera ugovora i politike su računski izazovni zadaci koji bi mogli biti previše zahtjevni za resurse ograničenih mobilnih uređaja.
- Proizvođači generiraju ugovore bez poznavanja krajnjeg korisnika i na taj način moraju osigurati velike, sveobuhvatne opise njihovih aplikacija.
- Sxc-ov tijek zahtjeva da mobilni uređaj može provoditi aktivnosti koje bi mogao izgubiti, kao npr. ugovor i verifikacija police. Umjesto toga, korisnici bi trebali odlučiti hoće li preuzeti aplikaciju i instalirati ju nakon što uzmu u obzir njezinu sigurnost.

Zbog svih ovih razloga smatra se da SxC nije direktno primjenjiv za trgovine aplikacijama ili BYOD okolini.

Arhitektura SMM-a. Centraliziranjem sigurnosne provjere na mobilnim uređajima, SMM model izbjegava većinu problema povezanih sa SxC-om: ublažava opterećenje na:

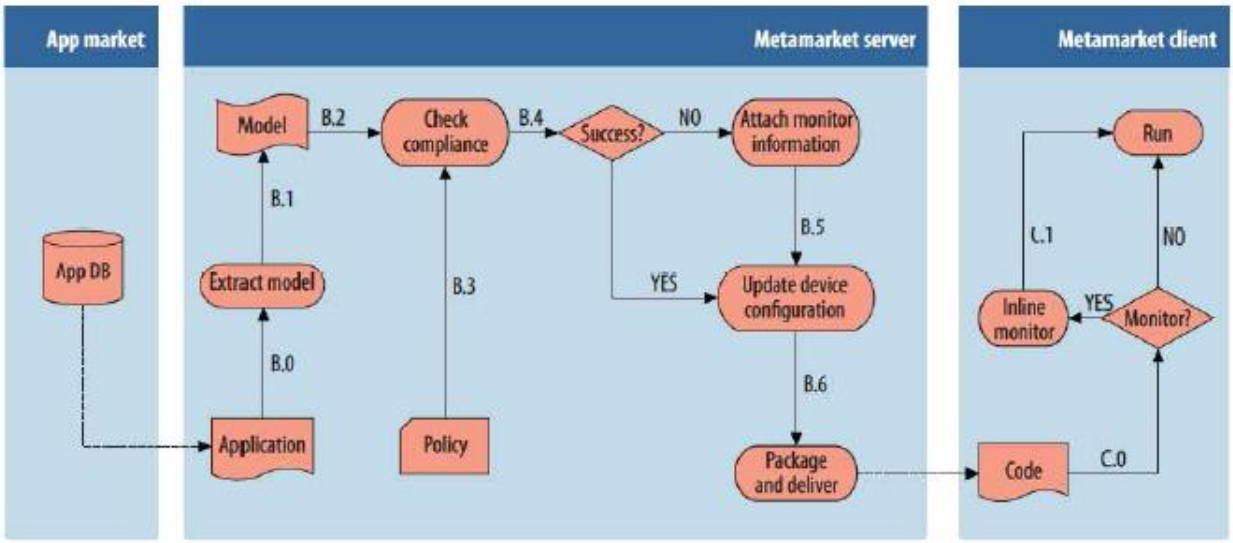
- Mobilnim uređajima sa računski skupim sigurnosnim analizama

- Korisnicima sa sigurnosno- kritičnim odlukama
- I aplikacijskim developerima iz nefunkcionalnih sigurnosno povezanih aktivnosti

Ovaj predloženi model je optimiziran za BYOD scenarij prikazan na slici 3.1. Organizacija raspoređuje ili se preplaćuje SMM-u, koji osigurava da registrirani uređaji pokreću samo usklađene kodove. SMM arhitektura se sastoji od klijenta i servera. SMM klijent se instalira na uređaj prilikom registracije; onemogućuje i zamjenjuje klijente drugih trgovina za sprječavanje nesigurnih instaliranja. Jednom kada je instaliran, SMM klijent prikuplja podatke o sustavu, kao što su instalirane aplikacije i verzije operativnih sustava, i traži od SMM servera provjeru konfiguracije uređaja. Ako je uređaj u skladu sa sigurnosnom politikom organizacije, onda se uspješno pridružuje BYOD mreži. Server SMM-a je odgovoran za održavanje konfiguracije svih registriranih uređaja i posredovanje pristupima trgovine aplikacijama, omogućujući instaliranje samo ako je u skladu sa politikom aplikacije.

Tijek rada SMM-a. Slika 3.4. prikazuje SMM-ov način rada. SMM poslužitelj vraća aplikacijski paket iz javnog tržišta i raspakirava model (B.0), koji opisuje sve sigurnosti aplikacije i ponašanja okoline. Poslužitelj zatim potvrđuje ponašanje modela (B.2) protiv sigurnosne politike organizacije (B.3) preko provjere modela (B.4). Ako provjera uspije, poslužitelj ažurira konfiguracije uređaja i donosi aplikaciju na SMM klijenta za ugradnju (B.6); ako provjera ne uspije, poslužitelj označava aplikaciju za praćenje izvršenja (B.5) prije nego što je isporuči. Kada klijent primi zahtjev paketa, provjerava je li aplikacija označena za praćenje izvršenja (C.0). Ako je to slučaj, klijent obavlja kod instrumenta (C.1)- unosi sigurnosne provjere u aplikacijski kod koristeći informaciju u prilogu od poslužitelja, inače se izravno instalira aplikacija [9]. Poslužitelj raspakirava aplikaciju izgradnjom grafa kontrole toka (*engl. CFG-control flow graph*), u kojem čvorovi predstavljaju programske blokove i rubovi predstavljaju kontrolu toka preko tih blokova. Kako model sadrži sigurnosne relevantne operacije, veličina modela ovisi o interakciji aplikacije sa sustavom i o sigurnosnoj polici. Ta sigurnosna polica mora biti navedena preko odgovarajućeg jezika koji podržava statičke tehnike kao i provjeru izvršenja [9]. Provjera sigurnosne politike uključuje provjeravanje traga ne izvršavanja aplikacije. Općenito, tehnika provjere može vratiti lažne pozitivne i negativne rezultate. Pogrešne negativne štetne aplikacije koje prolaze verifikacijski proces mogu se izbjeći ako su modeli kompletni. U slučaju SMM-a, postupak ekstrakcije kontrolnog grafa *(engl. CFG)* jamči sigurnost procesa. Aplikacije koje padnu na testu sigurnosti moraju se kontrolirati pri izvršavanju. Vrijeme izvršavanja

za praćenje mora biti neinvazivno i ne zahtijevati prilagodbu operativnog sustava. SMM model nudi bitne prednosti, jedna od najbitnijih je da se samo kod instrumenta obavlja na uređaju. Proces provjere u potpunosti izvršava SMM poslužitelj, čime je očuvan sustav jednog klika prilikom instalacije. Stvoren je prototip SMM-u, BYODroid koji podržava operativni sustav Android i BYOD sigurnosnu politiku. Pretpostavlja se da će klijent imati Android uređaj i da će trgovina aplikacijama biti kompatibilna sa Androidom, kao što je Google Play. Kada je bilo moguće korištena je tehnologija za komponente za implementaciju radnih procesa prikazanih na slici 3.4.



Sl. 3.4. Način rada SMM-a, [9]

3.2.2. Provjera modela i instrumentacija koda

Da bismo provjerili usklađenost s pravilima, BYODroid koristi Spin, model ispitivač koji je uspješno primijenjen u brojnim stvarnim problemima. BYODroid provodi i aplikaciju CFG [10]. U tu svrhu, prilagođen je postojećim tehnikama koji se primjenjuju na CFG-ovu specifikaciju za kodiranje konačnih čimbenika. Za sprječavanje neograničenih proračuna koji su neprihvatljiv tijek u SMM-u, model za provjeru sadrži *timeout* mehanizam. Model za provjeru sadrži tri moguća rezultata:

1) DA-model je u skladu sa pravilima
2) NE-model krši sigurnosnu politiku
3) TIMEOUT(TO)-vrijeme praga je postignuto

BYODroid-ov model provjere ne dopušta da se dogodi NE događaj.

Instrumentacija koda. Praćenje izvršenja omogućeno je putem koda instrumentacije. BYODroid klijent obavlja takve instrumentacije izravno na uređaju, unošenjem sigurnosne provjere u kodu aplikacije, model provjerava rezultat u oba slučaja (NE ili TIMEOUT) [10]. Sigurnosne provjere zamotavaju svaku aplikacijsku policu relevantne instrukcije. Ako izvršenje prolazi kroz sigurnosnu provjeru, postupak se kontrolira. Ako se krše pravila, sigurnosna provjera pravi iznimku. Na taj način sve sigurnosne police ili pravila su provediva: instrumentirana aplikacija nikada mu ne može naštetiti. BYODroid se oslanja na Redexer za instrumentacijsku proceduru koji uključuje nekoliko značajki za upravljanje Android aplikacijama. Ovaj pristup poboljšava sigurnosne mehanizme kod Androida. Nakon instalacije, prvi zadatak BYODroid klijenta je da provjeri konfiguraciju uređaja: klijent skuplja podatke o svim instaliranim aplikacijama i zahtjevima servera da provjeri da li uređaj može sigurno pristupiti BYOD mreži. Da bi uređaj bio primljen, vlasnik mora ukloniti ili ažurirati neke od aplikacija. Klijent pruža korisniku sigurnosne pojedinosti u vezi mogućeg instaliranja aplikacije. Također obavlja uobičajene operacije klijenta u vezi trgovine, kao što su kupovine i povratni procesi. Kada korisnik odabere neku aplikaciju za instaliranje, klijent aktivira zahtjev za softverski paket i shodno tome SMM tijek rada. Poslužitelj vraća paket kojemu je ovisno o ishodu postupka potreban kod instrumentacije. Ako je tako, poslužitelj pruža klijentu originalnu aplikaciju. Važno je napomenuti da kod instrumentacije poništava originalan aplikacijski potpis. No, ipak potreban je ispravan potpis za instaliranje aplikacije na Android uređaj. Kada je kod instrumentacije poznat BYODroid poslužitelju, izračunava se novi potpis koristeći *proxy ključ (*privatni ključ) generiran od strane poslužitelja za svaku aplikaciju [10]. Kao dio instrumentacijske podrške, poslužitelj šalje novi potpis BYODroid klijentu, koji provjerava originalni potpis za provjeru integriteta koda. Konačno, klijent instrumentira kod, daje ga poslužitelju pod uvjetom da ga da novoj aplikaciji i instalira je.

4. SIGURNOST APLIKACIJA U iOS OPERATIVNOM SUSTAVU

Aplikacije su među kritičnim elementima moderne sigurnosne arhitekture pametnih telefona. Dok aplikacije pružaju nevjerojatne prednosti za korisnike, one također imaju potencijal da negativno utječu na sigurnost cijelog sustava, za stabilnost korisničkih podataka ako se s njima ne rukuje ispravno. Zbog toga iOS pruža slojeve zaštite kako bi se osigurali pravi potpisi aplikacija. Ovi elementi pružaju stabilnu, sigurnu platformu, omogućujući programerima lakši pristup za isporuku mnoštva aplikacija bez utjecaja na sustav. Korisnici također mogu pristupati aplikacijama bez straha od virusa, zlonamjernih programa ili napada [11].

Potpisivanje aplikacijskog koda

Nakon što se aktivira iOS kernel, kontrolira se koje se aplikacije i procesi mogu pokrenuti. Kako bi se osiguralo da sve aplikacije dolaze iz poznatih izvora i nisu bile mijenjane, iOS zahtjeva da se svi izvršeni kodovi potpisuju pomoću Apple-ovih potvrda. Treće aplikacije također moraju biti ovjerene i potpisane sa istim certifikatom koji propisuje Apple. Kako bi zaštitili sustav i druge aplikacije od unosa koda unutar njihovog prostora, sustav će obaviti potpis koda svih dinamičkih biblioteka. Ovaj se proces postiže pomoću identifikatora *Team ID*, koji je izvađen iz Apple-ovih certifikata. To je 10-znamenkasti alfanumerički niz, kao npr. 1A2B3C4D5F. Program se može povezati sa bibliotekom koja se isporučuje sa sustavom [11]. Za razliku od drugih mobilnih platformi, iOS ne dopušta korisnicima instaliranje potencijalno zlonamjerne nepotpisane aplikacije sa web stranica, ili nepouzdanog koda. Za vrijeme izvođenja, potpis koda provjerava se je li nešto izmijenjeno u kodu aplikacije od vremena instaliranja do pokretanja.

Sigurnost za vrijeme izvršavanja aplikacije

Nakon što je aplikacija provjerena, iOS provodi sigurnosne mjere kako bi se spriječilo ugrožavanje drugih aplikacija ili sustava. Sve *treće* aplikacije su ograđene, tako da je zabranjen pristup drugim podacima pohranjenim u uređaju ili aplikaciji. Time se sprječava izmjena podataka u njima. Svaka aplikacija ima jedinstveni polazni direktorij za svoje datoteke, on je nasumično dodijeljen prilikom instalacije aplikacije. Ako neka od *trećih* aplikacija zatreba pristup određenim informacijama, osim vlastitih, to čini samo pomoću izričito predviđenih alata od strane operativnog sustava. Cijela particija operativnog sustava je napravljena samo za *čitanje*. Nepotrebni alati, kao što je daljinska prijava, nisu

uključeni u softver sustava i API-ji ne dopuštaju aplikaciji razvoj svojih privilegija za izmjenu druge ili sustava.

Proširenja

Operativni sustav aplikacijama omogućuje pružanje funkcionalnosti drugim aplikacijama preko proširenja (ekstenzija). Sustav automatski prepoznaje proširenja tijekom procesa instalacije te ih čini dostupnima drugim aplikacijama [14]. Takav prostor koji dopušta sustav se zove točka proširenja (engl. Extension point). Svaka ta točka pruža API i provodi politiku za taj prostor. Sustav određuje koja proširenja su dostupna na temelju određenih pravila koja se moraju podudarati. Proširenja se automatski pokreću po potrebi procesa i upravljaju njihovim daljnjim tijekom. Izvode se u vlastitom adresnom prostoru. Komunikacije među proširenjima i aplikacijama koje su dopustile ta proširenja bivaju upravljane od okvira sustava. Oni nemaju pristup jedni drugima i njihovim datotekama ili memoriji, proširenja su dizajnirana tako da budu izolirane od drugih. Ograđene su kao i *treće* aplikacije i imaju spremnik odvojen od spremnika aplikacije koja ih koristi. Međutim, oni dijele isti pristup kontrole privatnosti. Tako da ako korisnik da pristup aplikaciji kontaktima, ovaj pristup će se proširiti i na proširenja koja su ugrađena unutar aplikacije. Prilagođene tipkovnice su posebne vrste proširenja jer su omogućene od strane korisnika za cijeli sustav. Kada je omogućeno, proširenje će biti korišteno za bilo koje tekstno polje osim teksta za unos lozinke ili bilo kojeg sigurnosnog teksta. Zbog privatnosti, prilagođene tipkovnice automatski se pokreću u vrlo restriktivnom prostoru koji blokira pristup mreži. Developeri tih tipkovnica mogu zahtijevati da njihova proširenja imaju otvoreni pristup, koji će dopustiti sustavu pokretanje proširenja u određenom prostoru nakon dobivanja suglasnosti sa korisnikom.

Aplikacijske grupe

Aplikacije i proširenja u vlasništvu određenog developera mogu dijeliti svoj sadržaj kada je konfiguriran u dio aplikacijske grupe [14]. To je do developera kada izradi odgovarajuću skupinu na Apple-ovom portalu i uključi željeni set aplikacija i proširenja. Jednom kada je konfiguriran da bude dio grupe, aplikacije imaju pristup sljedećem sadržaju:

- Zajedničko skladište on-diska za dijeljenje, koje ostaje na uređaju sve dok je najmanje jedna aplikacija iz grupe instalirana
- Zajedničke preference

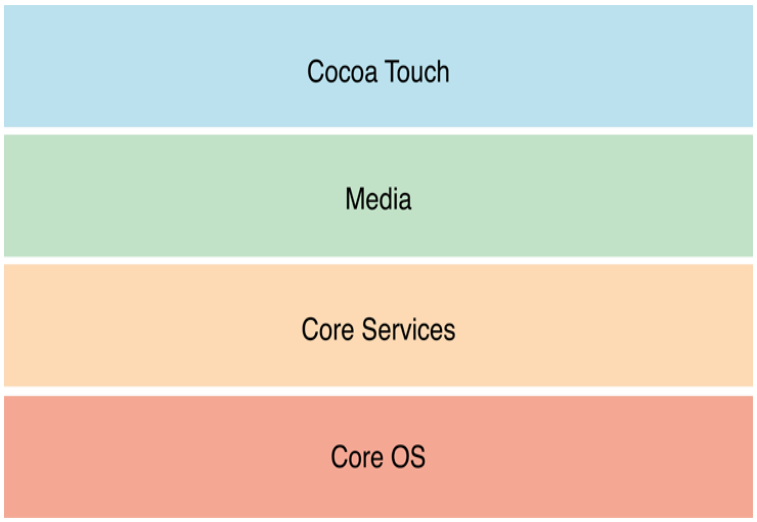
- Zajedničke ključne stavke

Zaštita podataka u aplikacijama

SDK (*engl.Software Development Kit*) nudi cijeli paket API-ja koji olakšavaju developerima trećih aplikacija zaštitu podataka i osiguranje najviše razine u svojim aplikacijama. Zaštita podataka je dostupna za datoteke i bazu podataka API-ja, uključujući NSFileManager, CoreData, NSData i SQLite.

4.1. Arhitektura operacijskog sustava iOS

Operacijski sustav kao što je ranije navedeno posreduje između cijeloga sklopa mobilnog uređaja i same aplikacije koju u sebi nosi. Komunikacija se odvija u različitim slojevima i prema strogo definiranim pravilima. U noviju arhitekturu samog operacijskog sustava dodani su slojevi za implementaciju aplikacija na platformu [11]. Na slici 4.1. je prikazana arhitektura operacijskog sustava te je opisan svaki sloj zasebno.



Sl.4.1. Prikaz arhitekture iOS [11]

Core OS: Najniža razina operacijskog sustava oko koje je izgrađena većina novih tehnologija koja se koriste kroz razne programske okvire (*engl. framework*). U situacijama gdje je potrebno se baviti

sigurnošću ili sa komunikacijom sa vanjskim uređajima ili hardverom programeri se služe sa ovim slojem. Core OS se sastoji od:

- Okvir za ubrzanje- sadrži sučelje za digitalnu obradu signala (DSP), linearnu algebru i stvarne izračune stvaranja slika. Prednost korištenja ovog okvira je ta da je optimizirana za sve hardverske konfiguracije koje su prisutne u iOS uređajima, tj. jednom kada se napiše kod upotrebljava se na svim uređajima.
- Temeljni Bluetooth okvir- omogućava programerima interakciju na niskoj energiji. Omogućava: traženje Bluetooth pribora za spajanje i odspajanje za aplikacije tako što pretvara iOS uređaj u periferni u odnosu na Bluetooth, prijenos podataka, očuvanje veze i obnavljanje iste
- Okvir za komunikaciju- Pruža podršku za komunikaciju s hardverskim priborom u iOS uređaju. Pribor se može spojiti preko konektora s 30 pinova ili bežičnim putem.
- Okvir za sigurnosne servise- pruža standardni set sigurnosnih usluga za iOS aplikacije. Osim standardnih sučelja nude se još dodaci za upravljanje koji su potrebni pojedinim aplikacijama.
- Sigurnosni okvir- Uz ugrađene sigurnosne značajke, iOS također pruža eksplicitan sigurnosni okvir koji se može koristiti kako bi se osigurala sigurnost podataka aplikacija kojim se upravlja. Ovaj okvir pruža sučelja za upravljanje certifikatima, javnim i privatnim ključevima, i policama. Podržava stvaranje kriptografskih pseudoslučajnih brojeva, spremanje istih te raznih certifikata. Zajednička knjižnica pruža dodatnu podršku za simetrično šifriranje, za provjeru autentičnosti kodova poruke.

Core services: Ovaj sloj sadrži osnovne usluge sustava za aplikacije. Sadrži pojedine tehnologije za potporu značajki kao što su: lokacija, iCloud, društveni medij i umrežavanje. Sastoji se od: **Funkcionalnosti viših razina**:

- Peer-to-Peer usluge- Ovaj okvir pruža peer-to-peer povezivanje putem Bluetootha. Može se koristiti za iniciranje komunikacije sjednice s uređajima u blizini. Iako se koristi većinom samo u igricama, može poslužiti i u drugim vrstama aplikacija.
- iCloud pohrana- Omogućuje aplikaciji pisanje korisničkih dokumenata i podataka za središnju lokaciju. Korisnik im može pristupiti sa bilo kojeg računala i iOS uređaja.

Sigurnost prilikom pohranjivanja određenih podataka u korisnikovom iCloudu pruža sloj sigurnosti za korisnike. Čak i ako korisnik izgubi uređaj, dokumenti na tom uređaju neće biti izgubljeni ako se nalaze u iCloud pohrani.

- Blok objekti- programski sadržaj koji se koristi u programskom kodu. Funkcije koje se najčešće koriste za opozivanje određenih funkcija.
- Zaštita podataka- omogućuje aplikaciji rad sa osjetljivim podacima. Kada aplikacija označava određenu datoteku kao zaštićenu, sustav pohranjuje te datoteke na disku u šifiranom obliku. Dok je uređaj zaključan od strane korisnika, ključ je stvoren kako bi se omogućilo aplikaciji pristup datoteci.
- *Grand Central Dispatch* (GCD)- tehnologija koja služi za upravljanje izvođenjem određenih radnji neke aplikacije. Tehnologija koja koristi oblik programiranja u kojem se u jednom trenutku odvija više proračuna.
- *In-App Purchase*- tehnologija za obradu financijskih transakcija

Sloj sadrži XML podršku i SQLite knjižnicu koja služi kao ugradnja baze podataka u aplikaciju.

Programska okruženja:

- Okvir računa- pruža pojedinačne sign-on modele za određene korisničke račune
- Knjiga adresa- programski pristup za upravljanje kontaktima
- CFNetwork- skup raznih sučelja programiranim za rad s mrežnim protokolima
- Core Data- tehnologija za upravljanje podacima aplikacija
- Core Location- tehnologija za određivanja lokacije uz pomoć GPS-a

Sloj sadrži razna sučelja kao što su za događaje u kalendaru, za upravljanje podacima, za kupovinu u trgovinama, upravljanje multimedijom, upravljanje pregledom i sl.

Media: Sloj koji sadrži grafiku, audio i video tehnologiju koja se koristi za provedbu multimedijskog sadržaja u svim aplikacijama. Ovaj sloj olakšava izradu aplikacije koja izgleda i zvuči odlično. Upravlja grafikom u situacijama kada je određenoj aplikaciji potrebna napredna tehnologija za stvaranje bogatijeg grafičkog sadržaja. Koristi 2D i 3D obrađivanje slike, upravlja prikazom teksta, formatom slike i sl. Pomaže u stvaranje boljeg zvuka, daje nekoliko načina za reprodukciju i snimanje audio sadržaja kao i za video sadržaj (Media Player, AV Foundation i Core Media) [14].

Cocoa Touch: Sloj sadrži ključna programska rješenja za izradu aplikacije iOS. U ovom se sloju definiraju osnovne infrastrukture aplikacije i definira se *multitasking, push* obavijesti, itd.

Funkcionalnosti viših razina:

- *Multitasking*- omogućuje aplikacijama više radnji istovremeno u pozadini
- Push obavijesti- omogućuje interaktivne obavijesti
- Data transfer- dijeljenje datoteka među korisnicima kroz aplikaciju iTunes
- Data protection- zaštita datoteka pomoću kriptiranja
- Sistemski kontroleri korisničkog sučelja- skup za stvaranje korisničkih sučelja pojedine aplikacije

Programska okruženja:

- Address Book UI- korisničko sučelje za upravljanje kontaktima
- Event Kit UI- upravljanje kalendarom
- Game Kit- upravljanje mrežnim igrama
- Map Kit- upravljanje geografskim kartama
- Message UI- korisničko sučelje za upravljanje SMS porukama
- UIKit- upravljanje aplikacijom, korisničkim sučeljem, itd.

Sloj sadrži još razne okvire kao što su PushKit Framework (podrška za registraciju VoIp aplikacija), Twitter okvir, itd.

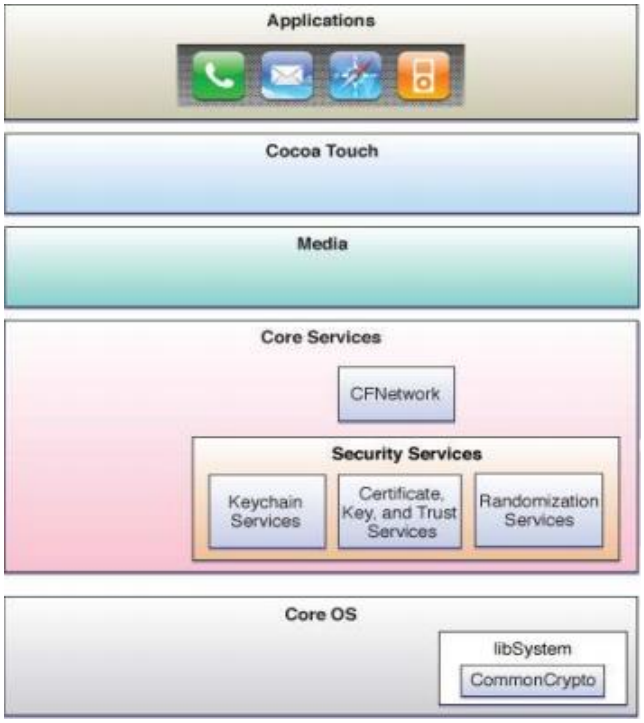
4.2. Sigurnost u operacijskom sustavu

Svaka mobilna platforma u posljednih nekoliko godina ima problema sa sigurnosti [16]. Rastom popularnosti raznih aplikacija, pametni telefoni, tablet uređaji i ostali pametni uređaji sa pristupom internetu i aplikacijama postaju meta i napadači upravo izvršavaju napade na njih. Vrsta napada ovisi o korisniku te o samoj mobilnoj platformi koju korisnik ima instaliranu na svome uređaju. Samu sigurnost koju nude te platforme nisu dovoljne te je korisnik često potaknut da se sam brine o svome uređaju, što nekad i nije pametno, jer to pokušava na pogrešne načine. U tekstu koji

slijedi biti će otkriveni sami propusti ovoga sustava, najčešći napadi na njega te kako iste napade spriječiti.

4.2.1. Struktura sigurnosti operativnog sustava

Upravljanje sigurnošću u iOS operacijskom sustavu nalaze se u sučeljima u drugom sloju Core Services. Na slici se jasno vidi arhitektura operacijskog sustava te sučelje koje se temelji na servisima prvog sloja Core OS. Svaka aplikacija sučelja koristi direktno kroz ta dva sloja [16].



Sl. 4.2. Prikaz sigurnosnih sučelja u slojevima [16]

Zbog silnih napada koji su sve češći, iOS uvodi proces u pozadini pod nazivom Security Server, što je vidljivo na slici 4.2. U sebi nosi podršku za nekoliko protokola. Aplikacije koriste sigurnosna sučelja Keychain Services, Certificate, Key and Trust Services, Randomization Services, svaki od njih će biti opisan.

- Keychain Services- aplikacije ga koriste za spremanje lozinki, sigurnosne ključeve i sve tajne podatke. Aplikacije koriste kriptografiju kako bi se podaci držali pod šifrom. Ovo sučelje koristi prostor za pohranu gdje se spremaju razni algoritmi korišteni u kriptografiji.
- Certificate, Key and Trust Services- ovo sučelje sadrži funkcije za upravljanje certifikatima, ključevima, šiframa i sl.
- Randomization Services- sučelje se koristi za stvaranje pseudoslučajnih brojeva.

4.2.2. Zlonamjerne aplikacije

Problemi ovog operacijskog sustava su slični kao i na drugim sustavima. Među njima su i zlonamjerne aplikacije koje štete krađom podataka i sl. Takve aplikacije zadaju teške probleme sigurnosti cijeloga sustava. Korisnike ovog sustava ne bi trebao zamarati taj problem, jer su stručnjaci vrlo ozbiljno prišli tom problemu i svakim danom ga sve bolje rješavaju. Glavni problem u ovim aplikacijama je što bez ovlaštenog pristupa upadaju u pametne uređaje. Operativni sustav se rješava takvih aplikacija uglavnom već na početku, tj. ne daje im pristup trgovini te korisnici ne mogu doći tako lako do njih. Ove se aplikacije najčešće transferiraju s uređaja na uređaj preko Bluetootha, mail-a, SMS porukom i sl. App Store ili trgovina iOS sustava ograničava širenje takvih aplikacija. Svaka aplikacija prolazi kroz strogu provjeru te se sa svojim digitalnim potpisom povezuju sa svojim autorom [16].

Jailbreak

Otključavanje (engl. *jailbreaking*) je proces u kojem se na svim uređajima sa iOS platformom pokušavaju ukloniti ograničenja koja je definirala tvrtka Apple. Slično kao i kod Androida, iste probleme imaju korisnici kada pokušavaju napraviti neovlašteni *root* proces [16]. Otključavanje omogućava korisnicima potpun pristup cijelom operacijskom sustavu. Uz sva ta prava na pristup operacijskom sustavu dolazi i korištenje nevaljanih odnosno nepotpisanih aplikacija i svih dodataka koji dolaze uz njih. Svako otključavanje se može i poništiti kroz procese korištenjem iTunes-a. Nakon što se instalira nepotpisana aplikacija na uređaj, ona dobija sva moguća prava na vaš uređaj. Takve aplikacije sa sobom nose sigurnosne rizike, pojavili su se razni virusi preko kojih je moguće doći do svih podataka o korisniku (ime, prezime, bankovni računi, lozinke, itd.). Također zlonamjerne aplikacije dolaze i kroz PDF format, kroz koji si omogućuju pristup identitetu korisnika.

Podaci u sustavu i razmjena podataka

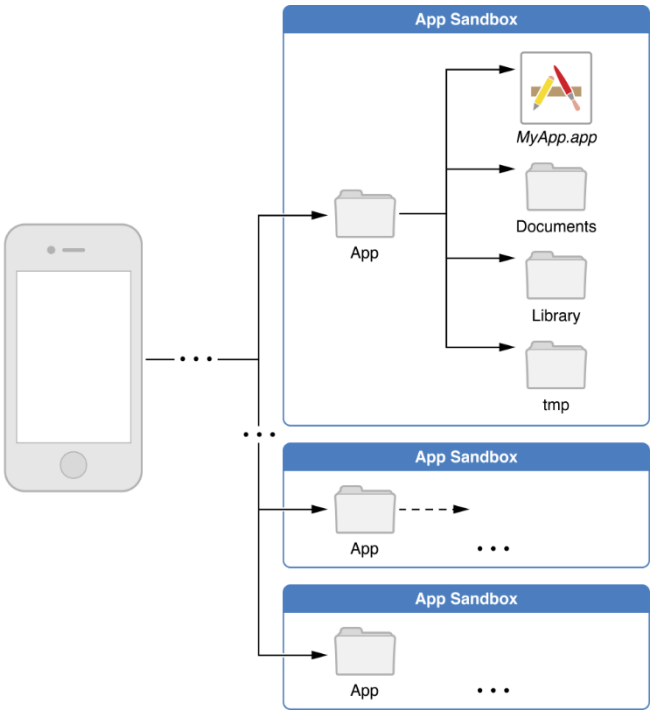
Važno je u svakom trenutku znati kakvi se podaci nalaze na uređaju i u kojemu su obliku. Velika većina podataka kao što su: SMS-ovi, mailovi, lista kontakata, razni dokumenti, lozinke, GPS podaci se nalaze u *Property List* koji su čitljivi u SQLite bazi podataka. Da bi se došlo do podataka koji su bili izbrisani postoje razni alati, kao što su: Lantern, Device Seizure, itd. Ti nam alati služe za prikaz tih dokumenata nakon njihovog brisanja sa uređaja. Veliki problemi nastaju u razmjeni podataka, pogotovo preko mreže. Napadači presreću razne poruke te ih čitaju, tako što dolaze do kriptirane poruke koju zatim lako dekriptira te ih lako čita svojim novim ključem. Ranjivost se također nalazi nakon umrežavanja, gdje korisnik veoma često mijenja mjesto spajanja na lokalnu mrežu koja nije zaštićena. Većina korisnika još ne zna pravu mjeru zaštite te tako postaju lake mete napadačima i upravo zbog toga ostaju bez osjetljivih podataka. U idućem poglavlju biti će opisani neki od sigurnosnih mehanizama i metode zaštite.

4.3. Sigurnosni mehanizmi

Programeri aplikacija a i sami korisnici mogu prilikom korištenja uređaja ili programiranja koda koristiti neki od sigurnosnih mehanizama koji nudi operativni sustav iOS [12].

4.3.1. Sandboxing

Sandboxing aplikacije je odličan način zaštite sustava i korisnika ograničavajući privilegije aplikaciji i njezinoj funkcionalnosti. Od početka iOS platforma je osmišljena tako da bude što zaštićenija i sigurnija. Zato je Apple stvorio stroga ograničenja za aplikacije koje rade u okruženju iOS-a; svaka aplikacija mora biti potpisana i ovjerena od strane Apple-a i svaki zahtjev mora biti sandboxan [12]. To znači da svaka aplikacija ima svoj sigurnosni okvir ili okoliš u kojemu se mogu izvršavati radnje čitanja i pisanja. Nijedna aplikacija ne može preuzeti podatke od druge niti direktno pisati u nju.



Sl. 4.3. Prikaz sandboxinga aplikacije [12]

Prije verzije 8 operativnog sustava dvije aplikacije su mogle razmjenjivati podatke samo pomoću pozivanja funkcije "Open In". Svaki put kada odaberete dokument i odlučite ga otvoriti u drugoj aplikaciji, sustav stvara novu kopiju dokumenta i prenosi ga u sigurnosni okvir aplikacije. Da bi se napravile promjene u njoj potrebno je pozvati funkciju "Open In" kako bi se vratili promijenjeni podaci u izvornom obliku. Zatim se dobiju dva dokumenta; jedan bez promjene, a drugi s promjenama. Novi način preko kojeg aplikacije komuniciraju je već raniji spomenuti proces proširenja (engl. extensions). To su proširenja za glavne aplikacije, proširuju programsku funkciju i omogućuju posrednu komunikaciju s drugim aplikacijama, npr. proširenja omogućuju da odaberete fotografiju iz memorije fotografija i da ju dijelite na omiljenu društvenu mrežu ili spremite u iCloud [12].

Zaštita podataka: Ako uređaj podržava iOS platformu, politika bi se trebala usmjeriti na korištenje lozinke kako bi se zaštitili osjetljivi podaci [16]. Treba imati na umu da nisu sve lozinke jednake. Na iOS uređajima, lozinke dolaze u dvije vrste: jednostavan, 4 znamenke, numerički pristupni kod i složeniji alfanumerički kod, koji je uobičajeno nešto duži. Na primjer, prema podacima Apple-a, napad na uređaj koji koristi 9-znamenkasti numeričku lozinku će trajati 2,5 godine dok se isprobaju sve kombinacije. Zbog toga iOS provodi vrijeme kašnjenja kako bi se obeshrabrili ti napadi. S druge

strane, 6-znamenkasti kod u kojemu su pomiješani brojevi i slova biti će na udaru 5,5 godina. Zbog toga ne bi trebali koristiti uopće lozinke od 4 znamenke. Čak i ako napadač izvrši *jailbreak* na uređaj i zaobiđe lozinku, podaci će mu i dalje biti nedostupni jer ne zna lozinku. Ipak nije samo dovoljno koristiti lozinku u slučaju napada. Aplikacija mora biti dizajnirana za korištenje aplikacijskog programa za zaštitu podataka sučelja kako bi se osigurali podaci. Također treba osigurati da se podaci ne mogu premjestiti na aplikacije koje ne koriste zaštitu podataka. No treba biti oprezan, jer zaštita podataka nije moguća u datotekama koje sudjeluju u iCloud pohrani. Od izdavanja verzije 6 iOS-a, korisnici imaju više kontrole nad aplikacijama kako bi mogli pristupiti sigurnim podacima drugih aplikacija. Iako se mehanizam šifriranja nije mijenjao u ovoj verziji, operativni sustav sada traži dopuštenje korisnika prije nego novoinstalirane aplikacije pristupe osobnim podacima (kalendarima, kontaktima, podsjetnicima i sl.). Korisnici sada također mogu mijenjati pristup podacima za određenu primjenu u postavkama uređaja (odabirom Privacy). Treba uzeti u obzir da puno održavanje kontrole nad aplikacijama pridonosi osiguranju sustava.

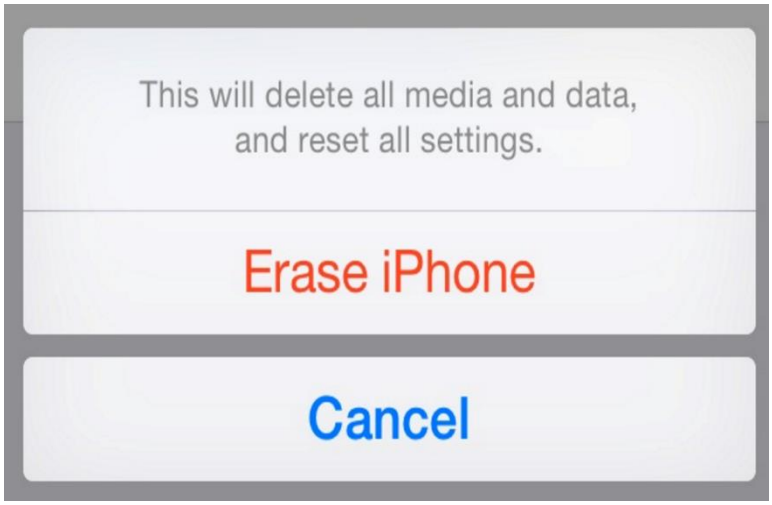
4.3.2. Upravljanje digitalnim pravima

Često izdavači ograničavaju upotrebu digitalnog sadržaja uvođenjem DRM (*engl. DRM-Digital Rights Management*) procesa, koji je oblik nadzora pristupa tom digitalnom sadržaju [16]. Takva tehnologija u Apple se naziva FairPlay koji se koristi u većini njegovih uređaja. U procesu se obavlja proces šifriranja audio datoteka koristeći AES (engl. *Advanced Encryption Standard*) algoritam u kombinaciji sa MD5 sažetkom. Tim procesom se sprječava korisnika da pogleda video sadržaj na neovlaštenom računalu. Ključ potreban za dekripciju samog sadržaja spremljen je u datoteci ali u šifriranom obliku tako da je teško doći do njega. Dobiva se na autoriziranom računalu pomoću aplikacije iTunes. Svaki put kada novi korisnik želi koristiti iTunes da bi kupio novu pjesmu, novi slučajni korisnički ključ se generira i koristi za šifriranje glavnog kluča. Taj ključ je zatim pohranjen, zajedno s podacima na računu. iTunes pohranjuje ključeve u vlastita spremišta, koristeći ta spremišta može dohvatiti ključ potreban korisniku za dešifriranje glavnog ključa [16]. Kada korisnik vrši autorizaciju novog računala, iTunes šalje jedinstveni identifikator Apple-ovom serveru. Za uzvrat dobija sve korisničke ključeve koji su pohranjeni s podacima o računu. Takav pristup osigurava Apple-u mogućnost ograničavanja broja računala koja su ovlaštena i brine da svako ovlašteno računalo ima sve korisničke ključeve potrebne za reprodukciju nekog audio/video sadržaja. Obrnut

proces je kada korisnik se pokušava deautorizirati sa računala. FairPlay proces ne utječe na sposobnost datoteke da se kopira, proces samo upravlja dešifriranjem audio/video sadržaja.

Sigurnosno brisanje podataka

U pokušaju nedozvoljenog probijanja u sustav od strane napadača ili lopova u slučaju krađe uređaja sustav nudi mogućnost sigurnosnog brisanja podataka sa uređaja [16]. Najčešće je sustav postavljen na tri pokušaja probijanja lozinke, nakon toga se aktivira ta mogućnost. Moguće je izvršiti i udaljeno brisanje sa drugog uređaja kroz sustav iCloud. Preko sustava je moguće vidjeti geografski položaj, moguće je čak i izmijeniti postojeću lozinku na uređaju. Proces ima neke nedostatke, kada pokrenemo akciju brisanja podataka sa drugog uređaja, napadač ili lopov može pokrenuti ponovno cijeli sustav, tako da se akcija brisanja zaustavlja. To se može spriječiti postavljanjem zaštite na samu lozinku. Akcija brisanja je dostupna samo onda kada je ukradeni uređaj spojen na mrežu u suprotnom to neće biti moguće.



Sl. 4.4. Prikaz akcije brisanja podataka sa uređaja [13]

VPN (*engl.* Virtual private network)

Virtualna privatna mreža je tehnologija koja stvara šifriranu vezu preko manje sigurne mreže. Prednost korištenja VPN-a je ta šta osigurava odgovarajuću razinu sigurnosti preko spojenog sustava kada je mrežna infrastruktura nedovoljno sigurna [16]. Najčešće vrste VPN-a su daljinski pristup i *site-to-site* VPN. Daljinski pristup koristi javnu telekomunikacijsku infrastrukturu poput interneta

kako bi pružio korisnicima siguran pristup mreži. Ovaj sustav je posebno važan kada zaposlenici neke tvrtke koriste javne W-Fi Hotspotove. VPN se spaja na korisnikovo računalo ili se mobilni uređaj spaja na VPN ulaz u mrežnoj organizaciji. Prilikom uporabe sustav obično zahtjeva provjeru identiteta korisnika. Zatim se stvara mrežna veza natrag prema uređaju koja omogućava pristup internim mrežnim resursima. *Site-to-site* VPN koristi ulaz uređaja za spajanje na cijelu mrežu sa jednog mjesta na drugo. Uređaji sa krajnjeg čvora na udaljenom mjestu ne trebaju VPN klijente jer ulaz upravlja sa konekcijom uređaja. Većina konekcija preko interneta koristi Ipsec. Moguće je koristiti i MPLS oblake, umjesto javnog interneta.

5. SIGURNOSNE PREPORUKE ZA KRAJNJEG KORISNIKA

5.1. Podjela sustava za detekciju

Na osnovu trenutka u kojemu se napad otkriva, sustavi se dijele na one koji napad detektiraju u realnom vremenu i naknadno [17]. Većina današnjih sustava za detektiranje počinje djelovati onda kada se pojavi napad, zatim pokušava odbaciti opasne pakete i prekinuti sesiju za napade koji su zasnovani na TCP-u. Sistem dinamički postavlja određena pravila mrežne zaštite. Svi poznati izvori napada mogu biti zaustavljeni ili im se blokira daljnji pristup mreži tako što se stavljaju na "crnu listu". Svi ostali izvori kojima se vjeruje imaju stalni pristup i oni se nalaze na "bijeloj listi". Sustavi se dijele na pasivne i aktivne. Pasivni sustavi djeluju tako da se generira alarm i šalje korisniku, te od samog korisnika ovisi hoće li poduzeti bilo kakvu akciju kako bi blokirao aktivnost. Aktivni sustavi djeluju tako što alarmiraju korisnika te sami poduzimaju određene korake u daljnjem sprječavanju štetne radnje. Neke od faza odgovora na napade i rješavanje su:

- Priprema- pravljenje kontrolnih sabirnica za binarne datoteke, koje su smještene na uređaju s kojih se može samo čitati. Generiranje "otiska" sustava, na osnovu njega se prate sva odstupanja i neovlašteni upadi u sustav
- Identifikacija- Jedan od osnovnih uloga sustava. S ovim se postupkom karakterizira napad, procjenjuju se mogućnosti napadača.
- Ograđivanje- pasivni nadzor; ograničavanje pristupa primjenom mjera, kao što je zaključavanje pristupa sustavu
- Iskorjenjivanje- Faza se provodi da bi se sustav zaštitio
- Oporavak- Vraćanje sistema u stabilno stanje
- Nastavak- Praćenje je posljednja faza i obuhvaća neke od mjera (poduzimanje akcija protiv napadača, obavještenje ostalih korisnika, provjera i analiza onoga što se dogodilo, traženje sigurnosnih propusta i grešaka u samome softveru).

IPS (*engl. Intrusion prevention systems*) je tehnologija koja sprječava upade u mrežu. To je mrežna barijera koja kombinira filtriranje na razini mreže i aplikacije s aktivnim sustavom [17]. Najbolje rješenje koje se preporučuje da mrežna barijera bude konfigurirana da odbija sav dolazeći promet. Hakeri će uvijek pronaći rješenje za novi napad koji još nije otkriven, tj. nisu poznati onima koji

proizvode IDS (*engl. Intrusion detection system)* sisteme. Veliki je problem održavanje baze potpisa ažuriranim, te se tako suočava sa problemom prevelike količine podataka koje treba analizirati. Veliki problem ovog sustava je taj što je jako osjetljiv na napade, lako ga je zavarati. Nakon nekih napada, suludo troši vlastite resurse čekajući na nepostojeći odgovor od strane mreže. Dodatno zbunjujuće za sustav je skeniranje ili lažiranje IP adrese. 4 su moguća slučaja u vezi IDS-a i njegovog detektiranja napada:

- TP (true positive)- napad je ispravno detektiran
- FP (false positive)- IDS je detektirao nepostojeći napad
- FN (false negative)- napad je postojao, IDS ga nije detektirao
- TN (true negative)- nije detektiran nepostojeći napad

Osjetljivost- definira se kao broj ponavljanja pravih alarma, tj. količnik broja stvarnih upada koje je IDS detektirao i zbroja pravih alarma i propuštenih. Prema istraživanju bitniji su propušteni alarmi [17].

$$osjetljivost = TPR = TP / (TP + FN) \qquad (5\text{-}1)$$

Određenost- definira se kao odnos ispravno detektiranih legitimnih aktivnosti, tj. kao količnik ispravno detektiranih aktivnosti i zbroja stvarno negativnih i lažnih alarma:

$$određenost = TNR = TN / (TN + FP) \qquad (5\text{-}2)$$

IDS sa visokim postotkom određenosti su veoma važni za administratore mreža. Preporučuje se instaliranje IDS-ova sa velikim postotkom određenosti.

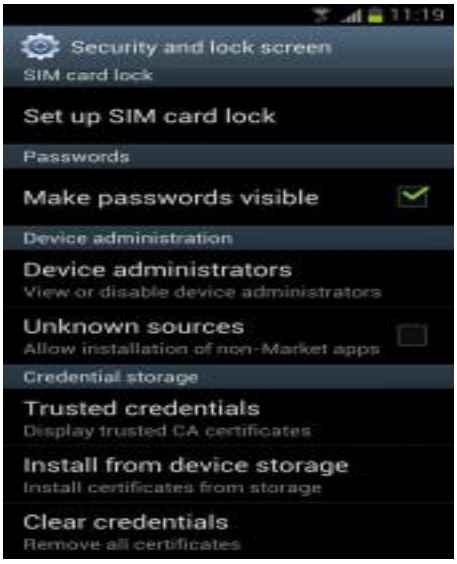
Točnost- Kompromis koji je potreban između osjetljivosti i određenosti. Predstavlja graničnu točku odstupanja od normalnog ponašanja sustava. Obuhvaća i određenost i osjetljivost, to je odnos svih IDS-ovih rezultata (pozitivnih i negativnih) koji su ispravljeni.

IPS je sustav koji automatski sprječava napad te ga blokira, dok IDS sustav prijavljuje taj isti napad korisniku ili administratoru koji bi sam trebao poduzeti odgovarajuće mjere [17].

5.2. Android zaštita

5.2.1. Izbjegavanje nepoznatih izvora

Svaki Android telefon uključuje mogućnost instaliranja aplikacije koje ne dolaze iz trgovine Google Play. Za većinu, ovo je velika prednost [16]. Ovo je mogućnost kako doći do nekih drugih aplikacija i Amazon Appstore-a, i to je jedna od stvari koji pomaže Androidu da ostane otvoren i fleksibilan za sve. Nažalost, vrlo mali dio aplikacija koji zatraži pristup osobnim informacijama se kratko zadržava u telefonu. Taj problem ostavlja širom otvorena vrata, što je jako opasno za uređaj. Kako bi se zaštitio uređaj od neočekivanog instaliranja stranog tijela u uređaju, trebala bi se uskratiti ta mogućnost. Ova mogućnost je obično uskraćena Android verzijama 4.0. pa nadalje. Na sl. 5.1. su prikazane postavke Android uređaja 4.0. pa nadalje gdje je moguće kliknuti na zabranu instaliranja aplikacija od nepoznatih izvora.

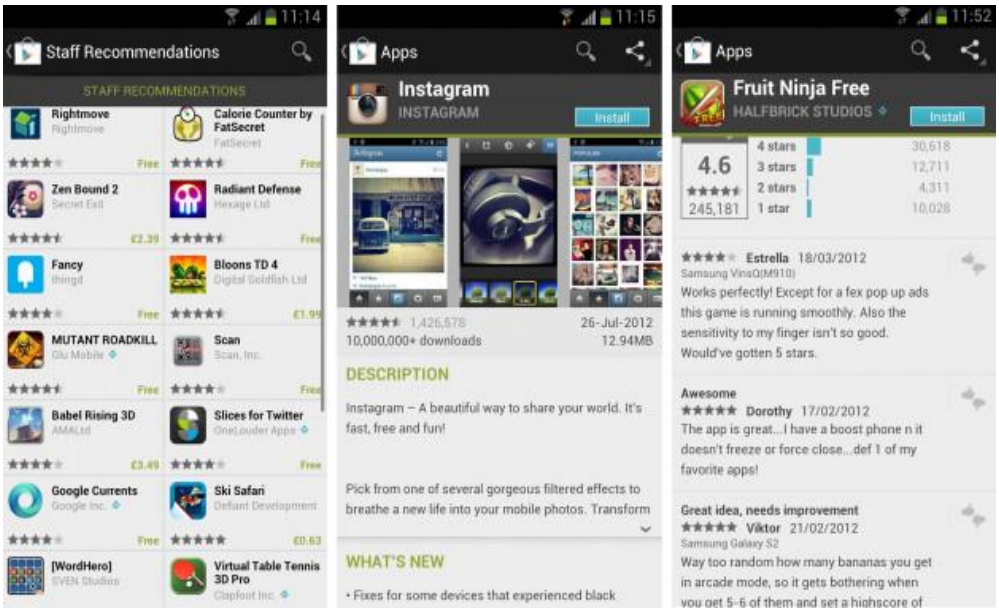


Sl. 5.1. Prikaz postavki Android uređaja [18]

5.2.2. Izbjegavanje ilegalnih aplikacija

Postoje dobri razlozi za instaliranje aplikacija koje se ne nalaze u Google Play trgovini, ali ne onoliko kao prijašnjih godina. Google je naporno radio na postavljanju korisničkih područja ispitivanja za tvrtke koje žele beta testiranja svih značajki, koje se koriste kao najveći razlog učitavanja

programa aplikacije. Postoje legitimne aplikacije koje krše Google-ove uvjete pružanja usluge nudeći vlastite aplikacijske trgovine ili ponudu koja se ne slaže sa pravima Google-a (aplikacije za odrasle i sl.). Preporuka je da se instalira aplikacija izvan preporučene trgovine samo ako ona dolazi iz sigurnih izvora i ako je lako provjerljiva. Preuzimanje aplikacija iz Google Play trgovine također nije sigurna, ako dolazi iz te trgovine ne znači da je definitivno sigurna za preuzimanje. Potrebno je pogledati recenziju i popularnost same aplikacije od drugih korisnika [14]. Mogu se uočiti potencijalni i tehnički problemi aplikacije. Što je veća popularnost i više preuzimanja aplikacije, sigurnije je preuzimanje. Najveća opasnost se pojavljuje od novih aplikacija sa malim brojem povratnih informacija. Potrebno je provjeriti je li programer legitiman . Treba provjeriti je li aplikacija ispravna. Svaka aplikacija koja se instalira na bilo kojem Android uređaju mora prikazati koji su dijelovi operacijskog sustava uključeni sa svojim osobnim podacima, tj. kojim podacima može upravljati i raspolagati. Ova informacija je vidljiva u trenutku donošenja odluke da se aplikacija instalira i stvarnog vremena da se softver zapravo instalira, dajući dovoljno vremena da se vidi koja su to dopuštenja. Ovi oglasi sa svim pravilima se mogu pročitati u desetak sekundi kako bi se prepoznalo je li sadržaj pun virusa ili je dobronamjeran.



Sl. 5.2. Prikaz povratnih informacija određenih aplikacija [18]

Većina virusa lako može preuzeti kontrolu nad kreditnim karticama, lozinkama i ostalim osobnim podacima. Neki su u mogućnosti prikupiti i slati listu kontakata, email adrese, snimati razgovore. Neki od znakova su usporavanje sustava, sumnjive poruke, preusmjeravanje na sumnjive stranice. Neki od virusa koji dolaze preko aplikacija na Android uređaje su: NotCompatible virus, virus koji se spaja na server uređaja te krade osobne informacije sa uređaja [14]. Drugi virus je Lastcloud virus, koji se ponaša kao trojanski konj. Pojavljuje se u obliku ažuriranja nekih aplikacija (npr. Whatsapp), pokušava ukrasti listu kontakata, ili broj računa. Treći virus je Android Police Virus, nova verzija FBI virusa, virus koji blokira cijeli sustav i može vršiti enkripciju na svim datotekama. Može se primijetiti tako što traži korisničko ime i lozinku na određenim web stranicama. Svaki ovaj virus moguće je ukloniti pomoću određenog antivirusnog programa koji se nudi Google trgovinama.

5.3. iOS zaštita

5.3.1. Ažuriranje sustava

Uređaj bi trebao biti u toku sa najnovijim promjenama operacijskog sustava. Potrebno je instalirati ažuriranje jer se s tim postupkom osigurava sustav od neželjenog upada nepoznate osobe. Skladište podataka je nepromjenjiv skup podataka, te je razumljivo da kao takav može zastarjeti, odnosno da je u zaostatku [15]. Pravilno vremensko ažuriranje sa novim podacima dovodi do znatnih promjena u sustavu, te skladište postaje stvarno-vremenski model. Ovakav model je prilagođen modernom modelu. Određena ažuriranja sustava omogućuju dohvat potrebnih resursa za sklopovlje čime se osigurava ubrzanje izvršavanja određenog zadatka, npr. igrice. Analize često otkrivaju da su radnje za održavanje osobnog uređaja prezahtjevne za običnog korisnika. Potrebno je izvršiti na desetke novih ažuriranja sustava kako bi se popravili deseci propusta u programima. Upravo ta složenost održavanja ukazuje na potrebu za preciznim sustavom i alatima za procese ažuriranja sustava [15]. No, ipak većina programera može konfigurirati automatska ažuriranja koristeći određene pravilnike . Neki od korisnika pokušavaju ažurirati svoj uređaj preko 3G ili 4G mreže,a ne preko Wi-Fi mreže. Nakon tog ažuriranja potrebno je spojiti uređaj na iTunes prije nego se ponovno pokrene

Sl. 5.3. Prikaz nadogradnje sustava [19]

5.3.2. Jaillbreak

Proces skidanja zaštite postavljene od strane Apple-a, tj. ukidanje programskih ograničenja. Proces dopušta pristup nadogradnje sustava svim datotekama sustava, preuzimanje dodatnih aplikacija, proširenja i svega onoga što je nedostupno putem Apple App Store-a [16]. iOS pokušava zaustaviti proces sa dodavanjem novog hardvera. U većini država to nije ilegalno, smatra se da hakeri time ne krše autorska prava. Proces se ne preporuča od strane Apple-a, dok ga hakeri preporučuju jer se sustav ubrzava do 6 puta, brži je pristup. Postupak je opasan jer se njime daje potpuni pristup nepotpisanim aplikacijama, što nosi dodatne rizike za uređaj [13].

Za razliku od drugih mobilnih platformi, iOS ne dopušta korisnicima da instalira potencijalno zlonamjerne nepotpisane aplikacije s web stranica, ili pokretanje nepouzdanog koda za vrijeme izvođenja. Operativni sustav je izgrađen za samo prihvaćanje i instaliranje softvera koji je odobren od strane Apple-a. Postoji niz aplikacija koje nude sigurnost uređaja, kao što su Symantec Mobile Encryption i Symantec Secure E-mail [13]. Dizajnirani su obično da se integriraju s iPad-om u poslovnom okruženju, omogućujući sigurno komuniciranje sa ostalim uređajima. Instaliranje anti-virusnih programa na Apple uređajima nije potrebno, jer je sustav dizajniran i izgrađen za prihvaćanje i instaliranje jedino softvera koji je pregledan od Apple i pokrenut kroz Apple trgovinu. Virusi se obično pojavljuju samo na uređajima koje je korisnik sam nadogradio procesom *jaillbreak*-a. Dobra strana sustava je izrada sigurnosne kopije putem iCloud-a ili iTunes-a. Kopija se stvara uz Wi-Fi mrežu na servisu iCloud, uređaj se ne mora spajati na osobno računalo i sl. Servisi obuhvaćaju gotovo

sve podatke i sve postavke spremaju [16]. ICloud ne pokriva spremanje osobnih fotografija te zbog toga može doći do curenja u javnost. Zbog toga bolje je verzija iTunes, jer izrađuje sigurnosnu kopiju cijelog sadržaja. Ne podržava preuzete podatke i fotografije. Na novijim verzijama sustava ovi programi su već automatski pokrenuti, i kada sami onemogućite spremanje na određeni servis, svi podaci će biti pohranjeni od strane Apple-a, kako bi se moglo kasnije pristupiti sadržaju.

6.ZAKLJUČAK

Detektiranje virusa u mobilnim aplikacijama sve je teže jer su autori istih sve vještiji u kreiranju. U praksi se koriste statičke i dinamičke metode otkrivanja. U operacijskom sustavu Android koriste se razni alati za otkrivanje i uspoređivanje virusa. Opisan je detaljno alat Alterdroid koji je korišten u otkrivanju zamaskiranih virusa u aplikacijskim dijelovima koda. Radi na principu analiziranja ponašanja originalne verzije aplikacije i blago modificirane. Promatraju se aktivni potpisi, te se putem Levenstheinove udaljenosti računa razlika koja je vidljiva u dijelu potpisa. Analizirao se diferencijalni potpis kroz proces uzimanja uzoraka i uspoređivanju dvaju uzoraka. Svaka aplikacija ima svoju klasu, te tako dolazi do očekivane entropije. Alat služi kao algoritam za pronalaženje određenih zaraženih dijelova koda u aplikaciji. Prednost alata je što koristi metodu niz po niz, te pronalazi i najmanju razliku u obje verzije aplikacije. Kroz primjere se može vidjeti na koji način se verzije razlikuju, kroz modifikaciju nekih piksela u slikama, kroz različite varijable niza koje sadrže pogrešku u nizu koda. Također pomoću alata je moguće otkriti iz kojeg dijela cure informacije (npr. GPS). Testiran je na štetnim aplikacijama koji su u "sivoj zoni" i na sigurnim aplikacijama, kako bi se utvrdile razlike. Na originalnim aplikacijama nije primijećeno nikakva sumnjiva radnja. Može se zaključiti da je Alterdroid alat koji više služi u detekciji problema nego u rješavanju. Drugi alat koji je opisan je RootGuard, alat koji nudi zaštitu od virusa prilikom procesa *root*-a. Omogućuje korisnicima preuzeti aplikaciju koje su potrebne za određene zahvate i pritom sadrže pojedine police sigurnosti. Sadrži u sebi alat preko kojeg je lakše doći do "podizanja" uređaja, bez ograničenja. RootGuard pruža veliku pomoć korisnicima preko praćenja i izmjena aplikacija prilikom *root*-a. Te tako korisnik može provjeriti sve podatke o aplikaciji. Mjerilo se vrijeme performansi RootGuard-a preko poznate aplikacije AnTuTu na dva uređaja. Može se zaključiti da ima određenih gubitaka u skladištu ulaza/izlaza baze podataka, brzina je niža ali je prihvatljiva. Opisani su detaljni tipovi prepoznavanja u sustavu, od kojih se najviše iskazala biometrijska ovjera koja je još u razvoju. Prednost takve ovjere je ta što je zasebna za svakog pojedinca, stabilna i jednostavna za ponavljanje. Dok se npr. ovjera prepoznavanja glasa može iskriviti zbog buke u pozadini. Tehnika prepoznavanja lica je na 90% uspješnosti te je još u razvoju. IdM je modul koji se pokazao kao najbolji za upravljanje korisnicima pametne trgovine, koristan je u tumačenju korisnikove okoline analizom i usporedbom parametara iz nekoliko uređaja. Pokazalo se kako je operacijski sustav iOS nešto sigurniji od Android-a. Ponajviše zbog toga što provodi sigurnosne mjere protiv ugrožavanja među aplikacijama. Svaka

aplikacija ima svoj polazni direktorij za svoje datoteke. U prednosti je u odnosu na Android jer je jednostavniji operacijski sustav. Na Android platformi se sve podešava ručno dok je to na iOS-u automatski postavljeno. No to ne mora značiti nikakvu sigurnost s pogleda korisničke strane. Nijedna platforma nije sto posto sigurna u današnje vrijeme. Za razliku od drugih mobilnih platformi, iOS ne dopušta korisnicima da instalira potencijalno zlonamjerne nepotpisane aplikacije s web stranica, ili pokretanje nepouzdanog koda za vrijeme izvođenja. Operativni sustav je izgrađen za samo prihvaćanje i instaliranje softvera koji je odobren od strane Apple-a i pokrenut u nj. trgovini. Nije potreban anti-virusni program kao npr. u Android-u, jer nikakve aplikacije koje nisu kontrolirane ne mogu pristupiti sustavu osim naravno ako korisnik ne izvrši proces jaillbreak-a.

7.LITERATURA

[1] Ying-Dar Lin, *National Chiao Tung University*, Chun-Ying Huang, Matthew Wright, Georgios Kambourakis : Mobile Application Security

[2] A.Takanen, J.D. DeMott, and C.Miller, *Fuzzing for Software and Privacy*

[3] Roberto di Pietro, Guillermo Suarez-Tangil and Juan E.Tapiador, Flavio Lombardi, *Alcatel Lucent Bell Labs:* Obfuscated malware via differential analysis

[4] J. Oberhalde and C.Miller. "Dissecting the Android Bouncer", presentation at Summercon 2012; https://jon.oberheide.org/blog/2012/06/21/dessecting-the-android-bouncer.

[5] Yuru Shao, Xiapu Luo, and Chenxiong Qjan, The Hong Kong Polytechnic University: RootGuard:Protecting rooted Android phones

[6] Y.Zhou and X.Jiang, "Dissecting Android Malware:Characterization and Evolution" *Security and Privacy*

[7] Juraj Kačur, Felix Gomez Marmol, Sebastian Schumann and Ondrej Labaj: Appstore:Expanding the froniers of smartphone ecosystems

[8] *ETSI TS 102 796 vl. 2.1. Hybrid Broadcast Broadband TV*, European Telecommunications Standars., 2012

[9] Alessandro Armando, Gabriele Costa and Luca Verdarame, Alessio Merlo: Securing the "Bring your own device" paradigm

[10] H.Lockheimer: "Android and Security". Blog.2.Feb.2012; http://googlemobile.blogspot.it/2012/02/android-and-security.html.

[11] Apple Developer-Ios Dev Center, http://developer.apple.com/devcenter/ios/index.action

[12] Gursev Kalra, Mobile application security testing

[13] iOS 6.0.1. Released.Fiy to iPhone 5 upgrade issue for iOS 6.0.1., *Way 2 iOS,by Krishna*

[14] Jason Murdock, *Apple iOS vs Google Android: Which is the more secure smartphone OS*

[15] Damir Pintar, *Implementacija stvarnovremenskog skladištenja podataka na temelju principa integracije poslovnih aplikacija*

[16] *iOS security guide*

[17] Dragan Pleskonjić, Nemanja Maček, Borislav Đorđević, Marko Carić*, Sigurnost računarskih sistema i mreža*

[18] Beatriz Rojo, *How to install Android apps without using Google Play*

[19] Charlie White, Apple Ios 6.0.1. Update Now Available

8. SAŽETAK

SIGURNOSNI ASPEKTI MOBILNIH APLIKACIJA

U ovom je radu je opisana problematika sigurnosti, detaljno je opisana i sustavno analizirana problematika sigurnosti mobilnih platformi i aplikacija. Istražene su i objašnjene pojedine metode i tehnike za detekciju zlonamjernih aktivnosti. U radu su opisani sustavi koji su fokusirani na analizu, detekciju i razvoj štetnih aplikacija. Detaljno su analizirani i prikazani rezultati odgovarajućih alata za protumjere. Analizirani su sigurnosni algoritmi, protokoli i okviri koji se primjenjuju na suvremenim mobilnim platformama. Uspoređene su mobilne platforme. Analizirani su njihovi parametri sigurnosti te mehanizmi zaštite.

Ključne riječi: Web aplikacije, RootGuard, Alterdroid, diferencijalna analiza, SMM, CoI-Components of interest, root, BYOD, sigurnost, iOS, Android, jaillbreak.

9.ABSTRACT

SAFETY ASPECTS OF MOBILE APPLICATIONS

This thesis describes the problem of mobile platforms and applications security in detail and analyses systematically the safety problems. The detection of malicious activity has been investigated and explained with various methods and techniques. The thesis describes systems that are focused on the analysis, detection and development of harmful applications and presents the results of appropriate tools for counter measures. Security algorithms, protocols, and frameworks that apply to modern mobile platforms have been analyzed. Different mobile platforms have been compared. Their security parameters and safety mechanisms have been analysed.

10.ŽIVOTOPIS

Ilija Dumančić, rođen u Ebersbergu (Njemačka), 09. listopada 1992.godine. Osnovno obrazovanje stječe u Osnovnoj školi fra Miroslava Džaje u Kupresu (BiH). Po završetku škole upisuje Opću gimnaziju u Kupresu. Tijekom srednjoškolskog obrazovanja sudjeluje na državnom natjecanju iz "Njemačkog jezika" u Sarajevu gdje osvaja 2.mjesto. Srednju školu završava 2011.godine te upisuje preddiplomski studij Računarstva u Splitu na FESB-u. 2013.godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku.

U Osijeku, 2016.

Ilija Dumančić

_____