Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Žitnik

Detekcija botnetov iz podatkov omrežnega prometa z razširitvijo na mobilnih napravah

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Dobravec

Ljubljana, 2015

Izjava o avtorstvu magistrskega dela

Spodaj podpisani Anže Žitnik sem avtor magistrskega dela z naslovom:

Detekcija botnetov iz podatkov omrežnega prometa z razširitvijo na mobilnih napravah

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Dobravca,
- so elektronska oblika magistrskega dela, naslov (slovenski, angleški), povzetek (slovenski, angleški) ter ključne besede (slovenske, angleške) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, 7. septembra 2015 Podpis avtorja:

Zahvala

Za pomoč se zahvaljujem mentorju dr. Tomažu Dobravcu. Zahvala gre tudi podjetju XLAB d.o.o. ter še posebej Alešu Černivcu. Hvala Brini za moralno podporo.

Anže Žitnik, 2015

Kazalo

Povzetek

Cilj te magistrske naloge je bil spoznavanje klasičnih in mobilnih botnetov ter možnosti njihove detekcije, implementacija detektorja botnetov iz podatkov omrežnega prometa in mobilne aplikacije za detekcijo zlonamerne programske opreme na operacijskem sistemu Android. Izdelali smo detektor botnetov, ki uporablja model strojnega učenja za uvrščanje omrežnih tokov med legitimen promet ali promet botnetov. Detektor smo ovrednotili s testiranjem na dva različna načina ter komentirali njegove prednosti in omejitve. Razvili smo aplikacijo za platformo Android, ki zaznava zlonamerno programsko opremo s spremljanjem omrežnih povezav na nevarne vire in izkoriščanja nekaterih znanih varnostnih ranljivosti v operacijskem sistemu. Aplikacijo smo testirali na nekaj primerih zlonamernih programov ter jo ponudili za prenos uporabnikom uradne trgovine Android.

Ključne besede

računalniška varnost, botnet, detekcija botnetov, analiza omrežnega prometa, varnost mobilnih naprav, zlonamerna programska oprema, Android

Abstract

The goal of this thesis was a study of classic and mobile botnets and the possibilities of their detection, implementation of a network traffic based botnet detector and a mobile application for malware detection on the Android operating system. We created a botnet detector that uses a machine learning model for classification of network flows as either legitimate or botnet-induced traffic. We evaluated the detector by two distinct testing procedures and commented on its advantages and limitations. We developed an Android application that detects malware by observing network connections to malicious resources and exploiting some of the known security vulnerabilities in the operating system. We tested the application on some malware samples and offered it to the users of the official Android marketplace.

Keywords

computer security, botnet, botnet detection, network traffic analysis, security of mobile devices, malware, Android

Poglavje 1

Uvod

Na področju računalniške varnosti smo v zadnjih letih priča širjenju botnetov kot infrastrukture za izvrševanje vseh vrst spletnih napadov in različnih kriminalnih aktivnosti. Aktivnosti spletnih napadalcev, ki jih omogočajo botneti, škodujejo nešteto uporabnikom interneta, napadalcem pa prinašajo velike dobičke. To je pripeljalo do razvoja rešitev za detekcijo botnetov, katerim pa se botneti poskušajo izogniti s pogostim posodabljanjem. Z razmahom pametnih mobilnih naprav se botneti pojavljajo tudi tam, kar prinaša dodatne nevarnosti za uporabnike ter izzive za detekcijo in obrambo pred botneti.

Večina obstoječih metod detekcije botnetov se zanaša na detekcijo znanih vzorcev bodisi v omrežnem prometu bodisi v zlonamernih izvršljivih datotekah, slabosti tega pa sta potreba po pogostem posodabljanju detektorjev in možnost izognitve detekcije s spreminjanjem delovanja ali programske kode botnetov. Obstoječe rešitve za detekcijo zlonamernih programov na mobilnih napravah se ne ukvarjajo z grožnjo botnetov, temveč opozarjajo na splošne varnostne grožnje.

V okviru tega magistrskega dela smo raziskali različne tipe botnetov, njihovo delovanje ter metode detekcije. Proučili smo pojav botnetov na mobilnih napravah, njihovo delovanje in razlike v primerjavi s klasičnimi botneti na osebnih računalnikih. Raziskali smo tudi varnostne ranljivosti operacijskega

sistema Android, ki omogočajo širjenje zlonamerne programske opreme.

Implementirali smo detektor botnetov delujoč na podatkih omrežnega prometa, ki s tehniko strojnega učenja, naključnimi gozdovi, klasificira omrežne tokove glede na izvor iz botnetov ali iz legitimnih aplikacij. Idejo za implementacijo detektorja smo dobili iz članka Singh et al. [1]. Sistem smo dopolnili z uvedbo dodatnega klasifikacijskega atributa. Uspešnost detekcije smo testirali na zajetem omrežnem prometu botnetov, ki smo ga pridobili iz spletne zbirke zlonamerne programske opreme, naš prispevek glede na drugo literaturo pa je testiranje detektorja botnetov na lastnem omrežju virtualnih računalnikov z nameščenimi primerki zbranih botnetov, kar je pokazalo slabše rezultate. Ocenili in komentirali smo tudi možnost zaznave mobilnih botnetov z uporabo izdelanega omrežnega detektorja.

Razvili smo mobilno aplikacijo za detekcijo zlonamerne programske opreme na operacijskem sistemu Android, ki je sposobna zaznavanja omrežnih povezav do nevarnih virov in detekcije zlonamernih aplikacij z ugotavljanjem uporabe privilegiranih pravic in izkoriščanja nekaterih varnostnih ranljivosti sistema Android. Glede na druge znane tovrstne rešitve je prednost naše aplikacije detekcija tako imenovanih ugrabitev sporočil SMS, česar se poslužuje mnogo zlonamernih programov na mobilnih napravah. Za podporo mobilni aplikaciji smo implementirali tudi spletni strežnik, ki zbira podatke o dogodkih iz mobilnih naprav in jim posreduje informacije o znanih zlonamernih spletnih virih in aplikacijah iz zunanjih podatkovnih baz.

V nadaljevanju tega poglavja bomo predstavili literaturo, ki opisuje sorodna dela s področja omrežne detekcije botnetov z zaznavo anomalij ter mobilnih detektorjev zlonamerne programske opreme. Poglavje 2 opisuje botnete na splošno, njihov razvoj, grožnje, ki jih predstavljajo ter možne metode detekcije botnetov. V poglavju 3 predstavimo botnete na mobilnih napravah in razlike s klasičnimi botneti, načine okužbe in metode detekcije zlonamernih programov ter nekatere varnostne ranljivosti mobilnega operacijskega sistema Android. V poglavju 4 opišemo implementacijo našega detektorja botnetov iz omrežnega prometa. Predstavimo in komentiramo

tudi podatke, uporabljene pri testiranju ter način in rezultate testiranj. Poglavje 5 govori o razvoju mobilne aplikacije za detekcijo zlonamerne opreme. Poleg implementacije aplikacije, opisa delovanja in pregleda uporabniškega vmesnika, predstavimo tudi pripadajoči spletni strežnik ter opišemo testiranje na dveh primerih zlonamernih aplikacij. Magistrsko nalogo zaključimo s sklepnimi ugotovitvami, kjer povzamemo in komentiramo rezultate našega dela ter predlagamo možnosti za nadaljnji razvoj.

1.1 Sorodna dela

Gu et al. [2] predlagajo metodo zaznavanja botnetov, ki temelji na predpostavki, da si posamezni okuženi računalniki (boti) znotraj botneta delijo podobne vzorce komunikacije in zlonamernih aktivnosti. Vzorce komunikacije pridobi z gručenjem (*clustering*) zapisov dnevnika omrežnega prometa, vzorce zlonamernih aktivnosti pa z gručenjem zapisov iz sistema za detekcijo vdorov. S korelacijo obeh vzorcev sistem zazna prisotnost botneta v omrežju ter njegove posamezne člane. Predlagana rešitev je ena prvih, ki je neodvisna od protokola in strukture komunikacije botneta. Članek poroča o zelo dobrih rezultatih testiranja, ki pa je bilo izvedeno z dokaj majhno množico vzorcev tedaj znanih botnetov. Slabost sistema je omejena skalabilnost ter nezmožnost zaznave botov, ki ne vršijo zaznavnih zlonamernih aktivnosti, prav tako pa sistem ne zazna okuženega računalnika, če je ta edini okužen znotraj zaznavnega omrežja.

Lu et al. [3] predstavljajo še eno metodo zaznavanja botnetov iz omrežnega prometa. Njihov sistem najprej razporedi omrežni promet med znane protokole in aplikacije, nato pa detektira anomalije v pogostosti pojavljanja n-gramov v vsebini paketov posamezne aplikacije s pomočjo gručenja z metodo voditeljev (*k-means*). Tak pristop dobro detektira botnete, ki temeljijo na protokolu IRC, velika slabost pa je nezmožnost analize šifriranega prometa in botnetov, ki uporabljajo neznane protokole.

Zhang et al. [4] predlagajo sistem za detekcijo botnetov, ki odpravlja ne-

katere pomanjkljivosti metode, opisane v [2]; sistem je zasnovan skalabilno, detekcija botnetov pa je tukaj neodvisna od zaznave zlonamernih aktivnosti. Metoda iz omrežnega prometa najprej izloči ves potencialno sumljiv promet P2P (vsak z vsakim), iz tega pa nato oblikuje gruče, ki ločijo promet legitimnih programov P2P od botnetov. Ideja pri ločevanju nevarnega od legitimnega prometa P2P je, da si je promet botov posameznega botneta med seboj bolj podoben kot pri legitimnih P2P programih. Slabost je v tem, da lahko detektira le botnete, temelječe na arhitekturi P2P, kot pri [2], pa je tudi tukaj slabost v nezmožnosti zaznave osamljenega okuženega računalnika. Avtorji predstavijo tudi nekaj možnih tehnik, kako bi se lahko botneti v prihodnje izognili detekciji s tem sistemom.

Singh et al. [1] predstavljajo metodo za zaznavanje botnetov v distribuiranem okolju, razvito z namenom visoke skalabilnosti in zmogljivosti pri omrežjih z visoko prepustnostjo v približno realnem času. Opisan sistem uporablja naključne gozdove za klasifikacijo omrežnih tokov kot varne oziroma nevarne. Vhod v klasifikator je množica enostavnih značilk posameznega omrežnega toka. Za ekstrakcijo uporabljenih značilk iz omrežnega prometa ni potreben pregled vsebine paketov, temveč le agregacija metapodatkov posameznih paketov, kar pomeni, da šifriran promet ne vpliva na zmožnost detekcije. Avtorji poročajo o dobrih rezultatih testiranja ter trdijo, da njihova metoda uspešno zaznava tudi nepoznane botnete, ki niso bili prisotni v učni množici. Zaradi omenjenih prednosti smo sistem, opisan v tem članku, uporabili za implementacijo našega detektorja.

Burguera et al. [5] predlagajo sistem za zaznavo zlonamernih programov na prenosnih napravah z operacijskim sistemom Android. Ugotavljajo, da je večina nevarnih programov na teh napravah skritih znotraj prepakiranih legitimnih aplikacij. Na podlagi zbranih vzorcev uporabe sistemskih klicev posameznih aplikacij na množici naprav zaznavajo anomalije in s tem detektirajo primerke aplikacij, ki se obnašajo nenavadno. Računsko zahtevne operacije se izvajajo na centralnem strežniku, kar olajša porabo energije na mobilnih napravah, ta pa vseeno ni zanemarljiva zaradi nenehnega spremlja-

nja sistemskih klicev. Slabost metode je v nezmožnosti zaznave zlonamernih programov, ki nimajo ustreznega legitimnega para. Dodatna slabost je, da potrebujemo za spremljanje sistemskih klicev na sistemu Android administratorske pravice, katerih pridobitev pa lahko, kot bomo pojasnili v poglavju 3.4.1, prinaša dodatne varnostne grožnje.

Shabtai et al. [6] predstavljajo metodo s podobno idejo kot pri [5], le, da namesto sistemskih klicev spremlja več značilk uporabe sistema, kot so: poraba energije, zasedenost procesorja, število prenesenih omrežnih paketov, poslanih sporočil SMS, ipd. Sistem zaznava anomalije v vzorcu teh značilk. Analiza se vrši na sami mobilni napravi, kar neugodno vpliva na porabo energije. Če zlonamerne akcije neželenih programov trajajo zelo kratek čas, ali pa imajo premalo vpliva na sistemske indikatorje, sistem tovrstnih groženj ne zazna. Prav tako metoda težko določi, kateri program na napravi je škodljiv, ker temelji na indikatorjih, ki so lahko skupni mnogim programom.

V našem delu smo implementirali omrežni detektor botnetov po ideji iz članka Singh et al. [1]. Prednost pred ostalimi omenjenimi rešitvami je predvsem v splošnosti zaznavanja — sistem se ne omejuje na posebne vrste botnetov. V članku so več pozornosti posvetili povzporejanju algoritmov ter zmogljivostnemu vidiku detekcije, mi pa smo se osredotočili na dodatno testiranje uspešnosti detekcije. Izvorno rešitev smo dopolnili z dodatnim klasifikacijskim atributom in testiranjem na omrežju virtualnih računalnikov.

Detekcije zlonamernih programov na mobilnih napravah smo se lotili z aplikacijo za zaznavanje nevarnih povezav in izkoriščanja varnostnih ranljivosti. Pri razvoju smo upoštevali porabo energije in se omejili na rešitve, ki ne zahtevajo administratorskega dostopa do naprave.

Poglavje 2

Botneti

Botnet je množica okuženih računalnikov, ki jih, z namenom izvajanja zlonamernih aktivnosti, upravlja skupen gospodar. Beseda botnet izvira iz okrajšanih angleških besed *robot network*, pomeni torej omrežje robotov, uveljavila pa se je skoraj izključno z negativno konotacijo — za označevanje zlonamernih tovrstnih omrežij. Ustreznega slovenskega prevoda besede nismo zasledili, prav tako pa se ga nismo domislili. Posamezen računalnik član botneta imenujemo bot, gospodarju botneta pa s tujko pravimo tudi botmaster. Boti sprejemajo ukaze, izvajajo akcije in posredujejo pridobljene podatke gospodarju. Ukazi lahko pripadajo vnaprej definiranemu naboru, upravitelj pa lahko botom posreduje tudi poljubno programsko kodo. Strežnik, ki ga upravlja gospodar botneta za posredovanje ukazov posameznim članom, z angleško kratico imenujemo strežnik C&C[1] (tudi CnC), oziroma ukaznonadzorni strežnik. Takih strežnikov je lahko zavoljo redundance več, kot bomo spoznali v poglavju 2.3, pa napredni botneti delujejo brez strežnikov C&C. Botneti se tipično širijo avtomatsko (več o širjenju v poglavju 2.4), praviloma pa se uporabniki računalnikov, ki pripadajo botnetu, tega ne zavedajo.

[1] *Command and control server*

2.1 Motivacija avtorjev in napadalcev

V nasprotju s črvi in virusi iz skrajnega konca prejšnjega stoletja, katerih glavni namen je bil širjenje slave njihovih avtorjev, so sodobni botneti zasnovani z motivacijo finančnih dobičkov. Zato so avtorji ali uporabniki botnetov pogosto velike, dobro organizirane kriminalne skupine. Kot ugotavljata Wang in Ramsbrock [7], sodelujejo v izdelavi botnetov tudi nekateri izmed najboljših strokovnjakov na področju računalniške varnosti, še posebej v državah vzhodne Evrope in v Rusiji, kjer je mnogo takih strokovnjakov, a so zaposlitvene priložnosti v legitimnem računalništvu pomanjkljive.

Botnet je torej potencialno zelo veliko distribuirano omrežje računalnikov, ki je lahko uporabljeno kot orodje za mnogo različnih namenov, med najpogostejšimi, kot so ugotovili v študijah [8, 9, 7], pa so:

Izvajanje distribuiranih napadov za zavrnitev storitev (DDoS)

Pri napadih DDoS[2] potrebuje napadalec veliko pasovno širino povezave, zato so botneti kot nalašč za izvedbo takih napadov. Napade DDoS lahko izrabljajo podjetja za onemogočanje njihove konkurence, ali pa napadalci za izsiljevanje legitimnih podjetij. Pogosto podjetja raje plačajo odkupnino, kot da bi tvegala izpad njihove storitve.

Masovno pošiljanje neželenih sporočil

Neželena sporočila (*spam*) so lahko uporabljena za oglaševanje, tako imenovano ribarjenje podatkov (*phishing*), ali razne prevare, kot sta znani Nigerijska [10] in Loterijska prevara. Pošiljanje neželenih sporočil prinaša napadalcem velike dobičke, čeprav se nanje odzove zelo majhen delež naslovnikov. Botneti napadalcem priskrbijo mnogo različnih naslovov, odkoder lahko pošiljajo pošto, in jim tako pomagajo pri izogibanju črnih seznamov pošiljateljev.

Kraja osebnih podatkov

Osebne podatke, kot so naslovi elektronske pošte, številke kreditnih kar-

[2] *Distributed denial of service*

tic, ali uporabniška imena in gesla spletnih storitev, botneti iz okuženih računalnikov pridobijo z iskanjem podatkov po disku, nadziranjem tipk na tipkovnici (*keylogging*), ali pa s spreminjanjem DNS nastavitev na računalniku sodelujejo pri ribarjenju uporabniških podatkov.

Razširjanje botneta ali drugih zlonamernih programov

Boti lahko na gostiteljski računalnik namestijo drugo škodljivo programsko opremo, ali pa poskušajo okužiti druge računalnike na omrežju, na primer z izkoriščanjem ranljivosti v programski opremi.

Zloraba računalnikov za povečanje števila ogledov oglasov

Boti so lahko uporabljeni za avtomatsko večkratno klikanje na oglase na spletnih straneh napadalcev. Napadalci služijo z denarjem oglaševalcev, ti pa od tega seveda nimajo koristi, saj boti ne kupujejo njihovih izdelkov.

Manj pogosti primeri uporabe botnetov so:

Rudarjenje digitalnih valut

Ker se za rudarjenje digitalnih valut uporabljajo distribuirani algoritmi, se lahko botneti uporabijo tudi za ta namen. Čeprav obstaja več botnetov s funkcijo rudarjenja digitalnih valut, pa v poročilu podjetja McAfee [11] (prvo četrtletje 2014) raziskovalci ugotavljajo, da se, kljub veliki računski moči in nizki ceni botnetov, to napadalcem ne izplača več. Razlogi za to so povečujoča se zahtevnost rudarjenja kriptografskih valut, nespecializirana strojna oprema, ki jo botneti okužijo, ter velika verjetnost zgodnjega odkritja botov pri visoki porabi sistemskih virov.

Izsiljevalski programi

Izsiljevalski programi (*ransomware*) so programi, ki uporabniku onemogočijo uporabo sistema ali nekaterih datotek na sistemu, ter za povrnitev stanja zahtevajo odkupnino. Tipično to dosežejo tako, da šifrirajo uporabnikove datoteke in zahtevajo plačilo za izdajo dešifrirnega ključa.

Skladnost z definicijo botneta je pri samih izsiljevalskih programih vprašljiva, ker ne potrebujejo sprejema ukazov za svoje delovanje in je opisana funkcionalnost lahko njihova edina funkcija, prav tako pa so hitro opaženi s strani uporabnikov. Med primere uporabe botnetov pa jih uvrščamo, ker so botneti pogosto vir širjenja tovrstnih programov.

Umetno izboljšanje pozicije spletnih strani v iskalnikih

Z množičnim odpiranjem spletnih povezav preko iskalnika lahko napadalec doseže boljšo pozicijo želene strani v rezultatih spletnega iskalnika [12]. Posledično večji obisk lahko napadalec izkoristi za širjenje zlonamernih programov preko spletne strani.

Avtorji botnetov in spletni napadalci so pogosto ločene osebe ali organizacije. Avtorji botnetov prodajajo programsko opremo in podporo uporabnikom oziroma napadalcem, ti pa lahko kupijo ali za določen čas najamejo tudi storitve že vzpostavljenih botnetov. Ponujena programska oprema botnetov je torej lahko le ogrodje za širjenje naročnikove lastne zlonamerne kode, ali pa je že specializirana za določen namen. Ker posledično več botnetov izhaja iz enakega ali podobnega ogrodja, je določen del njihovega širjenja in delovanja soroden, to pa nam, po analizi nekega tipa botneta, olajšuje zaznavo in obrambo pred sorodnimi vrstami.

2.2 Zgodovina in razvoj

Kot pravi literatura [13, 7, 14, 15], ideja botnetov izvira iz omrežij IRC[3]. To so omrežja za sporočanje z enostavnim komunikacijskim protokolom z osrednjim strežnikom, kamor so povezani odjemalci, ki klepetajo v posameznih kanalih. V omrežjih IRC so boti programi, ki uporabnikom pomagajo pri upravljanju kanala, uporabljeni pa so tudi za beleženje statistike uporabe kanalov in igranje iger s človeškimi uporabniki. Eden bolj znanih programov take vrste je EggDrop [16], razvit leta 1993. EggDrop deluje na enem IRC

[3] *Internet relay chat*

kanalu, vsebuje pa tudi funkcijo "botnet", ki omogoča povezovanje več instanc botov z namenom enostavnejšega upravljanja več kanalov in deljenja pravic ter črnih in belih seznamov uporabnikov. Za razvoj botnetov je pomembno omeniti še, da EggDrop omogoča izvajanje poljubnih uporabniških skript, kar pomeni, da predstavlja dovolj zmogljivo ogrodje za izdelavo pravega botneta. Do botneta, kakršne obravnavamo v tej nalogi (po definiciji iz začetka poglavja 2), EggDropu manjka le še prikritje pred uporabnikom in zlonamerni nameni upravitelja.

Ker omrežje IRC omogoča učinkovito komunikacijo med osrednjim strežnikom in mnogo odjemalci, je bilo spoznano kot primeren komunikacijski protokol za oddaljeni nadzor zlonamerne programske opreme. Po ugotovitvah Canavana [13] je bil prvi primer zlonamerne programske opreme, ki je za oddaljeni nadzor uporabljal omrežje IRC, leta 1999 razvit črv PrettyPark. Povezal se je na strežnik IRC ter gospodarju omogočal pregled različnih podatkov o sistemu, na katerem je tekel, in nekaterih uporabniških podatkov, kot so uporabniška imena in gesla klicnih (*dial-up*) povezav z internetom. PrettyPark je imel tudi funkcijo za zagon izvršljive datoteke, prejete preko kanala IRC, kar je omogočalo njegove posodobitve. Zato ga lahko štejemo kot predhodnika pravih botnetov.

Omrežja IRC so bila med zlonamernimi botneti še v letu 2009 [7] najbolj pogosto uporabljen komunikacijski kanal, in se, čeprav redkeje, v ta namen uporabljajo še danes. Ker se omrežja IRC dandanes redko uporabljajo v legitimne namene, je botnete, ki jih uporabljajo, lahko odkriti, v marsikaterem poslovnem omrežju pa je promet IRC celo onemogočen s požarnimi zidovi. Avtorji botnetov so se zato začeli posluževati sprva IRC povezav prek nestandardnih vrat in posebnih implementacij protokola IRC, kljub temu pa se tak omrežni promet dokaj lahko razloči od legitimnega. Zato so botneti začeli uporabljati druge protokole, na primer HTTP, ki je dovoljen v vseh omrežnih postavitvah. Promet botnetov prek protokola HTTP se težko loči od legitimnega prometa, še posebej če je šifriran. Primer HTTP botneta je Bobax [17], ki izvira iz leta 2004.

Že leta 2003 pa so se pojavili prvi necentralizirani botneti, ki uporabljajo topologijo omrežja vsak z vsakim (P2P[4]), kar pomeni, da se boti povezujejo med seboj, torej delujejo brez centralnega upravljalnega strežnika C&C. To upravitelju botneta koristi zaradi težjega odkritja in otežuje prevzem ali uničenje botneta z odpravo kritične točke odpovedi. Do danes se je razvilo več podtipov decentraliziranih arhitektur upravljanja z botneti, podrobneje jih bomo opisali v poglavju 2.3.2.

Vzporedno z razvojem različnih komunikacijskih protokolov za nadzor in upravljanje botnetov so se razvijale tudi funkcionalnosti botov, tehnike za pridobivanje uporabnikovih podatkov in tehnike za prikrivanje na nivoju posameznega bota, kot so šifriranje in spreminjanje (polimorfizem) programske kode med širjenjem. Te funkcije niso omejene na botnete, temveč se pojavljajo v vseh vrstah zlonamerne programske opreme. V tej nalogi se osredotočamo na lastnosti, značilne za botnete, zato teh funkcionalnosti tukaj ne bomo natančneje obravnavali.

Tabela 2.1 prikazuje nekatere najbolj znane primere botnetov z ocenami velikosti v številu botov (okuženih računalnikov) v času njihove največje razširjenosti.

2.3 Tipi botnetov

Glede na način komunikacije z gospodarjem botneta in z drugimi boti v grobem delimo botnete na centralizirano in decentralizirano urejene.

2.3.1 Centralizirana arhitektura

V centraliziranem modelu vsi boti vzpostavljajo povezavo z enim strežnikom C&C, kot je prikazano na sliki 2.1. Prednosti tega modela sta enostavnost in hiter odzivni čas. Protokol, tradicionalno uporabljen v centraliziranih botnetih, je IRC. Omrežje IRC omogoča visoko skalabilnost, pa tudi vzpostavitev več skupin botov, s čimer lahko upravitelj razdeli botnet na skupine

[4] *Peer to peer*

Tabela 2.1: Nekateri najbolj znani botneti. Povzeto po [15].

Leto zaznave	Naziv	Arhitektura	Ocena največje velikosti
2002	SDBot	Centraliziran / IRC	-
2003	Spybot	Centraliziran / IRC	-
2003	Sinit	Decentraliziran (P2P)	-
2004	Bobax	Centraliziran / HTTP	180.000
2006	Nugache	Decentraliziran (P2P)	160.000
2006	Rustock	Centraliziran / HTTP	150.000
2007	Storm	Decentraliziran (P2P)	180.000
2007	Zeus	Centraliziran / HTTP	3.600.000
2008	Torpig	Centraliziran / HTTP	180.000
2008	Conficker	Decentraliziran (P2P)	10.500.000
2011	TDL4	Decentraliziran (P2P)	4.500.000

z lastnimi nalogami. Pri uporabi standardnih IRC ukazov ali standardnih omrežnih vrat lahko sistemi za detekcijo vdorov in požarni zidovi enostavno zaznajo ali preprečijo promet botnetom, ki uporabljajo IRC. Kot smo omenili že v prejšnjem poglavju, pa avtorji botnetov uporabljajo različne implementacije protokola in šifriranje sporočil, da bi se izognili takšni detekciji.

Enostavni detekciji, ki je posledica uporabe nestandardnih protokolov, se avtorji botnetov izogibajo z uporabo bolj pogostih, kot sta HTTP in FTP. Na večini omrežij je prometa HTTP veliko, zato promet botneta ne izstopa in ga težje zaznamo. V nasprotju s protokolom IRC, kjer strežnik pošlje ukaze botom, morajo pri uporabi protokola HTTP ti spraševati za nove ukaze. Poseben ukrep anonimizacije pri botnetih, ki uporabljajo HTTP, je objavljanje ukazov na javnih spletnih straneh, kot so forumi, blogi, in družabna omrežja. Upravitelj lahko na dogovorjenem javnem mestu objavi primerno oblikovano sporočilo, ki ga boti interpretirajo kot ukaz.

Pomanjkljivost pristopa centralizirane arhitekture je kritična točka odpo-

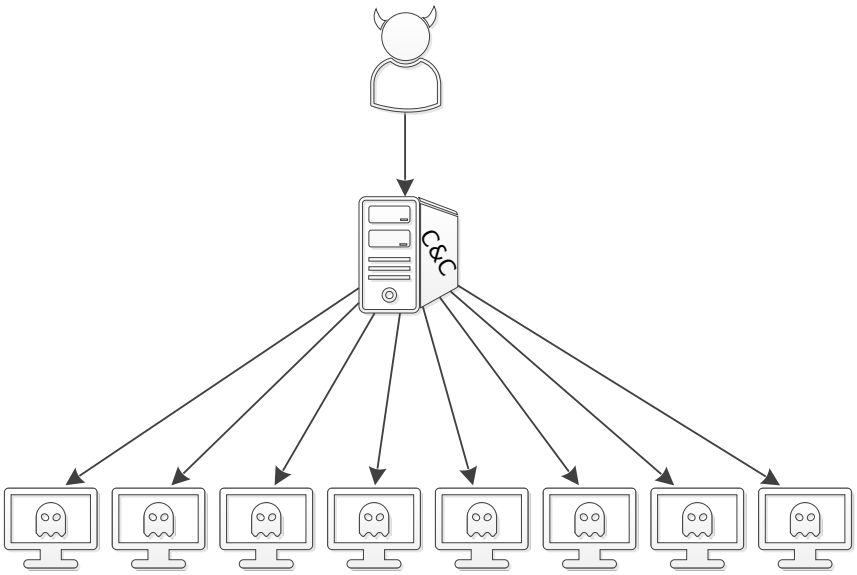
vedi, ki jo predstavlja centralni strežnik. Z analizo posameznega bota lahko ugotovimo njegov naslov in ga onemogočimo. Upravitelji botnetov se temu le do neke mere izognejo z uporabo več strežnikov C&C, pri čemer upravljajo vsakega od njih, ti pa skrbijo vsak za svojo množico botov. Tak model je prikazan na sliki 2.2.

Da bi dosegli še večjo odpornost botneta in anonimnost gospodarja, so lahko ukazno-nadzorni strežniki razporejeni hierarhično na več nivojih (slika 2.3). Za strežnike, razen na najvišjem nivoju, so uporabljeni kar okuženi gostiteljski računalniki botov. Za ta namen se lahko uporabi tiste računalnike, ki jim omrežja dovolijo vhodne povezave. Vsak nivo komunicira le z nivojema višje in nižje od sebe, gospodar pa le z najvišjim nivojem. Tako morajo boti za uspešno komunikacijo poznati le naslove strežnikov na najnižjem nivoju v hierarhiji. Poleg anonimizacije ta način pripomore tudi k izenačevanju obremenitev strežnikov, za ustavitev delovanja botneta pa je potrebno onemogočiti celoten nivo strežniških botov.

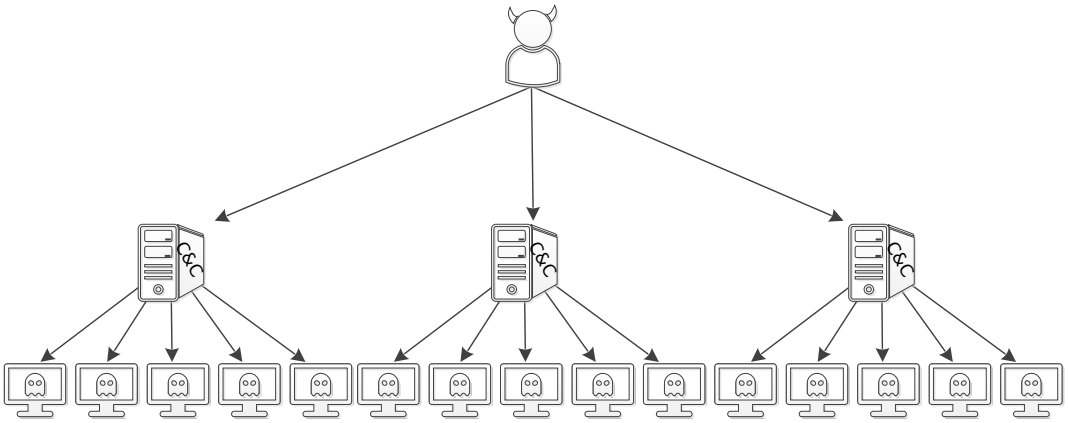
Dosegljivost velikega števila strežnikov na posameznem nivoju lahko dosežemo z uporabo hitro spreminjajočih se (*fast-flux*) vnosov DNS. Na strežniku DNS je pod enim domenskim naslovom lahko registriranih mnogo naslovov IP. Če ima tak vnos DNS nastavljen zelo kratek čas poteka, ga drugi strežniki DNS ne shranjujejo v svoj predpomnilnik, kar omogoča hitro menjavo naslovov IP strežnikov C&C, s tem da lahko boti za povezovanje uporabljajo enako domensko ime.

2.3.2 Decentralizirana arhitektura (P2P)

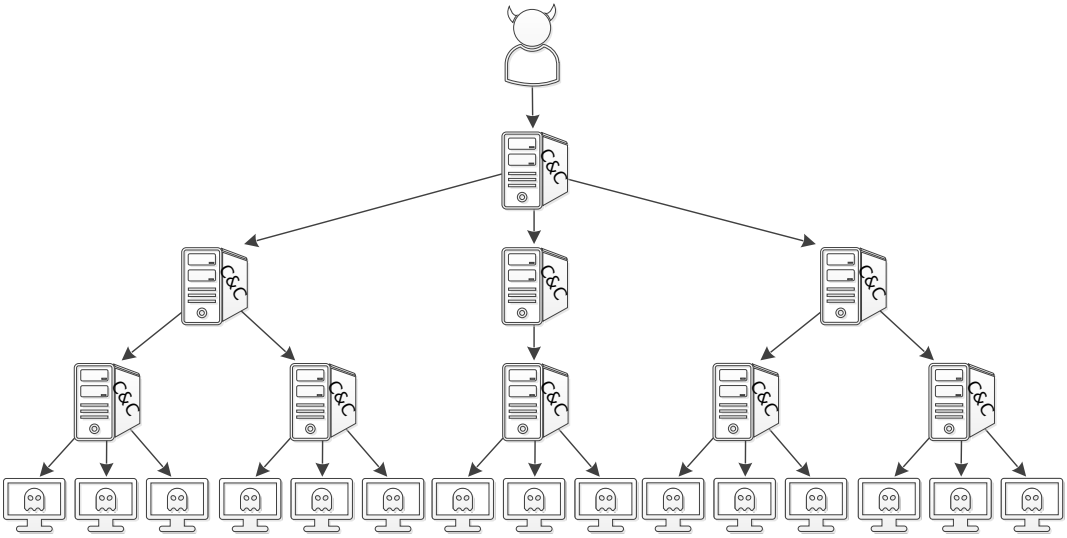
Pri uporabi decentralizirane arhitekture se boti povezujejo med seboj in botnet ne potrebuje centralnega strežnika C&C. Model decentralizirane arhitekture je prikazan na sliki 2.4. Sporočila in ukazi botom se propagirajo med njimi, pri čemer gospodar botneta deluje kot eden od botov, kar prinaša še večjo anonimnost ter težjo detekcijo, ker ni vozlišč, kamor bi se povezovalo veliko število botov. Izboljšana robustnost v primerjavi s centralizirano arhitekturo je posledica dejstva, da lahko botnet nadaljuje z delovanjem tudi po



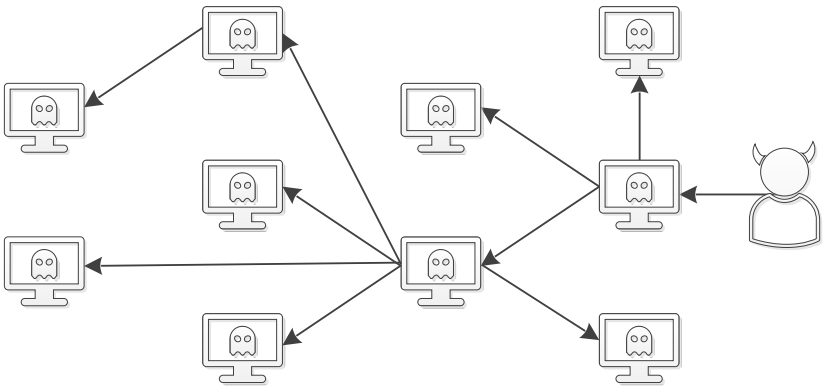
Slika 2.1: Centralizirana arhitektura botneta.[a]



Slika 2.2: Arhitektura botneta z več strežniki C&C.[a]



Slika 2.3: Hierarhična arhitektura botneta z več strežniki C&C.[a]



Slika 2.4: Decentralizirana (P2P) arhitektura botneta.[a]

[a]Posamezne ikone na slikah so intelektualna last Freepik, `www.freepik.com`.

izgubi večjega števila botov. Decentralizirani modeli temeljijo na protokolih P2P kot sta BitTorrent in Kademlia.

Povezovanje med boti poteka, podobno kot pri hierarhični ureditvi, tako, da so tisti boti, do katerih lahko drugi vzpostavijo povezavo, uporabljeni kot vozlišča. Način, kako bot najde druga vozlišča in s tem vzpostavi povezavo z omrežjem P2P, imenujemo začetno povezovanje (angl. *bootstraping*), in je predpisan z uporabljenim protokolom P2P. Silva et al. [15] navajajo naslednje načine:

1. Seznam nekaterih aktivnih vozlišč je generiran med procesom širjenja botneta in zapisan v programski kodi novega bota. Ob začetnem povezovanju bot poskuša vzpostaviti povezavo z vsakim od njih.

2. Seznam delujočih vozlišč je vsem botom dosegljiv na skupni spletni lokaciji. Tak način uporablja omrežje Gnutella.

3. Botnet vsebuje tako imenovana supervozlišča, katerih naslovi se redko spreminjajo in so zapisani v programski kodi botov. Po vzpostavitvi povezave z njimi, supervozlišča posredujejo botom seznam ostalih vozlišč. Tak način uporablja botnet Nugache, katerega koda vsebuje seznam 22 supervozlišč. Če bi bila vsa ta vozlišča onemogočena, se novi boti ne bi mogli povezati v omrežje.

4. Poseben primer je začetnik decentraliziranih botnetov, Sinit, ki namesto uporabe seznama vozlišča išče z naključnim preiskovanjem omrežja. Kot ugotavljata Wang in Ramsbrock [7], ima zato težave pri povezovanju in je lažje detektiran zaradi velike količine omrežnih podatkov med preiskovanjem.

Za naslove supervozlišč pri tretji ali seznama vozlišč pri drugi opciji se lahko, namesto naslovov IP, uporabljajo tako imenovani *fast-flux* vnosi DNS, omenjeni v zadnjem odstavku podpoglavja 2.3.1.

2.4 Življenjski cikel bota

Za boljše razumevanje problemov pri detekciji botnetov, je koristno, da si pogledamo osnovne stopnje v življenjskem ciklu posameznega bota. To so:

1. **Okužba**
   Gostiteljski sistem se lahko okuži na različne načine, primeri teh so [7]:
   - Ranljivosti v programski opremi: napadalec izrabi ranljivost v storitvi, ki teče na sistemu, za pridobitev dostopa in namestitev zlonamernega programa.
   - Prenos prek spletne strani: programska koda na spletni strani izkoristi ranljivost spletnega brskalnika in, brez uporabnikovega vedenja, namesti programsko opremo. Nevarna spletna stran je lahko napadalčeva, postavljena le za ta namen, pogosto pa se nevarna koda pojavlja na kompromitiranih spletnih strežnikih, ki sicer gostujejo legitimne strani.
   - Trojanski konji: to je programska oprema, ki izgleda legitimna in si jo uporabnik namesti po lastni volji. Poleg (ali namesto) legitimne funkcionalnosti se na sistem namesti še zlonameren program.
   - Neželena pošta
   - Obstoječi zlonamerni programi
2. **Javljanje**
   V začetni fazi delovanja se mora bot javiti centralnemu strežniku C&C (v primeru da gre za centralizirano arhitekturo) ali izvesti določen protokol za vzpostavitev povezave z drugimi boti (pri arhitekturi P2P, vsak z vsakim). Bot lahko pridobi posodobitve programske kode ali naslove strežnikov C&C, lahko pa že prejme prve ukaze. V tem koraku lahko program izvede tudi ukrepe za skrivanje pred uporabnikom in protivirusnimi programi. Ta faza se lahko ponavlja, na primer vsakič, ko se gostiteljski računalnik zažene. Med komunikacijo v tej fazi imamo možnost detekcije bota prek prepoznave vzorcev omrežnega prometa.

3. **Prevzem ukazov**
   Med čakanjem na ukaze je aktivnost bota majhna, prav tako pa je majhna količina podatkov, prenesenih po komunikacijskem kanalu, zato je v tej fazi detekcija težja. Generiran omrežni promet bota se razlikuje glede na tip komunikacijskega kanala, ki ga botnet uporablja. Če je to IRC, bodo med čakanjem na ukaze preneseni le občasni paketi za ohranjanje povezave. Če botnet uporablja protokol HTTP, pa periodično sprašuje strežnik za morebiten ukaz.

4. **Izvajanje ukazov**
   Odvisno od tipa aktivnosti, ki jo botnet izvaja, se obnašanje v tej fazi lahko močno razlikuje, navadno pa je izmenjava sporočil s strežnikom pogostejša kot v drugih fazah; med drugim se lahko po omrežju prenašajo podatki, pridobljeni iz žrtvinega računalnika, elektronska pošta, ki jo pošilja bot, ali velika količina paketov, potrebnih za izvajanje DDoS napada. Glede na aktivnost se razlikuje tudi težavnost detekcije.

2.5 Metode detekcije

Metode detekcije botnetov glede na mesto postavitve detektorja delimo na:

(a) detekcijo na nivoju naprave,

(b) detekcijo na nivoju omrežja,

ter glede na tip detekcije na:

(i) detekcijo na podlagi znanih vzorcev,

(ii) detekcijo na podlagi anomalij.

Poznamo vse štiri kombinacije naštetega. Tradicionalen pristop k detekciji botnetov je enak kot pri zaznavanju ostalih zlonamernih programov, to so protivirusni programi na računalnikih. Ti v datotekah iščejo znane podpise zlonamerne kode ter spremljajo nekatere metrike obnašanja programov,

na primer uporabo sistemskih klicev, in jih primerjajo z znanimi nevarnimi vzorci. To je primer zaznavanja vzorcev na nivoju naprave.

Slabost detekcije na nivoju naprave se pokaže pri večjih omrežjih v poslovnem okolju, kjer je potrebno programsko opremo za detekcijo namestiti in posodabljati na vseh računalnikih v omrežju. Čeprav so ti postopki v visoki meri avtomatizirani, je težko zagotoviti, da bodo vse naprave imele ustrezen detektor zlonamerne programske opreme. Za tako programsko opremo je navadno potrebno kupiti licence glede na število namestitev, kar lahko pomeni velik strošek, grožnjo pa lahko predstavljajo prenosni računalniki, ki jih zaposleni prinesejo s seboj. Detekcija na nivoju omrežja odpravlja te slabosti, ker detektor nadzira celotno omrežje ter ne potrebuje posebne programske opreme na posameznih napravah.

Detekcija na podlagi anomalij nadzira metrike delovanja sistema in, preko hevrističnih pravil, zaznava odstopanja od običajnega delovanja. Detektorji anomalij pogosto uporabljajo metode umetne inteligence. Prednost pred zaznavanjem znanih vzorcev je v tem, da lahko detekcija anomalij zazna tudi še nepoznane grožnje, detektor znanih vzorcev pa mora vzorec prepoznati med že poznanimi, zato je pri teh pomembno pogosto posodabljanje baze znanja. Detekciji znanih vzorcev se lahko zlonameren program izogne s tehnikami za spreminjanje kode ali s šifriranjem svojega omrežnega prometa. Zlonamerne programe, ki pri tem niso uspešni, pa zaznavanje znanih vzorcev zazna z visoko preciznostjo, ima torej malo lažnih pozitivnih primerov. Slabost zaznavanja na podlagi anomalij pa je v višjem deležu lažnih pozitivnih primerov (in zato nižji preciznosti), ker se lahko, odvisno od metrik, ki jih detektor spremlja, legitimni programi obnašajo podobno zlonamernim. Dodatna prednost zaznavanja znanih vzorcev je v možnosti zaznave zlonamerne kode še preden se ta požene. To lahko dosežemo z nadziranjem datotek (na posameznem računalniku ali na omrežju, če ne gre za šifriran promet) in s tem preprečimo varnostno grožnjo. Zaznavanje anomalij nam tega ne omogoča, ker pred zagonom zlonamernega programa anomalije ne nastanejo. Zato menimo, da je za najboljšo zaščito potrebna kombinacija obeh rešitev.

Primer zaznave vzorcev na omrežju sta odprtokodna sistema za detekcijo vdorov (IDS[5]) Snort [18] in Suricata [19]. Oba programa sta brezplačna, vendar za pravilno delovanje potrebujeta še bazo znanja — pravila. Tudi osnovna pravila lahko pridobimo brezplačno, obsežnejše baze znanja pa so plačljive. Pravila lahko vsebujejo naslove IP znanih strežnikov C&C, naslove URL znanih nevarnih virov, zgoščene vrednosti nevarnih datotek itn. Če detektor v omrežnem prometu zazna nevarnost, lahko, odvisno od konfiguracije, prepreči nevaren promet, ali pa uporabnike zgolj obvesti.

Za namen te magistrske naloge smo razvili dva sistema detekcije botnetov; detektor botnetov na nivoju omrežja, ki deluje na podlagi zaznave anomalij, ter detektor zlonamerne programske opreme na mobilnih napravah platforme Android, ki deluje na podlagi zaznave vzorcev na nivoju naprave. V zadnjem času je najaktivnejša raziskovalna tematika na področju botnetov detekcija na podlagi anomalij na omrežju.

[5] *Intrusion detection system*

Poglavje 3

Botneti na mobilnih napravah

V zadnjih nekaj letih se je razmahnila uporaba in možnosti uporabe pametnih mobilnih telefonov, s tem pa tudi zlonamerna programska oprema na teh napravah. Kaspersky Labs [12] poročajo o več kot desetkratnem povečanju zaznav primerkov zlonamernih programov na mobilnih platformah v času od začetka leta 2012 do konca leta 2014. Mobilne platforme so zaradi mnogih komunikacijskih zmožnosti zanimive tudi za avtorje botnetov. Prve mobilne botnete so odkrili v letu 2010 [20], med vsemi zlonamernimi programi na mobilnih napravah pa naj bi bilo botnetov približno 60% [21].

Mobilne naprave avtorjem zlonamerne programske opreme omogočajo dodatne možnosti zlorab, imajo pa tudi nekatere omejitve. Omejena računska moč in poraba energije v primerjavi z osebnimi računalniki pomenita omejitve tako za napadalce kot tudi za detekcijo botnetov. Čeprav napadalcev poraba energije najbrž ne skrbi neposredno, pa bi bili zlonamerni programi z veliko porabo energije enostavneje zaznani.

V nasprotju z osebnimi računalniki so mobilni telefoni praviloma prižgani vedno, kar gospodarjem botnetov omogoča večjo dosegljivost botneta.

3.1 Motivacija napadalcev

Motivacija avtorjev zlonamernih programov za mobilne naprave je pravzaprav enaka kot pri osebnih računalnikih — finančna korist. Metode oziroma poti do cilja pa se nekoliko razlikujejo.

Plačljive storitve SMS

Najbolj pogost način kraje denarja uporabnikov je pošiljanje sporočil SMS na plačljive številke [12]. Zlonamerni programi brez uporabnikove vednosti pošiljajo sporočila SMS na tako imenovane premium številke, ki so v lasti napadalcev. Ponudniki telefonskih storitev zaračunajo uporabnikom ter izplačajo denar napadalcem za naročnino na premium SMS storitve.

Kraja osebnih podatkov

Na pametnih mobilnih telefonih shranjujemo mnogo osebnih podatkov. Poleg kontaktnih podatkov uporabnika hranimo tudi podatke vseh stikov v imeniku, ki so dostopni lažje kot na osebnih računalnikih, ker so na vseh napravah shranjeni na istem mestu. Prav tako so lahko dostopna e-poštna sporočila in sporočila SMS ter kamera in mikrofon, ki omogočata zvočni in slikovni nadzor uporabnika brez njegove vednosti. Prenosne naprave beležijo tudi lokacijo, katere nadzor lahko napadalcem pomaga celo pri fizičnih napadih in ropih.

Zloraba dvofaktorskega overjanja

Dvofaktorsko overjanje je varnostna rešitev, ki jo uporabljajo mnoge spletne storitve. Za drugi faktor overitve so pogosto uporabljena sporočila SMS. Pri prijavi, po vnosu uporabniškega imena in gesla, uporabnik prejme še sporočilo SMS z enkratno naključno kodo, ki jo vnese v spletni vmesnik za uspešno overitev. Zlonamerni programi na mobilnem telefonu lahko prestrežejo sporočilo ter kodo za overjanje posredujejo napadalcem.

Zloraba mobilnega bančništva

V približno zadnjih dveh letih je postalo mobilno bančništvo zelo priljubljeno. Vse večje banke uporabnikom ponujajo mobilne aplikacije ali SMS storitve za upravljanje s financami. Skupaj s tem so se razvili zlonamerni programi, ki to izkoriščajo. Po poročilih Kaspersky Labs [12] se je število zaznanih tovrstnih groženj od začetka leta 2013 do oktobra 2014 povečalo iz manj kot 100 na kar 13000. Najbolj enostaven način zlorab je preko bančnih SMS storitev, kjer je že sámo sporočilo, poslano iz uporabnikovega telefona, dovolj za njegovo overjanje. Mnogo bank uporablja varnostno rešitev dvofaktorskega overjanja, kar lahko napadalci izkoristijo, kot je opisano v prejšnjem odstavku. Veliko nevarnost uporabnikom mobilnih bančniških aplikacij predstavljajo tudi zlonamerni programi, ki posnemajo uporabniški vmesnik uradne bančne aplikacije. Zlonamerna aplikacija se prikaže namesto legitimne in tako uporabnik vanjo vnese podatke za overjanje, ki so nato posredovani napadalcem.

Poleg naštetih so priljubljeni načini zlorab na mobilnih napravah tudi vsiljivo prikazovanje oglasov, izsiljevalski programi ter pošiljanje neželenih sporočil, kar pa se bistveno ne razlikuje od ekvivalentov na osebnih računalnikih.

3.2 Komunikacijske možnosti

Ena glavnih razlik med botneti na osebnih računalnikih in na mobilnih napravah je razpoložljivost različnih komunikacijskih kanalov. Tipično imajo mobilni telefoni dostop do interneta, vendar je ta povezljivost lahko motena, če uporabniki nimajo vedno vklopljene internetne povezave ali se gibljejo med brezžičnimi omrežji.

Dodaten komunikacijski kanal, ki je med mobilnimi botneti zelo priljubljen, so sporočila SMS. Botneti jih uporabljajo kot kanal C&C in kot način zlorabe (opisan v poglavju 3.1). Do sedaj odkriti botneti, ki sporočila SMS

uporabljajo za komunikacijo z ukazno-nadzornim strežnikom, uporabljajo le centralizirano arhitekturo, Zeng et al. [22] pa so predstavili študijo razvoja decentraliziranega (P2P) botneta, delujočega prek sporočil SMS.

Še en komunikacijski kanal, značilen za mobilne naprave, so storitve za potisna obvestila. Primer take storitve je *Google Cloud Messaging* (GCM), ki omogoča potisno pošiljanje sporočil iz strežnika na mobilne naprave.

Gre za posredovanje sporočil preko Googlove storitve, zato ni direktne povezave med ukazno-nadzornim strežnikom in botom. Operacijski sistemi mobilnih naprav združujejo pakete potisnih sporočil vseh nameščenih programov, da bi prihranili na energiji. Kot ugotavljajo Chen et al. [23], to onemogoča detekcijo mobilnih botnetov preko ukaznih sporočil GCM, tako na podlagi zaznave anomalij kot tudi prepoznave vzorcev.

Najbolj razširjen komunikacijski kanal pri mobilnih botnetih je sicer HTTP, sledi SMS, poročil o uporabi omrežja IRC za namen mobilnih botnetov pa nismo zasledili. Vsi znani mobilni botneti uporabljajo centralizirano arhitekturo C&C, kar omogoča enostavnejšo blokado delovanja botneta po odkritju strežnika C&C [20].

3.3 Načini okužbe

V nasprotju z botneti na osebnih računalnikih, kjer okužba z zlonamerno programsko opremo največkrat poteka brez uporabnikove vpletenosti, na mobilnih platformah skorajda ni varnostnih ranljivosti, ki bi omogočale okužbo brez uporabnikove potrditve [12]. Zato morajo avtorji zlonamernih programov uporabnike pripraviti do namestitve njihovih programov. Prevladujoč način okužbe mobilnih naprav so zato trojanski konji.

Oba najbolj razširjena operacijska sistema pametnih mobilnih telefonov, Googlov Android in Applov iOS, imata svoji uradni trgovini programske opreme. Kot poroča Kaspersky Lab [12], se v uradnih trgovinah zlonamerni programi pojavljajo redko, če že, pa so hitro odstranjeni. Večji del nevarnih aplikacij uporabniki prenesejo iz drugih, neuradnih, trgovin, kjer aplikacije

redkeje preverjajo za vsebnost zlonamerne kode. Iz teh trgovin uporabniki prenašajo programsko opremo zaradi dostopnosti nekaterih aplikacij, ki v uradnih trgovinah niso na voljo, ker jih tja vodijo spletne povezave, ali ker v njihovih državah uradne trgovine niso dosegljive — v nekaterih državah Googlova Play Store trgovina ne ponuja plačljivih aplikacij, na Kitajskem pa sploh ni na voljo.

Zlonamerni programi se prenašajo tudi iz raznih spletnih strani, ki ponujajo trojanske konje v obliki posodobitev programske opreme, protivirusnih programov ter raznih orodij in iger. Za njihovo gostovanje lahko napadalci izkoristijo kompromitirane legitimne spletne strani ali, tipično s pomočjo neželene e-pošte ali sporočil SMS, vabijo na lastne spletne strani z zlonamernimi aplikacijami.

3.4 Metode detekcije

Metode detekcije zlonamernih programov se zaradi specifik posamezne mobilne platforme med njimi razlikujejo. Daleč najbolj popularen mobilni operacijski sistem za avtorje zlonamernih programov je Googlov Android: Kaspersky Lab [21] poroča, da je bilo med vsemi zaznanimi zlonamernimi mobilnimi aplikacijami v letu 2013 kar 98% primerov na sistemu Android. Zaradi tega bomo študijo detekcije mobilnih zlonamernih programov v tej nalogi omejili na to platformo.

3.4.1 Varnostni mehanizmi operacijskega sistema Android

Varnost v operacijskem sistemu Android temelji na podeljevanju pravic posameznim aplikacijam. Te pravice dovoljujejo aplikaciji dostop do posameznih komponent naprave oziroma operacijskega sistema, na primer: dostop do interneta, dostop do kamere, branje ali pisanje datotek iz spominske kartice, uporaba Bluetooth vmesnika, branje imenika stikov, branje sporočil SMS,

pošiljanje sporočil SMS, ... Seznam vseh definiranih pravic je dostopen v [24]. Podeljevanje teh pravic poteka tako, da avtor aplikacije navede želene pravice, ob namestitvi pa jih uporabnik bodisi dovoli, bodisi ne dovoli, kar pomeni, da aplikacije ne more namestiti. Slabost je, da uporabnik aplikaciji ne more zavrniti posamezne pravice in mora sprejeti vse, če želi program uporabljati. Večina uporabnikov zato sprejme vse zahteve aplikacije brez (natančnega) pregleda.

Ne glede na nastavljene pravice operacijski sistem Android preprečuje aplikacijam dostop do sistemskih datotek in shranjenih podatkov drugih aplikacij. To pomeni, da uporabnik naprave nima administratorskih pravic. Pridobitev administratorskih pravic je možna s posebnimi nastavitvami (odvisno od proizvajalca naprave) ali z izkoriščanjem nekaterih varnostnih ranljivosti v sistemu, za kar so na voljo neuradna orodja. Uporabniki to izkoristijo, če želijo dodatno prilagajati svoje naprave. S stališča varnosti pridobivanje administratorskih pravic ni priporočljivo, ker pomeni, da lahko administratorski dostop uporabijo tudi nameščene aplikacije, ki utegnejo vsebovati zlonamerno kodo.

Dodatna varnostna funkcija sistema Android so certifikati aplikacij. Ti so uporabljeni predvsem v postopku posodabljanja aplikacij. Nameščena aplikacija je lahko posodobljena z novejšo verzijo, pri čemer ohrani svoje podatke na napravi. Do varnostne grožnje bi prišlo, če bi lahko neka aplikacija, ki se izdaja za posodobitev druge, že nameščene aplikacije, le-to nadomestila, s čimer bi pridobila dostop do shranjenih podatkov in dodeljenih pravic prejšnje aplikacije. Namestitvena datoteka vsake aplikacije mora biti podpisana z digitalnim certifikatom avtorja, sistem Android pa dovoli posodobitve le, če je nova verzija aplikacije podpisana z istim certifikatom kot že nameščena.

Poleg posodobitev se certifikati uporabljajo tudi pri aplikacijah, ki ponujajo svoje funkcije drugim aplikacijam. Aplikacija lahko uporabi ponujeno funkcijo, če je podpisana z enakim certifikatom ali če imata certifikata obeh aplikacij skupnega izdajatelja. Pravice, ki jih zahtevajo deljene funkcional-

nosti, so lahko definirane le v aplikaciji, ki funkcije ponuja. To pomeni, da aplikacija, ki je s certifikatom pooblaščena uporabljati deljeno funkcionalnost druge aplikacije, prevzame njene pravice brez dodatne odobritve uporabnika.

3.4.2 Obstoječe metode detekcije

Znani obstoječi programi za detekcijo zlonamernih programov na operacijskem sistemu Android se zanašajo predvsem na detekcijo znanih vzorcev. Na ta način detektorji prepoznavajo zlonamerne aplikacije s pomočjo primerjanja delov programske kode z znanimi zlonamernimi primeri. Detektorji so navadno del varnostnega programa, ki uporabnike opozarja tudi na sistemske nastavitve, ki bi utegnile ogroziti varnost naprave ter na nameščene aplikacije, ki imajo dodeljene potencialno nevarne pravice. Ti programi ponujajo tudi blokiranje neželenih klicev in sporočil iz izbranih številk ter razna orodja za fizično zaščito naprave, na primer zaklepanje, brisanje ali sporočanje lokacije naprave v primeru kraje. Poleg splošnih varnostnih rešitev pa obstajajo tudi specializirane aplikacije za detekcijo izkoriščanja posameznih znanih sistemskih ranljivosti.

V literaturi smo zasledili več predlogov in opisov prototipov naprednejših detektorjev, ki uporabljajo zaznavanje na podlagi anomalij, vendar teh v komercialno razširjenih rešitvah nismo zasledili. Predlagane metode vključujejo sledenje uporabe sistemskih klicev in virov ter odkrivanje anomalij s primerjanjem uporabe le-teh z običajno uporabo. Zaznavanje uporabe funkcij operacijskega sistema in nekaterih drugih virov, prav tako pa dostop do datotek nameščenih aplikacij, nam operacijski sistem preprečuje zaradi že omenjene omejitve administratorskega dostopa, kar otežuje tehnike detekcije, ki se opirajo na te vire.

3.5 Varnostne ranljivosti operacijskega sistema Android

Osnovna varnostna mehanizma operacijskega sistema Android sta omejitev pravic aplikacijam ter certificiranje aplikacij. V nadaljevanju bomo predstavili sistem za preverjanje certifikatov aplikacij in dve vrsti ranljivosti v tem sistemu, katerih izkoriščanje lahko z našo aplikacijo zaznamo.

Preverjanje certifikatov aplikacij se izvaja ob namestitvi vsake aplikacije. Aplikacije Android shranjujemo v paketnih datotekah APK, ki so zapisane v formatu ZIP. Vsebujejo programsko kodo in vse druge vire, potrebne za delovanje aplikacije, poleg tega pa vsebujejo tudi certifikat izdajatelja ter podatke, potrebne za preverjanje njegove veljavnosti.

Ob namestitvi aplikacije operacijski sistem preveri, če posamezne datoteke ustrezajo njihovim zgoščenim vrednostim in če podpis le-teh ustreza certifikatu, ter v nasprotnem primeru zavrne namestitev. Če je aplikacija z enakim imenom[1] že nameščena na napravi, bo izvedena posodobitev obstoječe aplikacije. To pomeni, da se shranjeni podatki aplikacije ohranijo, uporabniku pa ni potrebno ponovno odobriti zahtevanih pravic. Posodobitev aplikacije sistem odobri le, če je nova verzija podpisana z enakim certifikatom.

Če zlonamerna aplikacija, ki se izdaja za posodobitev obstoječe, uspe prelisičiti varnostni sistem za preverjanje certifikatov, lahko brez ponovne potrditve uporabnika pridobi dostop do pravic in shranjenih datotek, ki pripadajo prej nameščeni aplikaciji. Pri tem lahko zlonamerna aplikacija pridobi tudi posebne pravice, ki pripadajo le sistemskim aplikacijam, če se izdaja za aplikacijo, ki te pravice ima. Te posebne pravice vključujejo spreminjanje sistemskih nastavitev, brisanje vseh podatkov na napravi in nameščanje novih aplikacij, dodeljene pa so le tovarniško nameščenim aplikacijam.

Prednost širjenja zlonamerne kode preko posodobitev ima tudi psihološko prednost. Uporabnika je lažje prepričati v namestitev posodobitve za aplikacijo, ki ji zaupa, kot v namestitev nove aplikacije. Napadalci v ta namen

[1] Uporablja se polno ime aplikacije (*package name*), ki je neponovljivo.

navadno uporabljajo spletne oglase, ki ponujajo posodobitve že razširjenih aplikacij.

3.5.1 Ranljivost Master Key

Ranljivost Master Key je skupno ime za tri hrošče v programski kodi operacijskega sistema Android. Ranljivost izkorišča dejstvo, da preverjanje ujemanja datotek z njihovimi podpisi izvaja drug programski modul operacijskega sistema kot dejansko namestitev aplikacije. Z izrabo teh hroščev pride do različne obravnave posameznih datotek znotraj paketa APK s strani obeh modulov.

Format datoteke APK

Za razumevanje nadaljevanja si moramo najprej pogledati sestavo datotek formata ZIP (kakršne so tudi aplikacijske datoteke APK). Format ZIP bomo opisali le z natančnostjo, potrebno za razumevanje ranljivosti, podrobnosti pa bralec lahko najde v [25].

Paket ZIP je sestavljen iz množice datotek, ki so bodisi shranjene v originalnem zapisu, bodisi stisnjene. Posamezne datoteke so znotraj paketa ZIP zapisane ena za drugo, pred podatki vsake datoteke pa se nahaja lokalna glava datoteke. Lokalna glava vsebuje metapodatke o datoteki, kot so: oznaka algoritma, uporabljenega za stiskanje, velikost datoteke, kontrolna vsota CRC, ime datoteke in dodatno polje, uporabljeno za poljubne podatke. Centralni imenik paketa ZIP se nahaja na koncu datoteke ZIP in vsebuje centralne glave datotek. Centralna glava datoteke, podobno kot lokalna, vsebuje podatek o velikosti datoteke in ime datoteke, označeno pa ima tudi lokacijo lokalne glave datoteke znotraj paketa ZIP. Za centralnimi glavami datotek se nahaja zapis za konec centralnega imenika, ki vsebuje polje z njegovo dolžino.

Datoteka ZIP nima glave na začetku, zato moramo najprej z branjem od konca proti začetku datoteke prebrati dolžino centralnega imenika, da lahko preberemo centralne glave datotek, te pa označujejo lokacije lokalnih glav, katere potrebujemo za branje posameznih datotek.

Podatki za preverjanje ustreznosti certifikata izdajatelja so znotraj paketa APK zapisani v treh datotekah v imeniku META-INF. Datoteka MANIFEST.MF vsebuje imena in pripadajoče zgoščene vrednosti vseh aplikacijskih datotek. V datoteki CERT.SF so zgoščene vrednosti skupkov imen in zgoščenih vrednosti iz datoteke MANIFEST.MF ter celotne datoteke MANIFEST.MF. Datoteka CERT.RSA pa vsebuje certifikat izdajatelja in podpis datoteke CERT.SF.

Podvojena imena datotek

Prvo ranljivost Master Key je odkril Jeff Forristal februarja 2013 [26], podroben opis pa najdemo tudi v [27]. Izkorišča razliko v obravnavi centralnega imenika datoteke APK v modulu za preverjanje podpisov in modulu za namestitev, če je v paketu APK datoteka z enakim imenom (v ime štejemo tudi imena map — celotno pot znotraj paketa) zapisana večkrat. Zapis več datotek z enakimi imeni in potmi sicer nima smisla, vendar nam format ZIP tega ne preprečuje.

Program, ki preverja podpise, je napisan v Javi, za shranjevanje centralnih glav datotek pa uporablja zgoščeno tabelo (objekt razreda `HashMap`), kjer je kot ključ uporabljeno ime ciljne datoteke. Če pri branju centralnega imenika program naleti na več datotek z enakim imenom, se v zgoščeni tabeli vrednost z enakim ključem prepiše z novo vrednostjo. Pri preverjanju podpisov program iterira preko centralnih glav datotek v zgoščeni tabeli in za vsako od njih preveri ustreznost podpisa. Če imamo torej več datotek z enakim imenom, se preveri podpis le tiste datoteke, katere glava je zapisana zadnja po vrsti v centralnem imeniku.

Program, uporabljen za namestitev (kopiranje datotek iz paketa APK), pa uporablja lastno implementacijo zgoščene tabele (v jeziku C++). Ta ob vstavljanju preverja, če v zgoščeni tabeli že obstaja vnos z enakim ključem, in v tem primeru nov vnos shrani v novo polje. Ko program kopira datoteke iz paketa, pa z iskanjem po imenih datotek vedno vrne prvo zapisano glavo datoteke, torej se med datotekami z enakimi imeni upošteva prva zapisana v

centralnem imeniku.

Ranljivost lahko zlorabimo tako, da v legitimno podpisano datoteko APK, katere aplikacija je že nameščena na žrtvini napravi, dodamo poleg neke obstoječe datoteke še eno z istim imenom, s čimer zamenjamo uporabljeno vsebino datoteke brez da bi sistem zaznal spremembo podpisa. Z zamenjavo datoteke s programsko kodo lahko v aplikacijo podtaknemo zlonamerno kodo, na delovanje aplikacije pa lahko vplivamo tudi s spremembo drugih datotek znotraj paketa APK.

Negativna dolžina dodatnega polja

Druga ranljivost, ki je prav tako povezana z zgradbo datoteke APK, izkorišča dodatno polje (*extra*) v lokalnih glavah datotek. Podrobno jo opisuje Freeman v [28], odkrita pa je bila julija 2013. Tudi tu gre za razliko v implementaciji programa za preverjanje podpisov in programa za namestitev paketa APK. Lokalna glava datoteke vsebuje, poleg ostalih polj, dodatno polje, katerega namen s formatom ZIP ni predpisan. Za določanje pozicije začetka podatkov datoteke je pred dodatnim poljem še dvobajtno polje s podatkom o njegovi dolžini. Podatek o dolžini javanski program za preverjanje certifikata prebere z ukazom `readShort`, ki prebere število kot predznačeno, program za namestitev paketa pa bere dolžino polja kot nepredznačeno.

Ker se dodatno polje tipično ne uporablja (če že, pa je njegova dolžina manjša od 2^{15}), v normalnih okoliščinah programa delujeta pravilno. V kolikor pa nastavimo dolžino dodatnega polja na vsaj 2^{15}, pa se pojavi razlika med delovanjem programa za preverjanje podpisov in programa za nameščanje. Program za preverjanje podpisov vrednost interpretira kot negativno, program za nameščanje pa kot pozitivno. Ker lahko med datotekami v paketu ZIP pustimo prostor, lahko paket preoblikujemo tako, da vsebuje dve različici poljubne datoteke. Prav tako kot pri prejšnjem opisanem hrošču, se ena različica upošteva le za preverjanje podpisa, druga pa se uporabi pri namestitvi. Ranljivost lahko na enak način izkoristimo za nameščanje zlonamerne kode s pomočjo posodobitev legitimnih aplikacij.

Neskladje pri branju dolžine imen datotek

Še en hrošč, ki omogoča podobne zlorabe, je povezan z dolžino imena datoteke znotraj paketa APK. Opisan je v [29]. Dolžine imen so shranjene v centralnih in v lokalnih glavah datotek, uporabljene pa so za branje ali za preskok polja z imenom datoteke oziroma določanje pozicije začetka podatkov. Program za preverjanje certifikata datoteke si dolžine imen datotek zapomni iz branja centralnega imenika in jih pri branju lokalnih glav ne prebere ponovno. V nasprotju s tem pa program za nameščanje paketa APK bere dolžine imen datotek tudi v lokalnih glavah in na podlagi tega določi pozicijo začetka bloka s podatki datoteke.

Posledično spet lahko pridemo do razlike v branju datotek, če v polje z dolžino imena lokalne glave datoteke vpišemo nepravilno vrednost in na ustrezno mesto v paketu zapišemo želeno vsebino datoteke, ki jo bo operacijski sistem naložil in uporabil, vendar njenega podpisa ne bo preveril.

Trojka ranljivosti Master Key je bila prisotna vse od začetnih verzij operacijskega sistema Android do popravka prvih dveh hroščev v verziji 4.3, ki je izšla julija 2013. Ranljivost, povezana z neskladjem pri branju dolžine imen datotek je bila odpravljena v verziji 4.4. Po podatkih Googla [30] še v avgustu 2015 na kar 40% napravah z OS Android tečejo verzije operacijskega sistema, starejše od 4.3, kar pomeni, da imajo to ranljivost. V uradni trgovini ponujanje aplikacij, ki zlorabljajo omenjeno ranljivost, ni mogoče že od odkritja ranljivosti, prav tako pa ni bilo odkritih primerkov, ki bi ranljivost zlorabljali pred tem. V neuradnih trgovinah in drugih virih za prenos aplikacij pa le-te tipično niso preverjene in lahko vsebujejo zlorabe ranljivosti Master Key.

3.5.2 Ranljivost Fake ID

Kot smo omenili v poglavju 3.5, lahko aplikacija prevzame pravice druge aplikacije, če ta deli svoje funkcije in imata aplikaciji enak certifikat ali skupnega izdajatelja certifikata. V primeru skupnega izdajatelja mora biti v datoteki

s certifikati (CERT.RSA) poleg certifikata, s katerim so podpisane datoteke aplikacije, shranjena še celotna veriga certifikatov do certifikata izdajatelja. Forristal [31] je ugotovil, da program za nameščanje aplikacij nepravilno preverja pravilnost verige certifikatov. Namesto preverjanja kriptografskih podpisov program pogleda le, če se v verigi certifikatov ujemajo imena izdajateljev posameznih certifikatov.

Avtor zlonamerne aplikacije lahko ranljivost izkoristi tako, da ustvari neveljavno verigo certifikatov, kjer za vrhovni certifikat uporabi legitimen certifikat poljubne aplikacije, ki ponuja deljeno storitev, v svojem certifikatu, ki podpisuje zlonamerno aplikacijo, pa za izdajatelja navede ime legitimnega certifikata. Tako lahko zlonamerna aplikacija uporablja storitve legitimne aplikacije brez, da bi uporabnik moral potrditi dodatne pravice.

Forristal [31] je odkril tudi, da operacijski sistem Android omogoča posebne pravice aplikacijam, katerih certifikat je izdalo podjetje Adobe. Tem aplikacijam Android dovoljuje delovanje kot vtičnik v komponento WebView, ki skrbi za prikaz spletne vsebine znotraj aplikacij. To je omogočeno zaradi podpore vtičniku Adobe Flash. Posledično lahko aplikacija, ki se izdaja, da je njen certifikat podpisan s certifikatom Adoba, izvaja svojo programsko kodo znotraj vseh aplikacij, ki vključujejo komponento WebView, in tako uporablja njim dodeljene pravice.

Ta ranljivost je bila v sistemu Android prisotna od verzije 2.1, odpravili pa so jo z verzijo 4.4. V uradni trgovini Google ni našel aplikacij, ki bi izkoriščale ranljivost Fake ID.

Poglavje 4

Detektor botnetov iz podatkov omrežnega prometa

4.1 Cilji

Eden od ciljev te magistrske naloge je bil implementacija detektorja botnetov. Odločili smo se za detekcijo na podlagi anomalij iz podatkov omrežnega prometa. Kot jedro detektorja smo uporabili strojno učenje, in sicer metodo naključnih gozdov. Idejo za implementacijo detektorja smo dobili v članku [1], katerega smo opisali v poglavju 1.1. Klasifikator naključnih gozdov kot vhod uporablja značilke omrežnih tokov. To so značilnosti omrežnih tokov, ki ne obsegajo naslovov IP in vsebine (podatkov) omrežnih paketov, kar pomeni, da se detektor osredotoča le na način komunikacije. Poleg splošne detekcije ne glede na omrežne naslove vpletenih računalnikov, to pripomore tudi k varovanju osebnih podatkov uporabnikov.

Sistem detektorja botnetov sestoji iz več komponent. Za klasifikacijo omrežnih tokov skrbi klasifikator naključnih gozdov, opisan v poglavju 4.3. Njegov vhod so opisi omrežnih tokov, ki jih iz podatkov omrežnega prometa pridobiva orodje, katerega bomo opisali v poglavju 4.2. V poglavjih 4.4 in 4.5 pa bomo predstavili način in rezultate testiranja izdelanega detektorja botnetov.

4.2 Ekstrakcija omrežnih tokov

Cilj ekstrakcije omrežnih tokov je iz omrežnih paketov sestaviti opise omrežnih tokov, ki bodo v naslednji fazi sistema uporabljeni za klasifikacijo. Orodje, ki smo ga razvili v ta namen, smo poimenovali *Flow Extractor*. Za implementacijo smo uporabili programski jezik Java. V nadaljevanju si bomo pogledali zgradbo programa ter njegove funkcije.

4.2.1 Zgradba

Program Flow Extractor je sestavljen iz vstopnega razreda **Main**, razreda **FlowExtractor**, ki vsebuje glavno logiko programa ter razredov **Packet**, **Flow** in **AddressPair**, ki vsebujejo polja za predstavitev omrežnih paketov, omrežnih tokov, in parov omrežnih naslovov.

Za branje omrežnih paketov smo uporabili odprtokodno knjižnico jNetPcap [32], ki omogoča branje neposredno iz omrežnega vmesnika ali iz datoteke PCAP. Iz predstavitve knjižnice jNetPcap za vsak paket ustvarimo objekt razreda **Packet**, ki vsebuje vse za nas pomembne podatke:

- naslovni par tipa **AddressPair**,
- časovni žig zajema paketa,
- dolžino paketa,
- dolžino glav omrežnega (IP) in transportnega (TCP ali UDP) nivoja,
- sekvenčno številko paketa TCP,
- številko potrditve paketa TCP,
- vrednosti kontrolnih zastavic PSH[1], URG[2], FIN[3], ACK[4] v glavah pa-

[1]Zahteva, da prejemnikov operacijski sistem podatke takoj (brez predpomnenja) posreduje aplikaciji.

[2]Zahteva prioritetno obravnavo paketa.

[3]Označuje prekinitev povezave.

[4]Potrjuje sprejem paketa, označenega s številko potrditve.

ketov TCP.

Paketi so v tokove združeni na podlagi objektov **AddressPair**, ki vsebujejo:

- naslov IP 1,
- naslov IP 2,
- naslov vrat transportnega protokola 1,
- naslov vrat transportnega protokola 2,
- oznako transportnega protokola (TCP ali UDP),
- smer paketa.

Naslova IP ter naslova vrat niso shranjeni kot naslov pošiljatelja in naslovnika, ampak razporejeni v polja 1 oz. 2 glede na številsko velikost naslovov IP. Če je pošiljateljev naslov IP številsko manjši od naslovnikovega, sta naslovnikova naslova shranjena v poljih 1, sicer pa v poljih 2. Da lahko iz te predstavitve ugotovimo vir in ponor paketa, je shranjena tudi smer paketa. Taka predstavitev omogoča lažjo primerjavo naslovnih parov: naslovna para se obravnavata kot enaka, če so vsi podatki razen smeri paketa enaki. Vsi paketi, ki pripadajo skupnemu omrežnemu toku, imajo torej enake naslovne pare v predstavitvi **AddressPair**.

Pakete v tej predstavitvi sprejme metoda **acceptPacket** razreda **FlowExtractor**. Ta poskrbi za prištevanje paketa k obstoječemu omrežnemu toku ali za ustvarjanje novega toka. Po tej operaciji lahko podatke o prebranem paketu zavržemo. Za hiter dostop so omrežni tokovi shranjeni v zgoščeni tabeli, kjer je kot ključ uporabljen objekt razreda **AddressPair**. Predstavitev omrežnega toka (**Flow**) vsebuje naslednje podatke:

- smer prvega paketa v toku,
- časovni žig prvega paketa v toku,
- časovni žig zadnjega prejetega paketa v pozitivni smeri,

- časovni žig zadnjega prejetega paketa v negativni smeri,
- za vsako smer paketov:
  - števec paketov z zastavico PSH,
  - števec paketov z zastavico URG,
  - zastavica s sekvenčno številko paketa FIN,
- za vsako smer paketov hranimo minimum, maksimum, vsoto in vsoto kvadratov za:
  - dolžine paketov,
  - časovne razlike med paketi,
- kazalec na naslednji (kasneje posodobljen) tok,
- kazalec na prejšnji (prej posodobljen) tok.

Smer prvega paketa v toku hranimo zaradi uvrščanja nadaljnjih paketov — paketi, potujoči v enaki smeri kot prvi, so šteti v pozitivno smer (naprej), tisti, ki potujejo obratno, pa v negativno (nazaj). Kazalca na naslednji in prejšnji omrežni tok sta shranjena zaradi možnosti učinkovitega sprehoda po najstarejših tokovih, ki jih program periodično odstranjuje. Ostala polja v razredu **AddressPair** so uporabljena za izpis značilk omrežnega toka, ki so izhod programa *Flow Extractor*. V nadaljevanju bomo navedli atribute omrežnega toka, ki jih naš program izpisuje.

4.2.2 Značilke omrežnih tokov

Metapodatki (izvor in ponor sta enaka pošiljatelju in prejemniku prvega paketa):

srcip Izvorni naslov IP.

srcport Številka vrat transportnega nivoja izvora.

dstip Ponorni naslov IP.

dstport Številka vrat transportnega nivoja ponora.

Značilke, uporabljene za klasifikacijo:

proto Protokol transportnega nivoja (TCP ali UDP).

total_fpackets Število paketov v pozitivni smeri.

total_fvolume Skupna dolžina paketov v pozitivni smeri.

total_bpackets Število paketov v negativni smeri.

total_bvolume Skupna dolžina paketov v negativni smeri.

min_fpktl Najmanjša dolžina paketa v pozitivni smeri.

mean_fpktl Aritmetična sredina dolžin paketov v pozitivni smeri.

max_fpktl Najdaljša dolžina paketa v pozitivni smeri.

std_fpktl Standardni odklon dolžin paketov v pozitivni smeri.

min_bpktl Najmanjša dolžina paketa v negativni smeri.

mean_bpktl Aritmetična sredina dolžin paketov v negativni smeri.

max_bpktl Najdaljša dolžina paketa v negativni smeri.

std_bpktl Standardni odklon dolžin paketov v negativni smeri.

min_fiat Najmanjša časovna razlika med paketi v pozitivni smeri.[5]

mean_fiat Aritmetična sredina časovnih razlik med paketi v pozitivni smeri.

max_fiat Najdaljša časovna razlika med paketi v pozitivni smeri.

[5]Vrednosti vseh časovnih značilk so podane v milisekundah.

std_fiat	Standardni odklon časovnih razlik med paketi v pozitivni smeri.
min_biat	Najmanjša časovna razlika med paketi v negativni smeri.
mean_biat	Aritmetična sredina časovnih razlik med paketi v negativni smeri.
max_biat	Najdaljša časovna razlika med paketi v negativni smeri.
std_biat	Standardni odklon časovnih razlik med paketi v negativni smeri.
duration	Časovna dolžina celotnega toka (razlika v času zadnjega in prvega paketa).
fpsh_cnt	Število paketov z zastavico PSH v pozitivni smeri.
bpsh_cnt	Število paketov z zastavico URG v negativni smeri.
furg_cnt	Število paketov z zastavico PSH v pozitivni smeri.
burg_cnt	Število paketov z zastavico URG v negativni smeri.
total_fhlen	Skupna dolžina glav paketov v pozitivni smeri.
total_bhlen	Skupna dolžina glav paketov v negativni smeri.

Poleg naštetih značilk lahko, opcijsko, program izpisuje dodaten binaren parameter **p2p**, ki označuje, če sta izvorna in ponorna številka vrat transportnega nivoja obe večji ali enaki 1024. Večina standardnih internetnih storitev uporablja številke vrat, manjše od 1024, večina storitev, ki uporabljajo protokole P2P, pa višje. Katerakoli storitev seveda lahko poteka preko poljubnih vrat, še posebej to velja za nestandardne, kot je promet botnetov, vseeno pa nam parameter **p2p** služi kot enostaven indikator za verjetnost, da gre pri omrežnem toku za promet vsak z vsakim (P2P). Uporaba klasifikacijskega atributa **p2p** je naša dopolnitev originalnega sistema, opisanega v članku Singh et al. [1].

Značilki **proto** in **p2p** sta definirani kot diskretni (obe imata po dve možni vrednosti), vse ostale pa so zvezne.

4.2.3 Opis funkcionalnosti

Omrežni promet lahko pridobimo z zajemom na omrežni infrastrukturi (stikalu ali usmerjevalniku), kjer imamo dostop do prometa celotnega omrežja, ki ga želimo nadzirati. Kot vhod v naše orodje uporabimo tok omrežnih podatkov, če želimo analizirati vnaprej zajete omrežne podatke, pa lahko uporabimo tudi datoteko z omrežnim prometom formata PCAP. Med branjem posameznih omrežnih paketov program *Flow Extractor* sestavlja omrežne tokove ter njihove značilke izpisuje na standardni izhod, tako da za vsak tok na izhodu dobimo eno vrstico s podatki. Omrežni tok izpišemo, ko je zaključen in če je prejel vsaj en omrežni paket v vsaki smeri. Tok obravnavamo kot zaključen ob prejemu paketa z zastavico FIN in njegove potrditve ali če je 5 minut neaktiven (v tem času program ni dodal toku nobenega paketa). Ko je tok zaključen in izpisan, ga program zavrže, ob morebitnem kasnejšem prejemu paketa, ki bi spadal v ta tok, pa se ustvari nov omrežni tok.

4.2.4 Parametri programa

-i Oznaka omrežne kartice s katere želimo zajemati promet.

-f Ime vhodne datoteke formata PCAP (se izključuje s prejšnjim parametrom).

-oneWayFlows S tem parametrom omogočimo izpis omrežnih tokov, tudi če niso zabeležili vsaj enega paketa v vsaki smeri.

-maxFlowDuration S tem parametrom lahko nastavimo najdaljše trajanje omrežnega toka, preden je ta izpisan. S tem dosežemo obravnavo omrežnega toka v klasifikaciji še preden se konča, kar je koristno pri dolgo trajajočih tokovih.

-p2p S tem parametrom dosežemo izpis dodatne značilke **p2p**.

4.3 Klasifikacija omrežnih tokov

Klasifikacija omrežnih tokov predstavlja jedro našega detektorja. Cilj klasifikatorja je uvrščanje omrežnih tokov med varne (benigne) oziroma nevarne (maligne), kjer maligni omrežni tokovi predstavljajo promet botnetov. Za implementacijo klasifikatorja smo uporabili programski jezik Python s knjižnico Orange [33] za podporo metodam strojnega učenja.

Naključni gozdovi

Za klasifikator smo uporabili metodo naključnih gozdov. Metoda ustvari množico odločitvenih dreves, vsako drevo pa nauči na vzorcu primerov, izbranih iz učne množice naključno z vračanjem. Število primerov v vzorcu za učenje posameznega drevesa je enako velikosti učne množice, kar pomeni, da se nekateri primeri ponovijo, nekateri primeri iz učne množice pa v vzorcu niso zajeti. Pri izbiri najboljšega atributa za delitev v vsakem vozlišču drevesa se upošteva le del vseh atributov. Množica atributov za izbiro najboljšega se vsakič izbere naključno, velikost te množice pa je kvadratni koren števila vseh atributov. Pri klasifikaciji posameznega primera se uporabijo vsa drevesa, primer pa se klasificira v razred, v katerega ga uvrsti največ dreves. Metoda naključnih gozdov generira bolj uspešne napovedne modele od običajnega odločitvenega drevesa, ker z večjim številom modelov težje pride do prevelikega prileganja učni množici.

Singh et al. [1] so ugotovili, da so naključni gozdovi v primerjavi z drugimi metodami strojnega učenja zelo primerni za klasifikacijo omrežnih tokov zaradi visoke natančnosti napovedi, zmožnosti zaznavanja raznolikih botnetov in računske učinkovitosti klasifikacije.

4.3.1 Učenje modela

Za klasifikacijo potrebujemo zgrajen model strojnega učenja. Tega pridobimo z učenjem na množici podatkov, ki vsebuje varne in nevarne omrežne tokove. Naš programski paket vsebuje skripto **randomForest.py**, ki sprejme učno

množico omrežnih tokov in nauči model klasifikatorja ter ga zapiše v datoteko formata pickle. Skripta za učenje modela sprejme naslednje parametre:

-i Ime datoteke z vhodnimi podatki (učno množico).

-n Število dreves, uporabljenih v algoritmu naključnih gozdov.

-x Maksimalna globina drevesa.

-y Minimalno število primerov v listu drevesa.

-o Ime izhodne datoteke.

V poglavju 4.5 bomo opisali, kakšne vrednosti parametrov so prinesle najboljše modele klasifikatorjev pri naši množici podatkov.

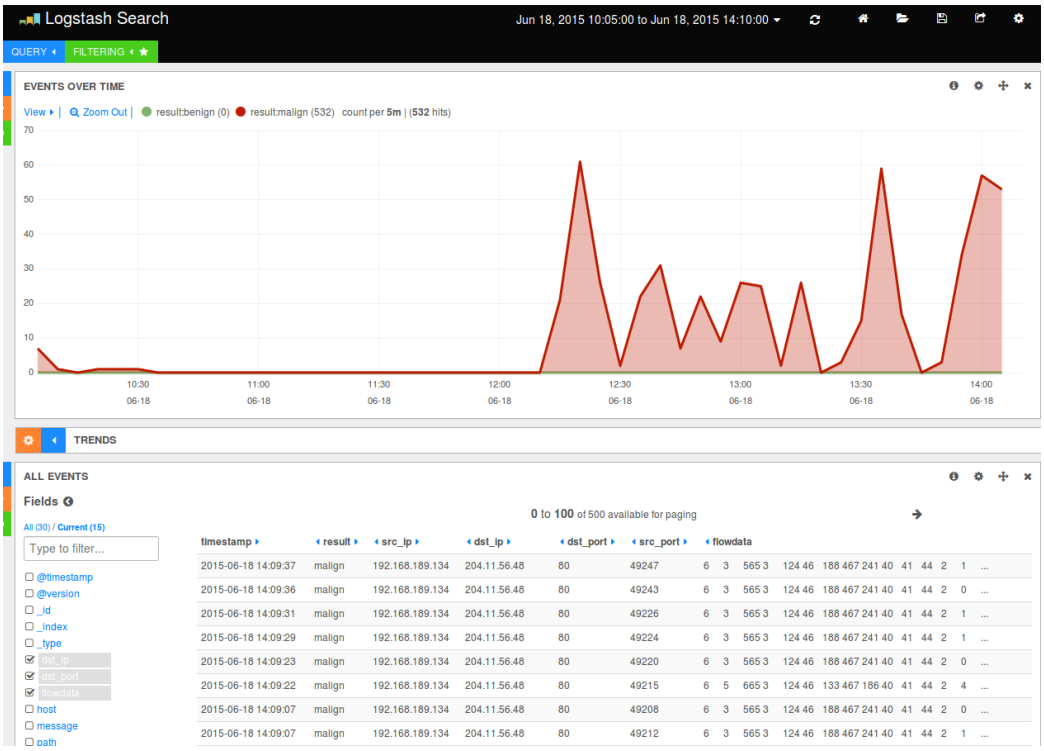
Pri učenju modela so uporabljene vse značilke omrežnih tokov, ki smo jih navedli v prejšnjem podpoglavju, razen naslovov IP in številk vrat, ki so uporabljeni zgolj kot metapodatki za identifikacijo omrežnega toka in lociranje zaznanega okuženega računalnika.

4.3.2 Klasifikacija

Skripta **streamClassify.py** skrbi za klasifikacijo omrežnih tokov z uporabo naučenega modela. Skripti podamo model strojnega učenja v datoteki formata pickle, nato pa na standardnem vhodu sprejema vrstice z opisom omrežnih tokov kot jih generira ekstrakcija tokov, ter na standardni izhod izpisuje rezultate klasifikacije. Poleg rezultata klasifikacije, ki ga označuje beseda **malign** za maligne ali **benign** za benigne tokove, je v posamezni vrstici izhoda še trenuten čas ter vsi podatki o omrežnem toku, ki ga je program sprejel.

4.3.3 Pregled rezultatov detekcije

Za lažji pregled izhoda klasifikacije in zato tudi detektiranih malignih omrežnih tokov, smo uporabili orodji Logstash in Kibana iz programskega paketa



Slika 4.1: Zaslonski posnetek pregledovalnika Kibana.

Elasticsearch [34] za grafični prikaz izhoda klasifikatorja. Logstash omogoča branje izhodne datoteke klasifikatorja, Kibana pa grafično prikazuje podatke v spletnem brskalniku. Zaslonski posnetek prikaza je viden na sliki 4.1. Na sliki so prikazani le zaznani maligni omrežni tokovi. V zgornjem delu vidimo graf, kjer posamezna točka grafa pomeni število zaznanih malignih omrežnih tokov v 5 minutni okolici časa, ki je prikazan na abscisni osi. V spodnjem delu zaslonskega prikaza pa je tabela s podatki o detektiranih nevarnih omrežnih tokovih. Prikaz omogoča nadzorniku omrežja, da opazi povečanje nevarnih tokov in identificira računalnik na omrežju, iz katerega izhajajo.

4.4 Opis podatkov

Za izdelavo in testiranje detektorja botnetov smo potrebovali množico malignih in benignih omrežnih tokov, za izdelavo te množice pa smo potrebovali

zajet benigen omrežni promet ter omrežni promet botnetov.

Da bi dobili splošen model, je koristno, da je učna množica raznolika, vsebovati mora torej legitimen promet različnih aplikacij ter primere prometa botnetov. Model, ki ga naučimo na taki učni množici, lahko uporabimo za detektorje v različnih omrežjih. Če bi uporabili učno množico, prilagojeno določenemu omrežju, z legitimnim prometom le tistih aplikacij, ki se v omrežju uporabljajo, bi dobili bolj specializiran detektor, ki pa bi bil zato bolj občutljiv na spremembe v omrežju (na primer uporabo novih aplikacij). Pri naših eksperimentih smo se osredotočili na učenje in testiranje splošnega modela klasifikatorja.

Primere prometa botnetov smo pridobili iz spletnega bloga Contagio Dump [35], kjer zbirajo zajeme prometa in programsko kodo zlonamerne programske opreme. Seznam pridobljenih omrežnih zajemov delovanja botnetov se nahaja v dodatku A. Avtorji primerkov zajemov ne navajajo, v kateri življenjski fazi (kot smo jih opisali v podpoglavju 2.4) so boti med zajemom delovali. Poleg omrežnih zajemov botnetov smo iz Contagio Dump pridobili tudi izvršljive datoteke botov, ki so generirali ta promet.

Zajeme benignega prometa smo naredili na našem domačem omrežju, na delu omrežja podjetja Xlab ter na računalniku z zagnanimi avtomatskimi spletnimi pajki. S tem smo želeli dobiti raznoliko množico realnega delujočega omrežja z mnogimi uporabljenimi protokoli.

Ker smo uporabili prave, dalj časa delujoče računalnike, ne moremo biti prepričani, da zajemi, ki smo jih označili kot benigne, ne vsebujejo prometa botnetov ali drugih zlonamernih programov. Prav tako ne moremo trditi, da so vsi omrežni tokovi v maligni množici zares značilni zgolj za botnete, čeprav so jih ustvarili boti. Tak primer so DNS poizvedbe, ki ustvarijo enake lastnosti omrežnega toka, ne glede na to, če jih izvajajo nevarni boti ali legitimni uporabniški programi.

Na domačem omrežju je večina prometa ustvarjenega s programi P2P za deljenje datotek, spletni pajki generirajo promet spletnih brskalnikov (http(s)), na službenem omrežju pa so v uporabi najrazličnejši protokoli.

Tabela 4.1: Viri benignih omrežnih tokov.

Vir	Število naprav	Čas zajema	Število omrežnih tokov
Domače omrežje	15	20 dni	5.466.541
Omrežje Xlab	233	82 minut	552.160
Spletni pajki	1	11 ur	180.659

Tabela 4.1 prikazuje osnovne podatke o naši množici benignih omrežnih tokov.

Iz omenjenih množic smo za sestavo učne in testne množice naključno vzeli:

- 500.000 omrežnih tokov iz množice domačega omrežja,
- 500.000 omrežnih tokov iz množice omrežja Xlab,
- vseh 180.659 omrežnih tokov iz množice spletnih pajkov.

Tako dobljenih 1.180.659 benignih primerov smo naključno razdelili na učno in testno množico, in sicer dve tretjini primerov za učno množico in eno tretjino za testno. Maligne primere smo na učno in testno množico razvrstili po posameznih zajemih, tako da je 64 zajemov (19.810 omrežnih tokov) pripadlo učni množici in 32 zajemov (16.861 tokov) pripadlo testni množici. Z delitvijo po zajemih botnetov (in ne po posameznih omrežnih tokovih) bomo ugotovili, če je detektor sposoben zaznave botnetov, ki niso prisotni v učni množici. Razdelitev posameznih zajemov prometa botnetov na učno in testno množico je označena v dodatku A. Podatki o številu primerov v učni in testni množici so prikazani v tabeli 4.2.

Tabela 4.2: Učna in testna množica omrežnih tokov.

	Benigni primeri	Maligni primeri	Skupaj	Delež malignih primerov
Učna množica	787.103	19.810	806.913	2,5%
Testna množica	393.553	16.861	410.414	4,1%
Skupaj	1.180.656	36.671	1.217.327	3,0%

4.5 Testiranje

Detektor botnetov smo testirali na opisani testni množici, s čimer smo tudi določali najprimernejše parametre algoritma naključnih gozdov, ter na omrežju virtualnih računalnikov, pri čemer smo na enem od njih zagnali nekaj primerov botnetov.

V nadaljevanju si bomo pogledali definicije meril uspešnosti, ki smo jih uporabljali za določanje kvalitete detekcije, ter podatke o testiranjih detektorja na testni množici in na modelu realnega sistema.

4.5.1 Merila uspešnosti

Ker naš detektor botnetov deluje tako, da uvršča omrežne tokove med benigne in maligne, imamo pri klasifikaciji dvorazredni problem — odločamo se med dvema razredoma. Podobno kot pri testih bolezni, bomo maligne primere (nevarne omrežne tokove) označili kot pozitivne, benigne pa kot negativne. Mere za ocenjevanje uspešnosti klasifikatorjev v dvorazrednih problemih izhajajo iz enostavne matrike zmot (*confusion matrix*), prikazane v tabeli 4.3. V nadaljevanju opisane in nekatere druge mere uspešnosti natančneje obravnavata Kononenko in Robnik Šikonja v [36].

Klasifikacijska točnost

Najosnovnejša mera, klasifikacijska točnost, predstavlja delež pravilno klasificiranih testnih primerov oziroma verjetnost pravilne klasifikacije naključno

Tabela 4.3: Model matrike zmot za dvorazredne probleme.

	Napovedani razred		
Pravi razred	P	N	Vsota
P	TP	FN	$POS = TP + FN$
N	FP	TN	$NEG = FP + TN$
Vsota	$PP = TP + FP$	$PN = FN + TN$	$n = POS + NEG$

Legenda:

P - pozitivni razred; POS - število pozitivnih primerov

N - negativni razred; NEG - število negativnih primerov

n - število vseh primerov

TP (*True Positives*) - število pravilno klasificiranih pozitivnih primerov

FP (*False Positives*) - število napačno klasificiranih negativnih primerov (lažni pozitivni primeri)

TN (*True Negatives*) - število pravilno klasificiranih negativnih primerov

FN (*False Negatives*) - število napačno klasificiranih pozitivnih primerov (lažni negativni primeri)

PP (*Predicted Positives*) - število pozitivno klasificiranih primerov

PN (*Predicted Negatives*) - število negativno klasificiranih primerov

izbranega primera:

$$T = \frac{TP + TN}{n} \tag{4.1}$$

Na testni množici, predstavljeni v tabeli 4.2, lahko s trivialnim klasifikatorjem, ki vsak primer označi kot benigen, dosegli klasifikacijsko točnost $T = 95,9\%$ (tak je delež benignih primerov v množici). Ob tako neuravnoteženi testni množici klasifikacijska točnost ni najbolj primerno merilo uspešnosti, saj ne označuje uspešnosti klasifikacije posameznih razredov.

Senzitivnost in specifičnost

Senzitivnost in specifičnost sta za nas bolj primerni meri od klasifikacijske točnosti, saj ocenjujeta sposobnost pravilne klasifikacije obeh posameznih razredov.

Senzitivnost (*sensitivity*) ali priklic (*recall*) označuje delež pravilno klasificiranih pozitivnih primerov, kar lahko v našem primeru prevedemo na verjetnost zaznave nevarnega omrežnega toka oziroma botneta na omrežju:

$$Senz = \frac{TP}{TP + FN} = \frac{TP}{POS} \tag{4.2}$$

Specifičnost pa ocenjuje delež pravilno klasificiranih negativnih primerov, kar pomeni verjetnost, da bo legitimen (benigen) omrežni tok pravilno klasificiran v negativni razred:

$$Spec = \frac{TN}{TN + FP} = \frac{TN}{NEG} \tag{4.3}$$

Delež lažnih pozitivnih primerov (*false positive rate*) definiramo kot 1 - Spec = $\frac{FP}{NEG}$.

Trivialna klasifikatorja, ki označita vsak primer kot benigen ali maligen, dosežeta senzitivnost 0 s specifičnostjo 1, in obratno. Za najboljšo uspešnost poskušamo dobiti klasifikator z maksimalnima vrednostma senzitivnosti in specifičnosti, vendar s povečanjem senzitivnosti (pri čemer dobimo klasifikator, bolj občutljiv na pozitivne primere) trpi specifičnost, kar pomeni da dobimo več lažnih pozitivnih primerov. Z uravnavanjem parametrov strojnega učenja se trudimo doseči optimalno razmerje obeh mer.

Preciznost

Preciznost (*precision*) prikazuje delež resnično pozitivnih primerov med vsemi pozitivno klasificiranimi, kar pomeni verjetnost, da je omrežni tok, detektiran kot nevaren, dejansko nevaren.

$$Prec = \frac{TP}{TP + FP} = \frac{TP}{PP} \tag{4.4}$$

Preciznost je v podobnem razmerju s senzitivnostjo kot specifičnost, ob enostavnem povečanju senzitivnosti izgubimo na preciznosti klasifikatorja.

Mera F

Mera F (*F-measure*) z eno vrednostjo upošteva priklic (senzitivnost) in preciznost klasifikatorja. Definirana je kot harmonična sredina senzitivnosti in preciznosti:

$$F = \frac{2 \cdot Senz \cdot Prec}{Senz + Prec} = \frac{2\ TP}{2\ TP + FP + FN} \tag{4.5}$$

Poleg standardne mere F poznamo tudi različico, uteženo glede na želeno stopnjo upoštevanja senzitivnosti oziroma preciznosti [37]:

$$F_\beta = \frac{(1 + \beta^2) \cdot Senz \cdot Prec}{Senz + \beta^2 \cdot Prec} = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP} \tag{4.6}$$

Parameter β omogoča uporabo mere F tako, da je senzitivnost obravnavana kot β-krat bolj pomembna od preciznosti. Standardna mera F je definirana z $\beta = 1$, zato jo označujemo tudi s F_1.

Pri naših testiranjih smo uporabljali tudi mero $F_{0,5}$, s katero smo kot bolj pomembno mero obravnavali preciznost, kar pomeni, da bolj strogo kaznujemo lažne pozitivne primere kot lažne negativne.

Mere uspešnosti senzitivnost (priklic), preciznost ter mera F, v nasprotju s specifičnostjo ne nagrajujejo pravilno klasificiranih negativnih primerov (TN). Teh (nenevarnih omrežnih tokov) je pri našem problemu klasifikacije zelo veliko, zato je dobro specifičnost relativno lahko doseči. Za ocenjevanje zmožnosti zaznave botnetov v omrežnem prometu so zato dovolj senzitivnost,

preciznost in mera F. Kljub temu pa je pri primerjavi uspešnosti klasifikatorjev pomembno tudi, da stremimo k čim višjim vrednostim specifičnosti. Visoka specifičnost namreč pomeni malo lažnih pozitivnih primerov. V tabeli 4.1 vidimo, da dokaj majhno omrežje (približno 200 naprav) v dobri uri generira približno 500.000 omrežnih tokov, to pa bi pri navidez visoki, 99,90% specifičnosti, pomenilo kar 500 lažnih alarmov.

4.5.2 Testiranje na testni množici

Učno in testno množico smo opisali v poglavju 4.4, v nadaljevanju pa bomo predstavili rezultate testiranja pri različnih parametrih učnega algoritma. Parametri, ki smo jih spreminjali, so:

- **število dreves** v naključnem gozdu,
- **najmanjše število primerov** v listih drevesa (algoritem ne deli drevesnih vozlišč s številom primerov, manjšim ali enakim vrednosti tega parametra),
- prisotnost značilke **p2p**.

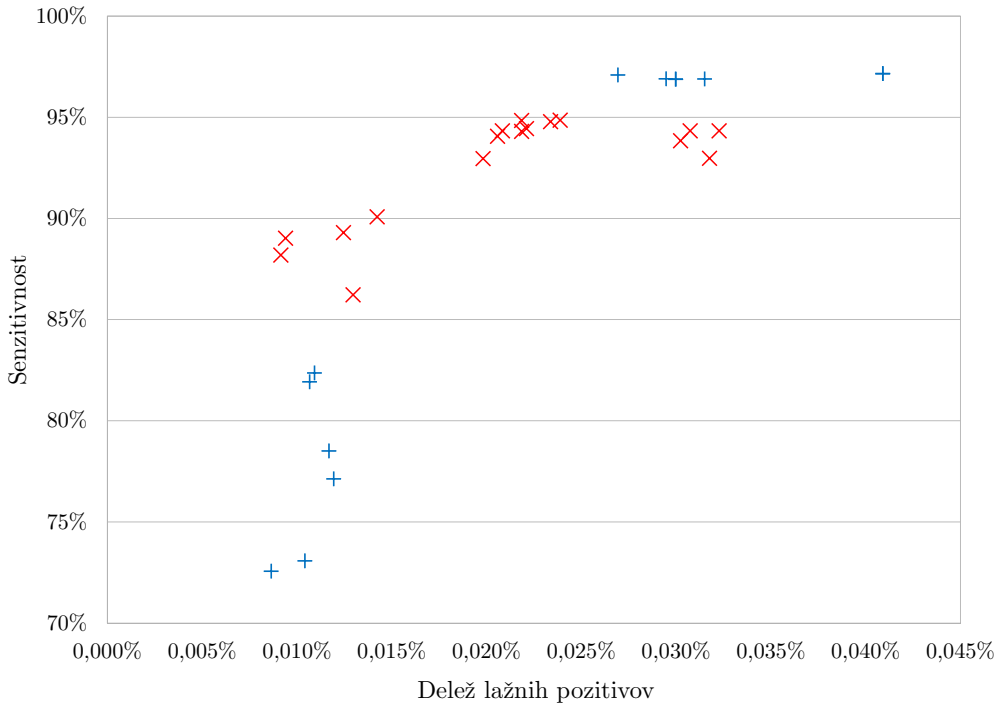
Poleg teh smo poskusili tudi z omejevanjem maksimalne globine dreves, vendar smo s tem dosegli slabše rezultate.

Nekaj rezultatov eksperimentov je prikazanih v tabeli 4.4. Razporejeni so po vrednostih mere $F_{0,5}$. Poleg teh so za primerjavo prikazani še navidezni rezultati treh trivialnih klasifikatorjev, ki vse primere klasificirajo pozitivno, negativno, oziroma naključno. Tabela z rezultati vseh eksperimentov se nahaja v dodatku B, kjer so poleg meril uspešnosti navedene tudi vse vrednosti iz matrike zmot (TP, FP, TN, FN).

Opazimo lahko, da imajo klasifikatorji z najvišjo specifičnostjo v splošnem slabšo senzitivnost, in obratno. Zato ne moremo razglasiti zmagovalne kombinacije parametrov, lahko pa glede na zahteve izberemo najbolj primeren klasifikator. Glede na velikost omrežja in količino prometa (omrežnih tokov) lahko določimo, koliko lažnih pozitivnih primerov še dovolimo klasifikatorju,

Tabela 4.4: Rezultati eksperimentov na testni množici.

Število dreves	Min. št. primerov	p2p	T	$Senz$	$Spec$	$Prec$	F	$F_{0,5}$
40	2	Ne	99,85%	97,09%	99,973%	99,36%	98,21%	98,89%
40	3	Ne	99,84%	96,90%	99,971%	99,30%	98,08%	98,81%
60	3	Ne	99,84%	96,89%	99,970%	99,28%	98,07%	98,79%
50	1	Ne	99,85%	97,15%	99,961%	99,06%	98,10%	98,67%
40	1	Ne	99,84%	97,15%	99,959%	99,03%	98,08%	98,65%
50	3	Da	99,77%	94,84%	99,978%	99,47%	97,10%	98,50%
40	4	Da	99,75%	94,33%	99,979%	99,49%	96,84%	98,41%
40	5	Da	99,54%	89,02%	99,991%	99,75%	94,08%	97,41%
20	5	Da	99,51%	88,19%	99,991%	99,76%	93,62%	97,21%
10	5	Ne	98,86%	72,57%	99,991%	99,72%	84,01%	92,78%
Vse pozitivno			4,11%	100,00%	0,00%	4,11%	7,89%	5,08%
Naključno			50,00%	50,00%	50,00%	4,11%	7,59%	5,03%
Vse negativno			95,89%	0,00%	100,00%	0,00%	0,00%	0,00%



Slika 4.2: Diagram senzitivnosti in specifičnosti klasifikatorjev. S simbolom × so označeni rezultati klasifikatorjev z vključenim atributom **p2p**, s + pa tisti brez.

in s tem najnižjo sprejemljivo specifičnost. Nato pa lahko izberemo klasifikator z najvišjo senzitivnostjo, ki temu ustreza.

Klasifikatorji, pri katerih smo uporabili značilko **p2p**, so dosegli boljšo specifičnost pri majhnem zmanjšanju senzitivnosti glede na najbolj senzitivne klasifikatorje, zgrajene brez značilke **p2p**. Diagram na sliki 4.2 prikazuje rezultate najboljših zgrajenih klasifikatorjev z in brez atributa **p2p**.

4.5.3 Testiranje na živem sistemu

Detektor botnetov smo pri tem testu testirali na omrežju virtualnih računalnikov. Detektor smo namestili na virtualni računalnik, ki je deloval kot omrežni prehod za ostale računalnike. Na detektorju smo uporabili dva klasifikacijska modela, naučena iz skupka prej opisanih učne in testne množice.

Na nekaterih virtualnih računalnikih smo poganjali uporabniške programe, ki ustvarjajo legitimen omrežni promet, na drugih pa primerke botnetov, pridobljene iz bloga Contagio Dump [35], ki ustrezajo zajemom nevarnega prometa iz učnih podatkov.

Večine botnetov iz zbirke nam ni uspelo zagnati, možni vzroki za to so: neustrezen operacijski sistem (uporabili smo Windows 7, ciljni operacijski sistem primerkov v zbirki ni označen), nedosegljivost ukazno-nadzornih strežnikov, izvajanje v virtualnem okolju (nekateri zlonamerni programi ne tečejo, če ugotovijo, da so bili zagnani na virtualiziranem sistemu, s tem skušajo otežiti njihovo analizo). Zaradi teh težav smo na enem virtualnem računalniku zagnali vse primerke botov, nato pa na vsakem od dveh drugih računalnikov še po en primerek, za katerega smo ugotovili, da deluje in generira omrežni promet.

V tabeli 4.5 so prikazani uporabljeni virtualni računalniki s podatki o delujoči aplikaciji in številu zaznanih omrežnih tokov. Za oceno uspešnosti klasifikacije smo vse omrežne tokove posameznega testnega računalnika obravnavali kot benigne (prvi del tabele) oziroma maligne (drugi del tabele). Mere uspešnosti, kot sta specifičnost in senzitivnost, so zato pri posameznih primerih tukaj nesmiselne. Klasifikacijska točnost je v primerih z benignimi omrežnimi tokovi enaka specifičnosti, v primerih z malignimi pa je enaka senzitivnosti.

V tabeli 4.6 smo navedli klasifikacijsko točnost za oba testirana klasifikacijska modela. Klasifikator A je bil zgrajen s 40 naključnimi drevesi, najmanj 2 primera v listih dreves in brez atributa **p2p**, klasifikator B pa prav tako s 40 drevesi ter z najmanj 5 primeri v listih in z atributom **p2p**. Tabela 4.7 prikazuje skupne rezultate vseh klasificiranih omrežnih tokov in primerjavo obeh klasifikatorjev s trivialnimi klasifikatorji.

Klasifikacija omrežnih tokov se je pri tem eksperimentu izkazala opazno slabše kot na testu, opisanem v poglavju 4.5.2. Klasifikator B, ki uporablja atribut **p2p** je dosegel nekoliko boljše rezultate kot klasifikator A. Najbolj vidna razlika med njima je pri testnem računalniku b7, na katerem smo pognali

program za izmenjavo datotek P2P. Tu je klasifikator B dosegel najvišjo klasifikacijsko točnost, klasifikator A pa najnižjo. Pri zaznavi malignih omrežnih tokov sta oba klasifikatorja dosegla podobne rezultate. Dobro sta klasificirala promet botneta Citadel (računalnik m3), druge botnete pa slabo — Kelihos (primer m2) z le 2,7% točnostjo. Skupna senzitivnost detektorja pri tem eksperimentu je bila pod 7% s približno 99% specifičnostjo.

Sklep

V praksi bi s tako uspešnostjo detektorja težko zanesljivo razločili okužene računalnike na omrežju. Ker se v klasifikaciji pojavljajo tudi lažni pozitivni primeri, ne bi bilo smiselno vsakega računalnika, pri katerem se pojavi maligno klasificiran omrežni tok, razglasiti za okuženega. Za nadzornika omrežja bi bilo smiselno pregledati računalnike z velikim številom ali z visokim deležem maligno klasificiranih omrežnih tokov. Slika 4.2 prikazuje delež maligno klasificiranih omrežnih tokov za oba testirana klasifikatorja na posameznih testnih računalnikih. Upoštevati moramo, da bi v resničnem primeru tudi najuspešneje klasificiran maligen primer (Citadel, m3) manj izstopal med ostalimi računalniki, če bi na njem tekli tudi legitimni programi, kar bi zmanjšalo delež zaznanih malignih omrežnih tokov.

Razlike v uspešnosti detekcije na testni množici in na živem eksperimentu si lahko razlagamo z razliko v delovanju večine botnetov med zajemom njihovega omrežnega prometa, katerega smo uporabili za učenje modelov, in med testiranjem na testnih računalnikih. Boti so lahko v drugi življenjski fazi (poglavje 2.4), izvajajo drugačne akcije, ali pa je celo spremenjen njihov način komunikacije z ukazno-nadzornimi strežniki oziroma drugimi boti (pri komunikaciji vsak z vsakim).

Iz tega lahko sklepamo, da se s spremembami v delovanju botnetov uspešnost detekcije zmanjšuje. To pomeni, da naučen model ni dovolj splošen za dolgotrajno uporabo v različnih omrežjih. Ocenjujemo, da bi bilo potrebno za izboljšanje detekcije v realnih primerih redno posodabljati detektor z modeli, naučenimi na najnovejših zajemih primerkov botnetov. S tem se približamo

delovanju detektorjev, temelječih na zaznavi podpisov, kar pa močno zmanjša uporabnost našega sistema.

4.5.4 Uspešnost zaznave mobilnih botnetov

Z detektorjem botnetov na omrežnem prometu smo poskušali zaznati tudi botnete na mobilnih napravah. Na virtualni računalnik z detektorjem smo priključili brezžično dostopno točko, nanjo pa povezali testno mobilno napravo. Iz spletne zbirke Contagio Dump [35] smo pridobili približno 30 primerkov zlonamernih programov, ki imajo funkcijo komunikacije s strežniki C&C, ter jih namestili na testno napravo. Ugotovili smo, da je večina strežnikov, do katerih so poskušale dostopati nameščene zlonamerne aplikacije, nedosegljivih. Pridobili smo namreč lahko le predhodno odkrite in analizirane zlonamerne aplikacije, zato je velika verjetnost, da so bili njihovi strežniki C&C že onemogočeni. Pri klasičnih botnetih je onemogočanje strežnikov C&C težje, ker tipično uporabljajo naprednejše protokole z večjim številom strežnikov C&C ali promet vsak z vsakim (P2P).

Med vsemi zaznanimi je naš omrežni detektor kot maligne klasificiral približno 1% omrežnih tokov, ti pa so bili enakomerno razporejeni čez celoten čas testiranja. Enak delež maligno klasificiranih omrežnih tokov smo dobili tudi z uporabo neokužene mobilne naprave, zato sklepamo, da gre za lažne pozitivne primere. Naša metoda detekcije botnetov na omrežnem prometu torej ni zaznala prisotnosti botnetov na mobilni napravi.

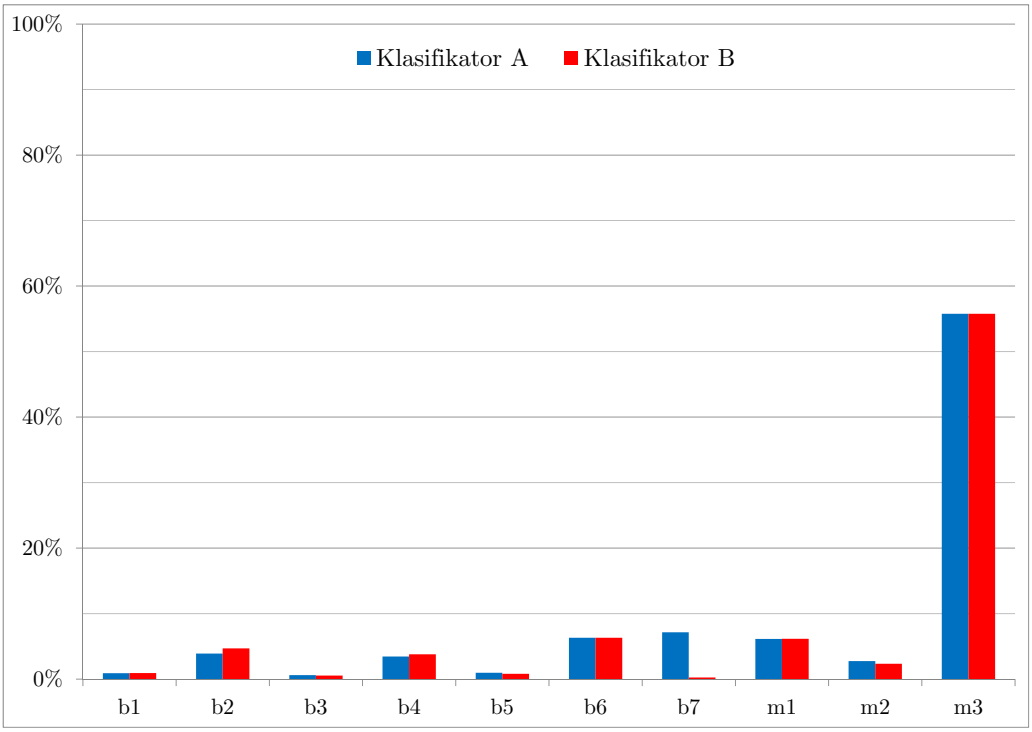
Vzrok za neuspešno zaznavo botnetov na mobilnih napravah so razlike v delovanju klasičnih in mobilnih botnetov. Opazili smo, da imajo boti za mobilne naprave svoje funkcije tipično definirane vnaprej in lahko delujejo brez začetnega javljanja strežniku in prevzema ukazov. Strežnik C&C zato kontaktirajo šele pri posredovanju rezultatov (kot so uporabnikovi osebni podatki), kar pomeni manj omrežnega prometa kot pri klasičnih botnetih. Dodatna težava za detekcijo je uporaba drugih komunikacijskih poti, kar smo že omenili v poglavju 3.2.

Tabela 4.5: Virtualni računalniki, uporabljeni za testiranje.

Oznaka	OS	Aplikacija	Število tokov	Čas delovanja	Št. tokov na minuto
b1	Windows XP	Spletni pajek	17638	5 ur 7 min	57,5
b2	Android	/	128	6 ur 11 min	0,3
b3	Ubuntu	Spletni pajek	19375	6 ur 27 min	50,1
b4	Lubuntu	Posodobitve OS	1769	6 ur 14 min	4,7
b5	Lubuntu	Spletni pajek	14139	6 ur 17 min	37,5
b6	Windows 7	/	190	5 ur 52 min	0,5
b7	Ubuntu	BitTorrent	1202	1 ura 13 min	16,5
m1	Windows 7	Vsi delujoči botneti	18904	6 ur 9 min	51,2
m2	Windows 7	Kelihos	255	52 min	4,9
m3	Windows 7	Citadel	312	1 ura 14 min	4,2

Tabela 4.6: Rezultati klasifikacije na testnih računalnikih.

	Klasifikator A			Klasifikator B		
	Tokovi zaznani kot			Tokovi zaznani kot		
Oznaka	benigni	maligni	T	benigni	maligni	T
b1	17477	161	99,1%	17474	164	99.1%
b2	123	5	96,1%	122	6	95.3%
b3	19254	121	99,4%	19269	106	99.5%
b4	1708	61	96,6%	1702	67	96.2%
b5	14002	137	99,0%	14025	114	99.2%
b6	178	12	93,7%	178	12	93.7%
b7	1116	86	92,9%	1199	3	99.8%
m1	17741	1163	6,2%	17737	1167	6.2%
m2	248	7	2,7%	249	6	2.4%
m3	138	174	55,8%	138	174	55.8%



Slika 4.3: Delež pozitivno klasificiranih omrežnih tokov za oba klasifikatorja po testnih računalnikih.

Tabela 4.7: Skupni rezultati klasifikacije in primerjava s trivialnimi klasifikatorji.

Klasifikator	T	$Senz$	$Spec$	$Prec$	F	$F_{0,5}$
A	74.69%	6.90%	98.93%	69.75%	12.56%	24.72%
B	74.84%	6.92%	99.13%	74.05%	12.65%	25.18%
Vse pozitivno	26.34%	100.00%	0.00%	26.34%	41.70%	30.89%
Naključno	50.00%	2.61%	97.39%	49.97%	4.95%	10.78%
Vse negativno	73.66%	0.00%	100.00%	0.00%	0.00%	0.00%

4.5.5 Primerjava z literaturo

Singh et al., avtorji članka *Big Data Analytics Framework for Peer-to-Peer Botnet Detection Using Random Forests* [1], po ideji katerega je bil zasnovan naš detektor, so za sestavo učne in testne množice omrežni promet botnetov delno zajeli z lastnim poganjanjem primerkov botnetov, del malignega omrežnega prometa pa so pridobili iz spletnih zbirk zlonamernih programov. Testiranje sistema so izvedli s podajanjem prometa testne množice v detektor. S tem so dosegli podoben način testiranja, kot mi s testiranjem na testni množici, kar smo opisali v poglavju 4.5.2. Dosegli so 99,7% specifičnost in 99,8% senzitivnost ter 99,9% preciznost, pri čemer mera F znaša 99,85%, mera $F_{0,5}$ pa 99,88%. Navedenih mer ne moremo direktno primerjati z našimi, saj sta uporabljeni testni množici precej različni. Singh et al. so za testno množico uporabili 10% primerov iz množice 84.030 omrežnih tokov, kjer je bilo 66% prometa botnetov in 34% benignih primerov. Vseeno lahko trdimo, da so rezultati našega testiranja podobni kot v omenjenem članku.

Zaradi specifik sistema rezultatov ni smiselno primerjati z rešitvami, ki jih opisujejo drugi članki, omenjeni v poglavju 1.1. Drugi sistemi se bodisi zanašajo na pojav več okuženih naprav v zaznavnem omrežju [2, 4], bodisi se osredotočajo na specifične protokole komunikacije botnetov [3]. Metoda, ki so jo predstavili Singh et al. in smo jo implementirali v okviru tega dela, odpravlja te pomanjkljivosti.

Poglavje 5

Detektor zlonamerne programske opreme na operacijskem sistemu Android

5.1 Cilji

Cilj razvoja detektorja zlonamerne programske opreme na mobilnih napravah je bil raziskava varnostnih ranljivosti operacijskega sistema Android in možnosti zaznavanja njihovega izkoriščanja ter implementacija celovite varnostne aplikacije, ki omogoča obveščanje uporabnika o dostopanju do nevarnih virov in o zlonamernih nameščenih aplikacijah ter poročanje centralnemu strežniku o nevarnih dogodkih na prenosnih napravah.

Pri implementaciji mobilne aplikacije moramo upoštevati omejitev dostopa do virov, dostopnih le uporabnikom s pravicami administratorja, poskrbeti moramo za čim boljše varovanje zasebnosti uporabnikov in za minimalno porabo energije na mobilnih napravah.

S tem namenom smo razvili aplikacijo Device Monitor [38], ki jo bomo predstavili v tem poglavju. Device Monitor deluje na podlagi zaznave znanih vzorcev. V aplikacijskih datotekah prepoznava izkoriščanje tako imenovanih Master Key in Fake ID ranljivosti, glede na pravice, ki jih aplikacije zahte-

vajo, ocenjuje stopnjo grožnje, ki jo aplikacija predstavlja za uporabnikovo varnost in zasebnost, na podlagi spremljanja zapisov v bazi sporočil zaznava tako imenovane ugrabitve sporočil SMS, ter opozarja o povezovanju na znane nevarne vire.

Uspešnost zaznave anomalij na način, kot smo ga uporabili pri detektorju botnetov na omrežju, je na mobilnih napravah vprašljiva zaradi omejenih računskih virov in težav s porabo energije, omejenih možnosti zaznave virov anomalij zaradi omejitve pravic ter zaradi različnih komunikacijskih kanalov, ki se jih poslužujejo zlonamerni programi na mobilnih napravah.

Poleg mobilne aplikacije Device Monitor smo implementirali tudi spletni strežnik z imenom GCM Server, ki skrbi za posredovanje podatkov, potrebnih za uspešno detekcijo nevarnih dogodkov, napravam, ter za zbiranje in pregled zaznanih nepravilnosti na mobilnih napravah.

Projekt ACDC

Aplikacija Device Monitor je bila uporabljena v okviru pilotnega projekta ACDC[1] [39]. ACDC je projekt 28-ih partnerjev iz 14-ih evropskih držav, financiran s strani Evropske unije, s skupnim ciljem boja proti botnetom. Partnerji projekta nudijo podporne centre in varnostna orodja za zaznavo, preprečevanje širjenja in ustavitev botnetov. Orodja povezuje skupni center za izmenjavo podatkov, ki se imenuje CCH[2].

Naše orodje GCM Server iz centra CCH pridobiva podatke o nevarnih virih (na primer naslove spletnih strani, ki izvajajo napade z ribarjenjem (*phishing*) ali naslove IP znanih strežnikov C&C), preko CCH pa s partnerji deli podatke o zaznanih zlonamernih aplikacijah na mobilnih napravah ter zaznanih povezavah na označene nevarne vire.

[1] Projekt ACDC (*Advanced Cyber Defence Centre*) je evropski projekt okvirnega programa FP7 (klic CIP-ICT-PSP-2012-6, št. pogodbe: 325188).

[2] *Central clearing house*

5.2 Zaznava groženj

Mobilni detektor zlonamerne programske opreme smo napisali v programskem jeziku Java. Za komunikacijo s strežnikom uporabljamo povezavo HTTPS, za shranjevanje podatkov pa podatkovno bazo SQLite. V podatkovno bazo aplikacija shranjuje pravila za zaznavo nevarnih omrežnih povezav, podatke o zaznanih dogodkih, ocene nameščenih aplikacij in t.i. beli seznam certifikatov.

Funkcije zaznave groženj so v aplikaciji zasnovane kot moduli. Vsak modul ob detekciji grožnje generira dogodek (objekt razreda **Event**), ga zapiše v podatkovno bazo in po potrebi sproži obvestilo uporabniku. Dogodek vsebuje čas zaznave, nivo nevarnosti in dodatne podatke o dogodku v obliki JSON. V dodatnih podatkih je zapisan tip dogodka ter podatki, odvisni od posameznega modula detekcije. Modularna zasnova zaznavanja omogoča enostavnejšo razširitev aplikacije z dodatnimi funkcijami zaznavanja. Nivo nevarnosti je določen glede na tip in podrobnosti dogodka ter je označen z eno izmed petih vrednosti:

1. varno (SAFE),
2. informacija (INFO),
3. nizka nevarnost (LOW),
4. srednja nevarnost (MEDIUM),
5. visoka nevarnost (HIGH).

Zaznani dogodki so sporočeni strežniku med postopkom sinhronizacije. V primeru dogodka z visoko stopnjo nevarnosti je sinhronizacija izvedena takoj po zaznavi grožnje, sicer pa v periodičnih intervalih.

5.2.1 Detekcija nevarnih omrežnih povezav

Device Monitor zaznava vzpostavljene omrežne povezave in opozarja na povezave k virom, ki so označeni kot nevarni. Funkcija je namenjena predvsem

zaznavanju povezav s strežniki C&C. Zaznavanje nevarnih povezav lahko na omrežju enostavno dosežemo s sistemom za detekcijo vdorov (IDS), z uporabo naše aplikacije pa lahko nevarne povezave zaznavamo tudi izven domačega omrežja. Poleg tega je na ta način mogoča tudi identifikacija aplikacije na napravi, ki je povezavo vzpostavila.

Detekcija nevarnih povezav deluje s primerjavo naslovov IP vzpostavljenih povezav na mobilni napravi s seznamom naslovov IP nevarnih virov. Seznam nevarnih virov aplikaciji posreduje strežnik, ki naslove strežnikov C&C pridobiva iz centralne baze CCH.

Vzpostavljene povezave pridobimo z branjem datotek `/proc/net/tcp` in `/proc/net/udp`. Datoteki lahko beremo s pravicami navadnega uporabnika, v njima pa je zapisan seznam vzpostavljenih omrežnih povezav vseh aplikacij. Poleg naslovov IP in omrežnih vrat na izvorni in ponorni strani je za vsako povezavo zapisano še stanje povezave in številka (ID) uporabnika lastnika povezave. Ker ima vsaka nameščena aplikacija na operacijskem sistemu Android svojo številko uporabnika, lahko preko te številke določimo tudi aplikacijo, ki dostopa do nevarnih virov.

Oznaka stanja povezave je smiselna pri povezavah protokola TCP in pove, ali je povezava aktivna, v procesu vzpostavitve, v procesu prekinitve, ali prekinjena. Povezave TCP ostanejo v seznamu še (privzeto) 60 sekund po prekinitvi, vendar v tem stanju nimajo zapisane številke uporabnika. Preverjanje povezav se vrši vsakih deset sekund kadar ima naprava vzpostavljeno povezavo z omrežjem in ni v stanju spanja (ima aktivno CPE). Komunikacija preko protokola UDP, ki traja kratek čas, se lahko detekciji izogne, če se med prenosom podatkov ne sproži preverjanje povezav. Temu bi se lahko izognili z zelo pogostim preverjanjem, vendar bi to vplivalo na uporabo računskih virov in energije.

Tip dogodka detekcije nevarne omrežne povezave je `SuspiciousConnectionEvent`. Poročilo o dogodku vsebuje izvorni in ponorni naslov IP ter številki vrat, oznako protokola transportnega nivoja, ime procesa (aplikacije) lastnika povezave in stanje povezave.

Bloomov filter

Poizvedovanje na strežnik glede nevarnosti vsake zaznane povezave bi zahtevalo veliko omrežnega prometa in virov na mobilni napravi, sprejemanje vseh nevarnih naslovov IP iz strežnika in hranjenje na napravi pa bi bilo podobno neučinkovito. Zato smo za prenašanje seznama nevarnih naslovov IP uporabili podatkovno strukturo Bloomov filter. Ta omogoča prostorsko učinkovito hrambo množice naslovov, pri čemer pa imamo možnost lažnih zadetkov. Opis podatkovne strukture Bloomov filter najdemo v [40]. Bloomov filter predstavimo z bitno tabelo z m polji. Vsaka od k zgoščevalnih funkcij preslika element množice (v našem primeru je to naslov IP) v eno lokacijo v tabeli. Naslov IP je tako predstavljen s k biti v m-bitni tabeli. Ob vstavljanju elementa v Bloomov filter postavimo vseh k pripadajočih bitov tabele na 1. Pripadnost elementa množici enostavno ugotovimo s pregledom elementu pripadajočih bitov v tabeli. Če je vseh k bitov postavljenih, trdimo, da je element v množici prisoten, pri čemer imamo možnost lažnega zadetka. Če kateri od bitov ni postavljen, elementa zagotovo ni v množici. Pri testiranju elementa, ki ga ni v množici (naslov IP, ki ni objavljen kot nevaren), je verjetnost za lažni zadetek:

$$P = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - \mathrm{e}^{-kn/m}\right)^k \tag{5.1}$$

kjer je n število vseh elementov v množici (vseh nevarnih naslovov IP). Pri gradnji Bloomovega filtra lahko želeno verjetnost P dosežemo z izbiro parametrov m in k glede na število elementov n:

$$m = \frac{-n \ln P}{(\ln 2)^2} \qquad\qquad k = \frac{m}{n} \ln 2 \tag{5.2}$$

V našem primeru želimo doseči verjetnost napake lažnega zadetka $P = 0,001$. Če ima strežnik v bazi $n = 5000$ naslovov IP, bo ustvaril bitno tabelo z $m = 71888$ mesti, v kateri je vsak element preslikan v $k = 10$ bitov.

Pravila detekcije nevarnih povezav se v obliki Bloomovega filtra v aplikaciji posodobijo ob sinhronizaciji podatkov s strežnikom. Strežnik pošlje

s številom predstavljeno bitno tabelo ter parametra k in m. Pri pregledu vzpostavljenih povezav Device Monitor za vsako od njih preveri ujemanje v Bloomovem filtru. Ob zadetku v filtru naredimo poizvedbo o naslovu IP na strežnik, ki nam vrne vrednost nivoja nevarnosti. Če je bil zadetek v Bloomovem filtru lažen, strežnik vrne vrednost "varno", sicer pa eno od ostalih stopenj nevarnosti. V tem primeru aplikacija ustvari dogodek nevarne povezave ter obvesti uporabnika. V lokalno podatkovno bazo na napravi si v obeh primerih shranimo podatek o nivoju nevarnosti za ta naslov IP. S tem odpravimo potrebo po ponovnih poizvedbah za enak naslov IP.

5.2.2 Preverjanje povezav URL

Preverjanje povezav URL je funkcija, ki ni povezana z zaznavo zlonamerne programske opreme, temveč je namenjena preprečevanju okužb in podatkovnega ribarjenja. Ob kliku na povezavo URL zunaj brskalnika (npr. v e-poštnem sporočilu ali sporočilu SMS) lahko uporabnik izbere odpiranje z Device Monitorjem. Ta s poizvedbo na strežnik preveri povezavo in uporabnika obvesti o morebitni nevarnosti ali, če je povezava varna, avtomatsko preusmeri v spletni brskalnik. Strežnik ob prejetju poizvedbe iz mobilne naprave preveri naslov URL v svoji podatkovni bazi, v katero vpisuje naslove nevarnih virov ob prejetju obvestil iz baze CCH, ter v drugih, oddaljeno dostopnih podatkovnih bazah nevarnih spletnih strani, kot je Google Safe Browsing. Z opozorilom o nevarnih spletnih straneh preprečujemo obisk strani podatkovnega rudarjenja in spletnih strani, ki ponujajo prenos zlonamernih aplikacij.

Med brskanjem po spletu znotraj spletnega brskalnika posameznih povezav ne moremo zaznati. Če želi uporabnik neko povezavo posebej preveriti, lahko to stori s funkcijo "deli z..." in v seznamu izbere Device Monitor.

Ob preverjanju povezave URL, ki se izkaže za nevarno, se ustvari poročilo o dogodku tipa `UriBrowseEvent`, ki vsebuje nevaren naslov URL.

5.2.3 Detekcija ugrabitev sporočil SMS

Sporočila SMS so na napravah s sistemom Android shranjena v posebni bazi, ki jo vodi operacijski sistem in je dostopna vsem aplikacijam s pravico za branje sporočil SMS. Aplikacije z nastavljeno pravico za prejemanje sporočil SMS se lahko registrirajo za prejemanje sporočil SMS, kar pomeni, da jim takoj ob prejetju sporočila operacijski sistem to sporočilo posreduje. Registrirane aplikacije imajo lahko nastavljeno prioriteto lastnega prejetja sporočila. Ob prejetju sporočila SMS operacijski sistem dostavi sporočilo aplikacijam po vrsti od tistih z najvišjo, do tistih z najnižjo prioriteto. Vrednosti prioritete, predpisane v dokumentaciji za razvijalce aplikacij Android, so od -999 do 999, kjer je privzeta vrednost 0. Kljub temu Android dovoli tudi višje vrednosti prioritete, vse do $2^{31} - 1 = 2147483647$. Privzeta sistemska aplikacija za branje sporočil piše sporočila v bazo ter obvešča uporabnika o prejetih sporočilih. Če ta aplikacija ne dobi obvestila o prejetem sporočilu, uporabnik ne more vedeti, da je sporočilo prejel. Aplikacija, ki dobi obvestilo o sporočilu SMS, pa lahko ustavi nadaljnjo dostavo obvestila aplikacijam z nižjo prioriteto. To pomeni, da lahko aplikacija, ki ima visoko prioriteto dostave sporočil SMS, sporočila prestreže in skrije pred ostalimi aplikacijami ter uporabnikom. Temu pravimo ugrabitev sporočila SMS.

Ugrabljanje sporočil SMS uporabljajo zlonamerne aplikacije za prikritje ukazno-nadzorne komunikacije prek sporočil SMS ali, še pogosteje, za prikritje sporočil SMS, ki so odgovor plačljivih storitev SMS. Zloraba plačljivih storitev SMS poteka tako, da se zlonamerna aplikacija naroči na tako storitev, storitev pa na uporabnikovo številko pošilja tako imenovana premium sporočila SMS, ki se uporabniku zaračunavajo. Pri tem želi zlonamerna aplikacija na mobilni napravi prikriti prejeta sporočila, zato, da uporabnik tega ne opazi.

Naša aplikacija detektira ugrabitve sporočil SMS, in sicer tako, da je registrirana kot prejemnik sporočil SMS z najvišjo možno prioriteto. Po prejemu sporočila si sporočilo zapomni in prepusti v obdelavo aplikacijam z nižjo prioriteto. Zatem se sproži preverjanje, ki enkrat na sekundo preveri, če je

prejeto sporočilo že zapisano v sistemski bazi sporočil. Če po 40 sekundah sporočila ni v bazi, se sproži dogodek o ugrabitvi sporočila SMS. 40 sekund časa je ocena, koliko časa največ lahko potrebujejo morebitne druge registrirane aplikacije za obdelavo sporočila, preden je ta zapisan v bazo. Če bi preverjali le enkrat ob koncu 40 sekundnega intervala, bi lahko uporabnik pred tem sporočilo že prebral in izbrisal, kljub temu pa bi to zaznali kot ugrabitev sporočila SMS. Ocenjujemo, da je ena sekunda od vpisa v bazo premalo, da bi uporabnik lahko sporočilo izbrisal. Če ima več aplikacij nastavljeno najvišjo prioriteto dostave sporočil SMS, je sporočilo najprej dostavljeno tisti aplikaciji, ki je bila na sistem nameščena najprej. Omejitev naše metode detekcije je, da mora biti detektor na napravo nameščen pred zlonamerno aplikacijo z najvišjo prioriteto dostave sporočil.

Ob detekciji ugrabitve sporočila SMS je uporabnik o tem obveščen, v obvestilu pa se izpiše telefonska številka pošiljatelja in tekst ugrabljenega sporočila, nemogoče pa je ugotoviti, katera aplikacija je krivec za ugrabitev sporočila.

Z verzijo operacijskega sistema Android 4.4, ki je izšla novembra 2013, se je način prejemanja sporočil spremenil, tako, da ima uporabnik možnost nastaviti aplikacijo, ki je privilegirana pri prejetju sporočil SMS. To pomeni, da nastavljena aplikacija zagotovo dobi obvestilo o novem sporočilu, samo ta aplikacija pa ga tudi lahko zapiše v sistemsko bazo sporočil. To do neke mere odpravlja možnost ugrabitve sporočil, saj mora uporabnik posebej nastaviti glavno aplikacijo za prejemanje sporočil, samo ta pa lahko sporočilo ugrabi. Če pride do ugrabitve sporočila, pa lahko to z našo aplikacijo vseeno zaznamo.

Tip dogodka ugrabitve sporočila SMS je `SMSHijackEvent`, poročilo o njem pa vsebuje besedilo sporočila, telefonsko številko pošiljatelja, številko centra za pošiljanje sporočil in, pri verzijah Android 4.4 in več, ime aplikacije, nastavljene za prejemanje sporočil SMS. Zaradi varovanja uporabnikove zasebnosti je namesto besedila strežniku posredovana le zgoščena vrednost besedila sporočila.

5.2.4 Detekcija zlonamernih aplikacij

Detekcija zlonamernih aplikacij obsega pregled zahtevanih pravic aplikacije, detekcijo izkoriščanja Master Key in Fake ID ranljivosti ter primerjavo zgoščene vrednosti namestitvene datoteke aplikacije z znanimi vrednostmi zlonamernih aplikacij. Vsak od teh modulov vrne oceno stopnje nevarnosti aplikacije, skupna stopnja pa je maksimum teh ocen. Detekcija se izvede po namestitvi Device Monitorja, po namestitvi ali posodobitvi druge aplikacije ali na zahtevo uporabnika. Uporabnik lahko v pregledu aplikacij (*application list*) pregleda ocene vseh nameščenih aplikacij in njihove razlage. Če je aplikacija ocenjena z oceno visoke nevarnosti, se sproži dogodek o nevarni aplikaciji, uporabnik prejme opozorilo, namestitvena datoteka nevarne aplikacije pa se ob povezavi WiFi (opcijsko) prenese na strežnik. Tam zbiramo primerke nevarnih aplikacij za dodatno analizo.

Poleg avtomatske ocene nameščenih aplikacij lahko ocenimo tudi nenameščene aplikacije, in sicer z odpiranjem datoteke APK z Device Monitorjem. To ni mogoče pri aplikacijah iz uradne Android trgovine, saj se te avtomatsko namestijo takoj po prenosu. Funkcija je namenjena aplikacijam, prenesenim iz drugih virov.

Za klasifikacijo stopnje nevarnosti aplikacije na podlagi zahtevanih pravic uporabljamo seznam pravic z njihovimi stopnjami, kjer na primer dostop do kamere, geografske lokacije ali podatkov o stikih pomeni stopnjo informacije, dostop do sporočil iz družabnih omrežij, zgodovine klicev in spletnega brskanja pomeni nizko nevarnost, branje in sprejemanje sporočil SMS, pošiljanje sporočil SMS in klicanje pa pomeni srednjo stopnjo nevarnosti. Če je prisotna pravica za sprejemanje sporočil SMS, se dodatno preveri tudi prioriteta sprejemnika sporočil, pri čemer stopnja prioritete 1000 in več pomeni visoko stopnjo nevarnosti, ker to omogoča ugrabitve sporočil SMS. Seznam zahtevanih pravic drugih nameščenih aplikacij pridobimo preko programskega vmesnika Android, za kar niso potrebne posebne pravice. Pri klasifikaciji nenameščenih aplikacij pa zahtevane pravice in druge podatke o aplikaciji preberemo iz datoteke `AndroidManifest.xml` v paketu APK.

Detekcija izkoriščanja Master Key ranljivosti deluje tako, da preverimo, če datoteka APK vsebuje več datotek z enakim imenom, če je dolžina katerega od dodatnih polj označena s previsoko vrednostjo, ali če dolžine imen datotek v lokalnih glavah niso skladne s centralnim imenikom. Izkoriščanje ranljivosti Fake ID detektiramo s pravilnim preverjanjem verige certifikatov.

Zgoščene vrednosti namestitvenih datotek (APK) vseh nameščenih aplikacij se ob vsaki sinhronizaciji s strežnikom pošljejo strežniku. Na strežniku preverimo, če se katera od njih ujema z vrednostmi v seznamu zlonamernih aplikacij in to sporočimo mobilni napravi. Seznam na strežniku se posodablja s podatki iz centralne baze CCH. Aplikacije, pri katerih zaznamo izkoriščanje Master Key ali Fake ID ranljivosti ali ujemanje z objavljeno zlonamerno zgoščeno vrednostjo, so označene z visoko stopnjo nevarnosti.

Aplikacije, ki niso nameščene preko uradne Android trgovine, imajo avtomatsko dodeljeno vsaj srednjo stopnjo nevarnosti.

Uporabnik lahko označi poljubne aplikacije kot zaupanja vredne in se s tem izogne detekciji vseh dogodkov teh aplikacij. Podoben učinek ima beli seznam aplikacij, ki je določen na strežniku in vsebuje certifikate zaupanja vrednih izdajateljev aplikacij. Aplikacije, podpisane s certifikati v belem seznamu, niso nikoli zaznane kot zlonamerne. Zaradi možnosti zlorab se izkoriščanje Master Key in Fake ID ranljivosti vseeno preveri. V beli seznam smo dodali certifikate nekaterih pogosto uporabljanih aplikacij, dodali pa smo tudi certifikat Device Monitorja. Aplikacija namreč ocenjuje tudi sama sebe, pri čemer je zaradi visoke prioritete sprejemnika sporočil SMS ocenjena s stopnjo visoke nevarnosti.

Zaznava zlonamerne aplikacije je sporočena v poročilu dogodka tipa `MaliciousAppEvent`, ki vsebuje čas namestitve in čas zadnje posodobitve aplikacije, polno in kratko ime aplikacije, oznako verzije, zgoščeno vrednost namestitvene datoteke in podrobnosti o klasifikaciji, ki povedo, zakaj je bila aplikacija klasificirana kot nevarna.

5.3 Uporabniški vmesnik

Glavni prikaz v aplikaciji ponuja informacije o preteklih zaznanih dogodkih, ki so združeni glede na aplikacijo, ki je dogodek sprožila. Dogodki, ki nimajo pripadajoče aplikacije, so združeni v poseben zavihek. Glavni prikaz z zaznanimi dogodki je prikazan na sliki 5.1.

S klikom na posamezen dogodek prikažemo podrobnosti o dogodku. Ta prikaz je dostopen tudi s klikom na obvestilo, ki se prikaže takoj po zaznavi dogodka v sistemskem meniju. Podatki, prikazani v podrobnostih dogodka, so odvisni od tipa dogodka. Slika 5.2 prikazuje dogodek ugrabitve sporočila SMS.

Iz menija v glavnem prikazu je dostopen seznam aplikacij, ki prikazuje vse nameščene aplikacije, urejene po ocenjeni stopnji nevarnosti. Ocena nevarnosti je ob imenu aplikacije slikovno prikazana z barvo ikone ob njej. Seznam aplikacij je prikazan na sliki 5.3. Opcijsko lahko uporabnik skrije sistemske aplikacije in tiste, katere je označil kot zaupljive. Sproži lahko tudi ponovno klasifikacijo vseh aplikacij.

Ob kliku na aplikacijo v seznamu se prikažejo podrobnosti o njeni oceni. Podrobnosti vsebujejo razlago ocene in detajle o aplikaciji skupaj z vsemi zahtevanimi pravicami. Uporabnik ima na tem prikazu možnost označiti aplikacijo kot zaupanja vredno ali to oznako odstraniti.

Prikaz nastavitev je na voljo iz menija glavnega prikaza. Vsebuje možnosti za izbris vseh dogodkov, ponoven prikaz skritih dogodkov, nastavitev pošiljanja zaznanih zlonamernih aplikacij na strežnik ter možnost vklopa oziroma izklopa avtomatske detekcije.

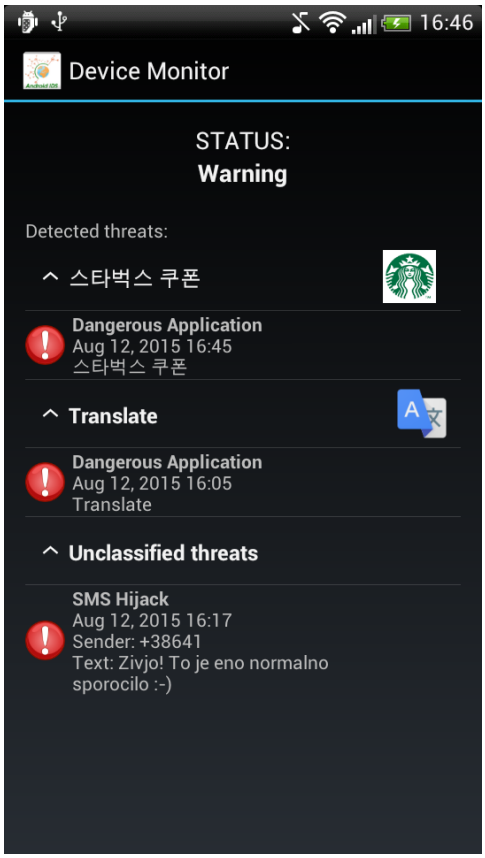
5.4 Spletni strežnik

Za podporo mobilni aplikaciji Device Monitor smo razvili spletni strežnik, ki smo ga poimenovali GCM Server. Za implementacijo smo uporabili platformo Java EE s podatkovno bazo MySQL. Strežnik zbira podatke o dogodkih iz mobilnih naprav in jim posreduje podatke, pomembne za zaznavo groženj.

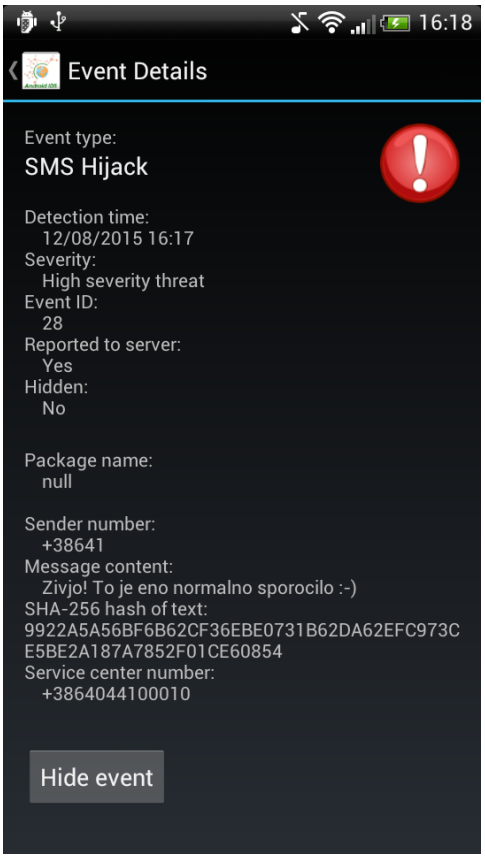
Proces izmenjave podatkov med aplikacijo na mobilni napravi in spletnim strežnikom imenujemo sinhronizacija podatkov. Proces sproži mobilna aplikacija enkrat na uro oziroma glede na nastavitve naprave. Med sinhronizacijo podatkov se iz mobilne naprave na strežnik prenesejo podatki o zaznanih dogodkih, številka verzije mobilne aplikacije (ta omogoča strežniku prilagajanje odgovora glede na podporo v posamezni verziji) in seznam zgoščenih vrednosti vseh nameščenih aplikacij. Strežnik napravi odgovori s podatki o nevarnih naslovih IP (v obliki Bloomovega filtra), spremembah (od zadnje sinhronizacije) belega seznama certifikatov aplikacij in z zgoščenimi vrednostmi aplikacij, ki so označene kot zlonamerne izmed tistih, poslanih s strani mobilne naprave. Poleg postopka sinhronizacije podatkov mobilna naprava pri strežniku preverja tudi morebitno nevarne povezave URL in naslove IP, ki so vsebovani v Bloomovem filtru. Vsi podatki, preneseni med mobilno aplikacijo in strežnikom, se izmenjujejo v obliki JSON.

Vsaki zahtevi, ki jo mobilna aplikacija pošlje na strežnik, mora priložiti ID naprave. To je 160-bitno število, ki ga Device Monitor naključno generira ob namestitvi in je uporabljeno za identifikacijo naprave. Ta način omogoča identifikacijo brez vnosa podatkov uporabnika. Ob prvi povezavi Device Monitorja s strežnikom se izvede registracija, pri čemer mora uporabnik rešiti slikovni test CAPTCHA, ki ga generira strežnik. To preprečuje avtomatsko množično registracijo, kar bi lahko izkoristili za napad na sistem.

Strežnik je z mobilnimi napravami povezan tudi preko sistema GCM (*Google Cloud Messaging*), ki omogoča pošiljanje sporočil iz strežnika na mobilne naprave preko Googlove storitve. Z uporabo te funkcije lahko upravnik sistema sproži sinhronizacijo podatkov posamezne ali vseh mobilnih naprav ali pošlje sporočilo, ki se prikaže kot obvestilo uporabniku na mobilni napravi. Napravo, kateri želimo poslati sporočilo, lahko na strani strežnika identificiramo s pomočjo naslova IP, ki se v podatkovni bazi strežnika osveži ob vsakem dostopu naprave do strežnika. Funkcija pošiljanja sporočil omogoča opozarjanje uporabnikov na detektirano prisotnost botnetov na mobilni napravi iz strani omrežnega detektorja botnetov.



Slika 5.1: Prikaz stanja detektorja.

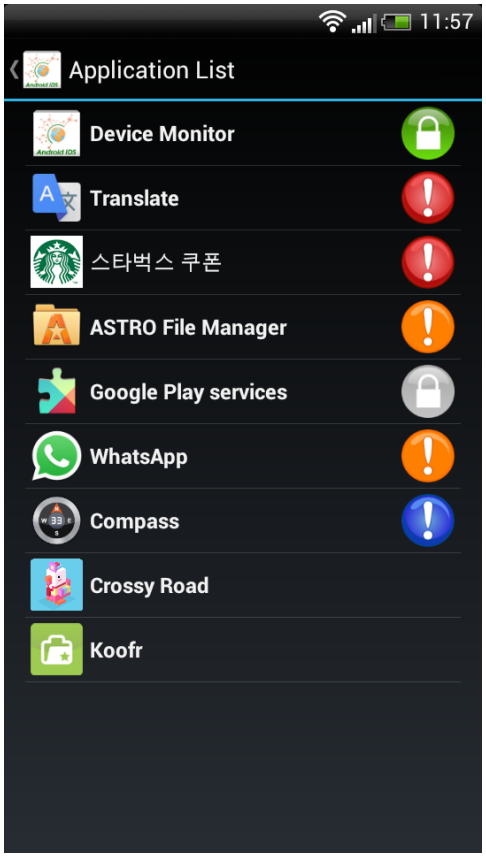


Slika 5.2: Prikaz podrobnosti o zaznanem dogodku.

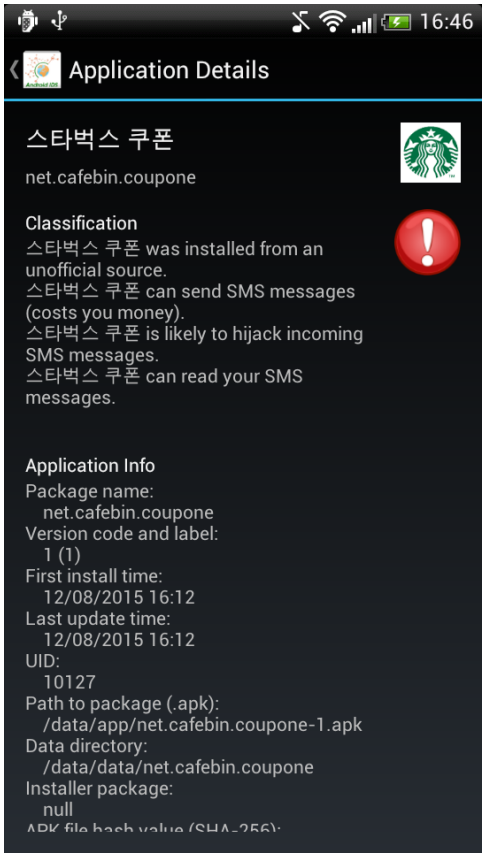
GCM Server ponuja spletni uporabniški vmesnik s prikazom seznama mobilnih naprav in sporočenih dogodkov. Omogoča tudi urejanje belega seznama certifikatov aplikacij in seznama zgoščenih vrednosti datotek zlonamernih aplikacij. Zaslonski prikaz seznama mobilnih naprav, povezanih na strežnik, je prikazan na sliki 5.5, prikaz dogodkov pa na sliki 5.6.

5.5 Testiranje

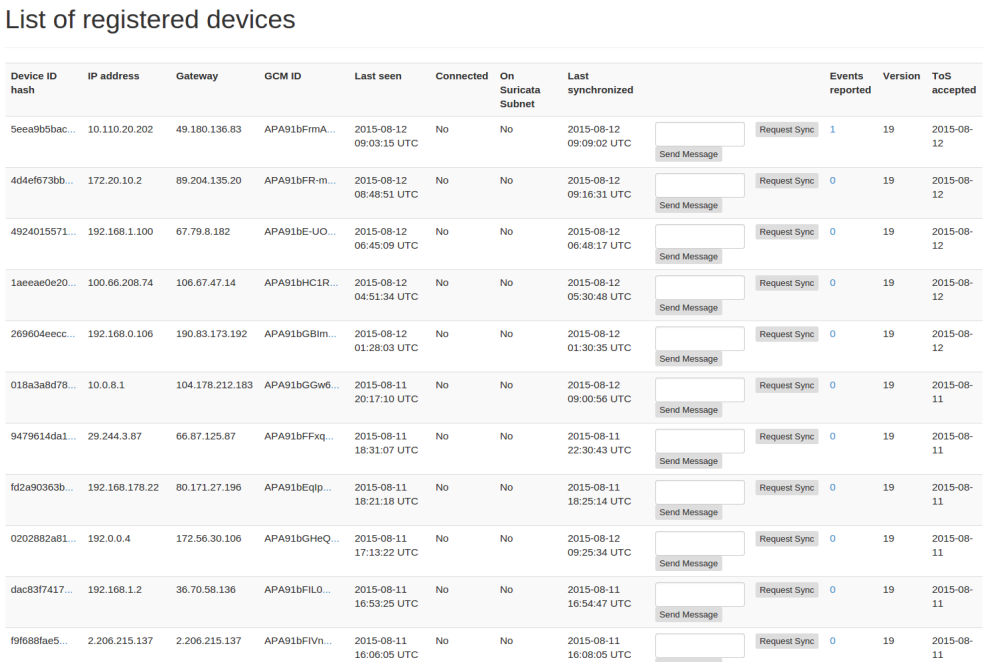
Za namen testiranja in demonstracije delovanja detektorja zlonamerne programske opreme na operacijskem sistemu Android smo na testno mobilno napravo poleg našega detektorja namestili dve aplikaciji, ki sta pri detek-



Slika 5.3: Seznam aplikacij z njihovimi ocenami.



Slika 5.4: Prikaz podrobnosti o aplikaciji.


List of registered devices

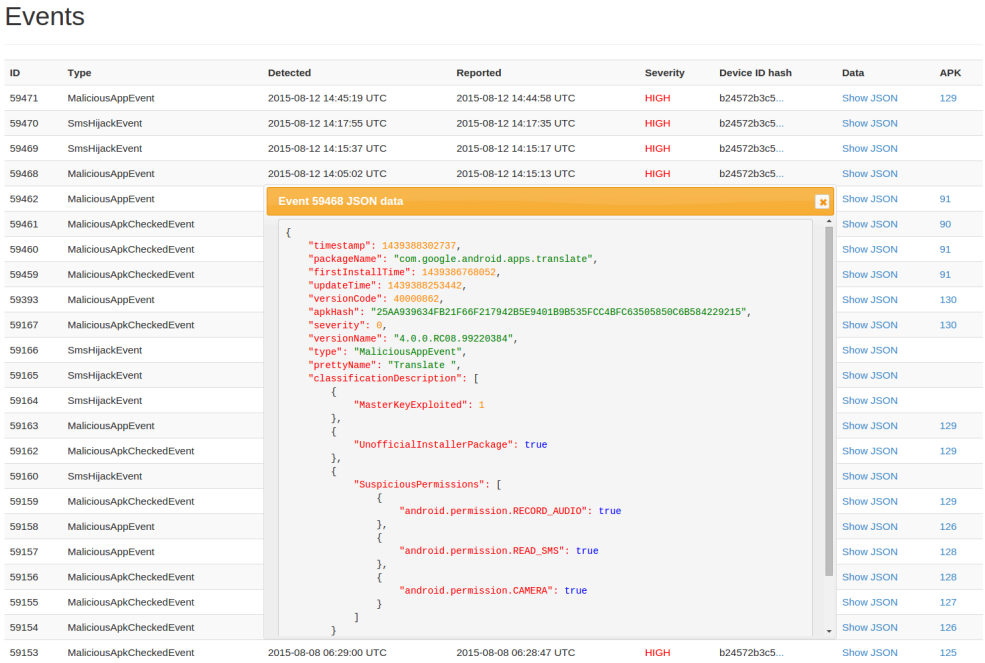
Device ID hash	IP address	Gateway	GCM ID	Last seen	Connected	On Suricata Subnet	Last synchronized		Events reported	Version	ToS accepted
5eea9b5bac...	10.110.20.202	49.180.136.83	APA91bFrmA...	2015-08-12 09:03:15 UTC	No	No	2015-08-12 09:09:02 UTC	Request Sync Send Message	1	19	2015-08-12
4d4ef673bb...	172.20.10.2	89.204.135.20	APA91bFR-m...	2015-08-12 08:48:51 UTC	No	No	2015-08-12 09:16:31 UTC	Request Sync Send Message	0	19	2015-08-12
4924015571...	192.168.1.100	67.79.8.182	APA91bE-UO...	2015-08-12 06:45:09 UTC	No	No	2015-08-12 06:48:17 UTC	Request Sync Send Message	0	19	2015-08-12
1aeeae0e20...	100.66.208.74	106.67.47.14	APA91bHC1R...	2015-08-12 04:51:34 UTC	No	No	2015-08-12 05:30:48 UTC	Request Sync Send Message	0	19	2015-08-12
269604eecc...	192.168.0.106	190.83.173.192	APA91bGBIm...	2015-08-12 01:28:03 UTC	No	No	2015-08-12 01:30:35 UTC	Request Sync Send Message	0	19	2015-08-12
018a3a8d78...	10.0.8.1	104.178.212.183	APA91bGGw6...	2015-08-11 20:17:10 UTC	No	No	2015-08-12 09:00:56 UTC	Request Sync Send Message	0	19	2015-08-11
9479614da1...	29.244.3.87	66.87.125.87	APA91bFFxq...	2015-08-11 18:31:07 UTC	No	No	2015-08-11 22:30:43 UTC	Request Sync Send Message	0	19	2015-08-11
fd2a90363b...	192.168.178.22	80.171.27.196	APA91bEqIp...	2015-08-11 18:21:18 UTC	No	No	2015-08-11 18:25:14 UTC	Request Sync Send Message	0	19	2015-08-11
0202882a81...	192.0.0.4	172.56.30.106	APA91bGHeQ...	2015-08-11 17:13:22 UTC	No	No	2015-08-12 09:25:34 UTC	Request Sync Send Message	0	19	2015-08-11
dac83f7417...	192.168.1.2	36.70.58.136	APA91bFIL0...	2015-08-11 16:53:25 UTC	No	No	2015-08-11 16:54:47 UTC	Request Sync Send Message	0	19	2015-08-11
f9f688fae5...	2.206.215.137	2.206.215.137	APA91bFlVn...	2015-08-11 16:06:05 UTC	No	No	2015-08-11 16:08:05 UTC	Request Sync	0	19	2015-08-11

Slika 5.5: Prikaz seznama mobilnih naprav.

torju sprožili zaznavo nevarnih dogodkov.

5.5.1 Detekcija ugrabitve sporočila SMS

Prvi primer zlonamerne aplikacije smo pridobili iz spletne zbirke zlonamernih programov Contagio Dump [35]. Gre za aplikacijo, ki ugrablja prejeta sporočila SMS in jih pošilja napadalcu. Opis zlonamernega programa najdemo v [41], zaznan pa je bil februarja 2013. Izdaja se za aplikacijo, ki nudi kupone za nakup kave v priljubljeni kavarni ter cilja na južnokorejske uporabnike. Uporabniški vmesnik je v korejskem jeziku, grafična ikona aplikacije pa je logotip znane kavarne. Način širjenja zlonamerne aplikacije v opisu ni omenjen. Ob zagonu aplikacija pokaže obvestilo, da je strežnik trenutno nedosegljiv in prosi uporabnika, da z zagonom poskusi kasneje. Cilj tega obvestila je prepričati uporabnika, da gre za legitimno aplikacijo, ki ima le trenutne težave. Ugotovili smo, da posredovanje sporočil napadalcu ne deluje več, saj strežniki DNS ne razrešijo naslova spletne domene strežnika C&C.



Slika 5.6: Prikaz seznama zaznanih dogodkov. V ospredju so prikazane podrobnosti o dogodku zaznave zlonamerne aplikacije.

Device Monitor je aplikacijo zaznal kot nevarno zaradi visoke prioritete sprejemnika sporočil SMS. Prikaz podrobnosti o aplikaciji je prikazan na sliki 5.4. Poleg branja prejetih sporočil SMS ima aplikacija tudi pravico do pošiljanja sporočil SMS. Pošiljanja sporočil z našim detektorjem ne moremo zaznati, s spremljanjem porabe denarja na računu smo ugotovili, da zlonameren program ni poslal nobenega sporočila. Na okuženo napravo smo poslali sporočilo SMS, to pa se ni pojavilo med prejetimi sporočili. Zlonameren program je sporočilo zadržal, naš detektor pa je zaznal dogodek ugrabitve sporočila SMS. Na sliki 5.2 je prikazan zaslonski posnetek obvestila o zaznanem dogodku.

Na testni napravi smo imeli nameščen operacijski sistem Android verzije 4.0.3, ki še omogoča skrivanje sporočil pred uporabnikom. Na novejših verzijah sistema (od 4.4 dalje) bi zlonamerna aplikacija še vedno lahko brala in posredovala vsebino sporočil, teh pa ne bi mogla skrivati pred uporabnikom, razen, če bi jo uporabnik nastavil za privzetega prejemnika sporočil SMS.

Pošiljanje in sprejemanje sporočil SMS je pogosto prisotna funkcija v zlonamernih mobilnih aplikacijah, pogosta pa je tudi nastavitev visoke prioritete sprejemnika sporočil. Zaradi tega se opozarjanje na aplikacije z nastavljeno visoko prioriteto sprejemnika sporočil SMS izkaže kot dober indikator zlonamerne programske opreme. Opazili pa smo tudi nekaj legitimnih programov, ki imajo nastavljeno najvišjo prioriteto sprejemnika sporočil, na primer Googlova aplikacija za sporočila *Messenger* (Sporočanje). Sklepamo, da je Google s to nastavitvijo želel preprečiti ugrabitve sporočil SMS s strani zlonamernih aplikacij. Kljub temu Device Monitor Googlovih aplikacij ne zazna kot nevarne, ker je njihov certifikat vsebovan na našem belem seznamu.

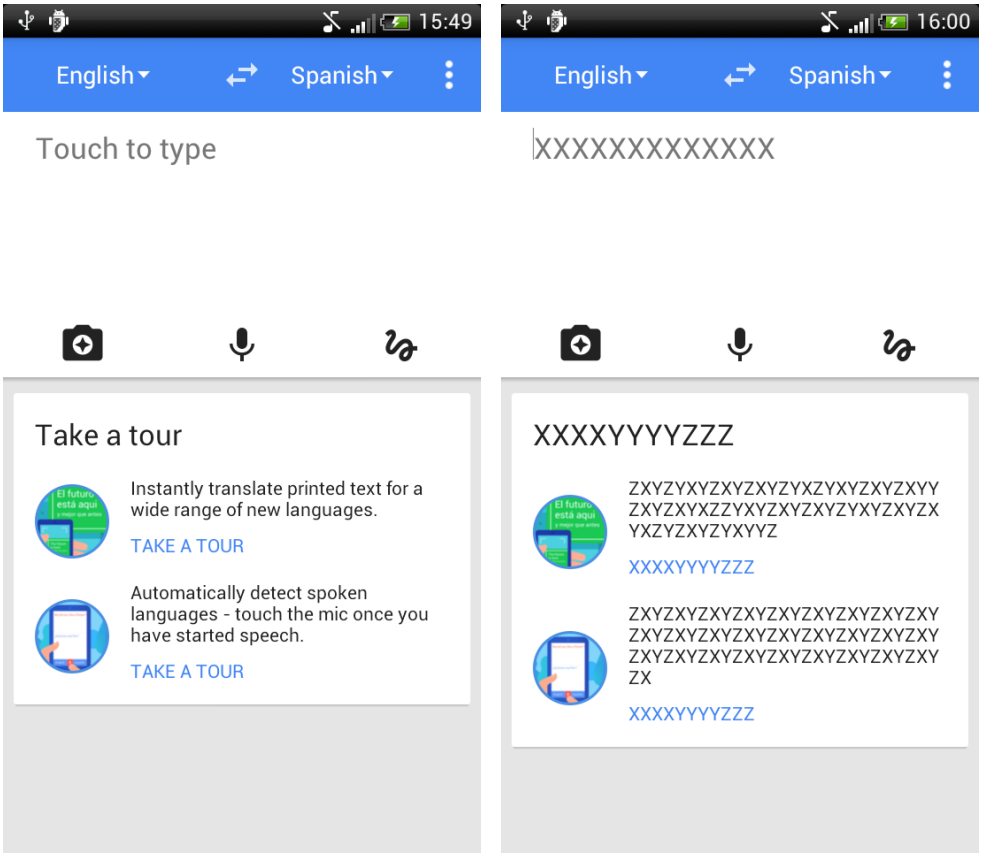
5.5.2 Detekcija izkoriščanja ranljivosti Master Key

V zbirkah zlonamernih programov nismo našli resničnih zlonamernih aplikacij, ki bi uporabljale ranljivost Master Key, zato smo primer uporabe te ranljivosti naredili sami. Iz uradne trgovine Android smo prenesli aplikacijo Prevajalnik (*Google Translate*) in jo namestili na testno napravo.

Namestitveni paket aplikacije smo nato spremenili tako, da smo v datoteki `resources.arsc`, ki vsebuje različne konstante in tekstovne nize, spremenili nekaj nizov, ki se prikažejo v uporabniškem vmesniku. Spremenjeno datoteko smo z enakim imenom dodali v paket APK poleg originalne. Z običajnimi orodji za urejanje paketov ZIP ni mogoče dodati datoteke z imenom, ki v paketu že obstaja. Zato smo dodali datoteko s spremenjenim imenom ter v paketu ZIP ročno spremenili ime datoteke v centralni in lokalni glavi.

Da bi dosegli večjo spremembo v delovanju aplikacije, bi lahko na ta način spremenili katerokoli datoteko v paketu APK, vključno z javansko kodo aplikacije v datoteki `classes.dex`. Sprememba nizov je enostavnejša, vendar zadošča za demonstracijo uporabe ranljivosti Master Key.

Spremenjen namestitveni paket smo namestili na testno napravo kot posodobitev originalne aplikacije. Slika 5.7 prikazuje uporabniški vmesnik originalne in spremenjene aplikacije. Po namestitvi je naš detektor zaznal nevarno aplikacijo ter prikazal opozorilo uporabniku. Dogodek, ustvarjen ob tej zaznavi je prikazan na uporabniškem vmesniku strežnika na sliki 5.6.



Slika 5.7: Zaslonska posnetka izvorne in spremenjene aplikacije.

Poglavje 6

Sklepne ugotovitve

V okviru tega magistrskega dela smo spoznali področje botnetov, njihove funkcije in načine detekcije botnetov. Raziskali smo tudi pojav botnetov in druge zlonamerne programske opreme na mobilnih napravah s poudarkom na operacijskem sistemu Android. Implementirali smo detektor botnetov iz omrežnega prometa, ki deluje s pomočjo strojnega učenja, in ga testirali na omrežnem prometu botnetov, pridobljenem iz zbirke podatkov o zlonamernih programih, ter na primeru omrežja, ki smo ga postavili sami. Razvili smo tudi detektor zlonamerne programske opreme za operacijski sistem Android, ki se osredotoča na grožnjo botnetov. V sodelovanju s spletnim strežnikom in podatkovnimi bazami zlonamernih virov lahko s spremljanjem več metrik zazna zlonamerne aplikacije in opozarja na grožnje.

Testiranje detektorja botnetov na omrežnem prometu je pokazalo, da je uporabljena metoda sposobna ločiti omrežni promet botnetov od legitimnega prometa, vendar ne dovolj učinkovito, da bi tovrsten detektor lahko uporabljali kot primarno zaščito proti botnetom. Pri primerjavi rezultatov z literaturo smo ugotovili, da pri testiranju na pridobljenem zajetem prometu botnetov dosegamo podobne rezultate. Glede na originalen članek smo uvedli dodaten atribut, s čimer smo nekoliko izboljšali uspešnost detekcije. Ugotovili smo tudi, da detektor uspešno zaznava zajet promet različic botnetov, katerih promet ni bil prisoten v učni množici. Naš dodaten prispevek

je testiranje na virtualnem omrežju z delujočimi primerki botnetov, to pa je pokazalo precej slabše rezultate.

Pri razvoju detektorja botnetov smo upali na splošno rešitev, kjer bi bil zgrajen model strojnega učenja za uporabo pri klasifikaciji uporaben dolgotrajno in v različnih omrežjih. Menimo, da je uspešnost detektorja, ki smo jo dosegli s testiranjem na virtualnih računalnikih, preslaba za praktično uporabo. Ocenjujemo, da bi bilo za boljšo uspešnost detekcije potrebno redno posodabljati model strojnega učenja. To pa je pomanjkljivost, kateri smo se želeli izogniti in je sicer značilna za detektorje, ki delujejo na podlagi zaznave znanih vzorcev.

Možna dodatna raziskava bi bila določanje potrebne pogostosti posodabljanja modela klasifikatorja z ugotavljanjem sposobnosti detekcije botnetov v odvisnosti od sprememb v njihovem delovanju. Možnost za izboljšavo detektorja bi bila tudi implementacija sistema za avtomatsko opozarjanje (na primer preko elektronske pošte) ob zaznavi malignega omrežnega prometa. Ker se pri detekciji pojavljajo tudi lažni pozitivni primeri, bi bilo smiselno proučiti možnost proženja takih alarmov ob presežku nastavljive meje števila zaznav malignih omrežnih tokov ali deleža malignih omrežnih tokov na posameznem računalniku v omrežju. Možno bi bilo obveščanje uporabnikov mobilnih naprav, na katerih bi omrežni detektor zaznal prisotnost botnetov, z uporabo funkcije pošiljanja sporočil mobilnim napravam. Dodatna možnost nadaljnjega razvoja detektorja je funkcija sprotnega učenja, kjer bi lahko nadzornik omrežja označeval že uvrščene omrežne tokove kot napačno klasificirane, kar bi za detektor pomenilo nove učne primere in sčasoma izboljšalo uspešnost detekcije.

Prvotno smo nameravali mobilne botnete zaznavati z uporabo izdelanega omrežnega detektorja, pri čemer bi k zanesljivosti detektiranih primerov pripomogel mobilni detektor s poročili o grožnjah in ocenami nevarnosti aplikacij na mobilnih napravah. Po raziskavi razlik med klasičnimi botneti na osebnih računalnikih in botneti na mobilnih napravah ter testiranju omrežnega detektorja smo ocenili, da detekcija botnetov na mobilnih napravah s tovrstnim

omrežnim detektorjem ni mogoča. Razvita mobilna aplikacija za detekcijo zlonamernih programov samostojno dobro zaznava grožnje na mobilnih napravah, in zato dobro nadomešča prvotno zamišljeno dodatno funkcionalnost omrežnega detektorja.

Naša mobilna aplikacija za detekcijo zlonamernih programov Device Monitor je edina nam znana aplikacija, ki v eni rešitvi ponuja zaznavo dostopov do nevarnih virov in zaznavo izkoriščanja nekaterih programskih ranljivosti ter edina, ki omogoča detekcijo ugrabitev sporočil SMS. Detekcija ugrabitev sporočil SMS in opozarjanje na aplikacije z visoko prioriteto sprejemnika sporočil SMS sta se pri testiranjih izkazali kot dober indikator zlonamernih aplikacij. Mobilno aplikacijo smo ponudili v brezplačen prenos uporabnikom uradne spletne trgovine Android, kjer je v približno enem letu od izdaje aplikacijo preneslo 1400 uporabnikov, trenutno pa jo uporablja približno 400 uporabnikov. Device Monitor je leta 2014 prejel nagrado združenja IPACSO za inovativen izdelek na področju računalniške varnosti [42].

Možnost izboljšave mobilne aplikacije je posodobitev grafičnega uporabniškega vmesnika za boljšo uporabniško izkušnjo (pri dosedanjem razvoju se s tem nismo ukvarjali). Aplikacija je zasnovana za enostavno nadgraditev sposobnosti detekcije z dodatnimi moduli, kar bi bilo potrebno ob odkritju novih varnostnih ranljivosti.

Literatura

[1] K. Singh, S. C. Guntuku, A. Thakur, C. Hota, Big Data Analytics Framework for Peer-to-Peer Botnet Detection Using Random Forests, Information Sciences 278 (2014) str. 488–497.

[2] G. Gu, R. Perdisci, J. Zhang, W. Lee, BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-independent Botnet Detection, v zborniku Proceedings of the 17th USENIX Security Symposium, SS'08, USENIX Association, Berkeley, CA, USA (2008) str. 139–154.

[3] W. Lu, G. Rammidi, A. A. Ghorbani, Clustering Botnet Communication Traffic Based on n-gram Feature Selection, Computer Communications 34 (3) (2011) str. 502–514.

[4] J. Zhang, R. Perdisci, W. Lee, X. Luo, U. Sarfraz, Building a Scalable System for Stealthy P2P-Botnet Detection, IEEE Transactions on Information Forensics and Security 9 (1) (2014) str. 27–38.

[5] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani, Crowdroid: Behavior-based Malware Detection System for Android, v zborniku Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11, ACM, New York, NY, USA (2011) str. 15–26.

[6] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss, Andromaly: A Behavioral Malware Detection Framework for Android Devices, Journal of Intelligent Information Systems 38 (1) (2012) str. 161–190.

[7] X. Wang, D. Ramsbrock, Chapter 8 - The Botnet Problem, v zborniku J. R. Vacca (Ed.), Computer and Information Security Handbook, Morgan Kaufmann, Boston (2009) str. 119 – 132.

[8] P. Bächer, T. Holz, M. Kötter, G. Wicherski, Know your Enemy: Tracking Botnets, dostopno na http://honeynet.org/papers/bots (pridobljeno 4.7.2015).

[9] P. Wood, et al., Symantec Global Internet Security Threat Report, Tech. Rep. 19, Symantec Corporation, dostopno na http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf (pridobljeno 4.7.2015).

[10] J. Isacenkova, O. Thonnard, A. Costin, D. Balzarotti, A. Francillon, Inside the SCAM Jungle: A Closer Look at 419 Scam Email Operations, v zborniku Security and Privacy Workshops (SPW), IEEE (2013) str. 143–150.

[11] B. Cruz, et al., McAfee Labs Threats Report, Tech. rep., McAfee Labs, dostopno na http://www.mcafee.com/mx/resources/reports/rp-quarterly-threat-q1-2014.pdf (pridobljeno 8.7.2015).

[12] V. Chebyshev, R. Unuchek, The Enemy on your Phone, Kaspersky Lab, dostopno na https://securelist.com/analysis/publications/68916/the-enemy-on-your-phone (pridobljeno 14.7.2015).

[13] J. Canavan, The Evolution of Malicious IRC Bots, v zborniku VB2005 Conference, Virus Bulletin (2005).

[14] C. Schiller, J. Binkley, Botnets: The Killer Web App, Syngress (2007).

[15] S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, Botnets: A Survey, Computer Networks 57 (2) (2013) str. 378–403

[16] Eggheads Development Team: Eggdrop, dostopno na http://www.eggheads.org (pridobljeno 9.7.2015).

[17] P. Royal, On the Kraken and Bobax Botnets, Tech. rep., Damballa, Inc., dostopno na www.damballa.com/downloads/r_pubs/Kraken_Response.pdf (pridobljeno 10.7.2015).

[18] Snort, dostopno na http://www.snort.org (pridobljeno 12.7.2015).

[19] Suricata, dostopno na http://suricata-ids.org (pridobljeno 12.7.2015).

[20] R. Nigam, A Timeline of Mobile Botnets, v zborniku BotConf, FortiGuard Labs, Fortinet (2014).

[21] V. Chebyshev, R. Unuchek, Mobile Malware Evolution: 2013, Kaspersky Lab (2014), dostopno na https://securelist.com/analysis/kaspersky-security-bulletin/58335/mobile-malware-evolution-2013 (pridobljeno 15.7.2015).

[22] Y. Zeng, K. G. Shin, X. Hu, Design of SMS Commanded-and-controlled and P2P-structured Mobile Botnets, v zborniku Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC '12, ACM, New York, NY, USA (2012) str. 137–148.

[23] W. Chen, P. Gong, L. Yu, G. Yang, An Adaptive Push-styled Command and Control Mechanism in Mobile Botnets, Wuhan University Journal of Natural Sciences 18 (5) (2013) str. 427–434.

[24] Android Open Source project: Seznam aplikacijskih pravic na OS Android, dostopno na http://developer.android.com/reference/android/Manifest.permission.html (pridobljeno 17.7.2015).

[25] F. Buchholz, The Structure of a PKZip file, dostopno na https://users.cs.jmu.edu/buchhofp/forensics/formats/pkzip.html (pridobljeno 6.8.2015).

[26] J. Forristal, Android: One Root to Own Them All, v zborniku Black Hat USA 2013, dostopno na http://bluebox.com/wp-content/uploads/2013/08/Forristal_Blackhat-US2013_-final.pdf (pridobljeno 6.8.2015).

[27] J. Freeman, Exploit (& Fix) Android "Master Key", dostopno na http://www.saurik.com/id/17 (pridobljeno 6.8.2015).

[28] J. Freeman, Android Bug Superior to Master Key, dostopno na http://www.saurik.com/id/18 (pridobljeno 6.8.2015).

[29] J. Freeman, Yet Another Android Master Key Bug, dostopno na http://www.saurik.com/id/19 (pridobljeno 6.8.2015).

[30] Google: Android Developers: Dashboards, dostopno na https://developer.android.com/about/dashboards/index.html (pridobljeno 7.8.2015).

[31] J. Forristal, Android Fake ID Vulnerability, v zborniku Black Hat USA 2014, dostopno na https://www.blackhat.com/docs/us-14/materials/us-14-Forristal-Android-FakeID-Vulnerability-Walkthrough.pdf (pridobljeno 8.8.2015).

[32] Sly Technologies: jNetPcap, dostopno na http://jnetpcap.com (pridobljeno 23.7.2015).

[33] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, B. Zupan, Orange: Data Mining Toolbox in Python, Journal of Machine Learning Research 14 (2013) str. 2349–2353.

[34] Elasticsearch, dostopno na https://www.elastic.co (pridobljeno 27.7.2015).

[35] M. Parkour, Contagio Dump, dostopno na http://contagiodump.blogspot.com (pridobljeno 27.7.2015).

[36] I. Kononenko, M. Robnik Šikonja, Inteligentni sistemi, Založba FE in FRI (2010).

[37] Wikipedia, F1 Score, dostopno na http://en.wikipedia.org/wiki/F1_score (pridobljeno 29.7.2015).

[38] XLAB: Device Monitor, dostopno na http://devicemonitor.eu (pridobljeno 4.8.2015).

[39] Advanced Cyber Defence Centre, dostopno na http://acdc-project.eu (pridobljeno 4.8.2015).

[40] Wikipedia, Bloom Filter, dostopno na http://en.wikipedia.org/wiki/Bloom_filter (pridobljeno 10.8.2015).

[41] M. Zhang, SMS Trojan Targets South Korean Android Devices, dosegljivo na https://blogs.mcafee.com/mcafee-labs/sms-trojan-targets-south-korean-android-devices (pridobljeno 12.8.2015).

[42] IPACSO Cyber Security & Privacy Innovation Awards 2014, dostopno na http://ipacso.eu/11-uncategorized/127-ipacso-cyber-security-privacy-innovation-awards-2014-finals.html (pridobljeno 16.8.2015).

Dodatek A

Tabela malignih omrežnih tokov

V pričujoči tabeli so našteti omrežni zajemi botnetov, pridobljeni iz množice podatkov Contagio Dump. Poleg imena je pri vsakem od primerov zapisano število omrežnih tokov, ki smo jih pridobili iz omrežnega zajema. Primeri so razdeljeni na učno in testno množico. Množici smo združili z zajetimi benignimi omrežnimi tokovi ter uporabili za učenje in testiranje klasifikatorjev, uporabljenih za omrežno detekcijo botnetov.

Naziv	Število omrežnih tokov
Učna množica (64 primerov)	
BIN_8202_6d2c12085f0018daeb9c1a53e53fd4d1	6
BIN_Alinav5.3_4C754150639AA3A86CA4D6B6342820BE	4
BIN_Andromeda_85F908A5BD0ADA2D72D138E038AECC7D_2013-04	8
BIN_ArdamaxKeylogger_E33AF9E602CBB7AC3634C2608150DD18	11
BIN_Bitcoinminer_12E717293715939C5196E604591A97DF-2013-05-12	10
BIN_Cutwail-Pushdo(2)_582DE032477E099EB1024D84C73E98C1	1888

BIN_Darkmegi_2012-04	138
BIN_DarknessDDoS_v8g_F03Bc8Dcc090607F38Ffb3A36Ccacf48_2011-01	11
BIN_dirtjumper_2011-10	3001
BIN_DNSWatch_protux_4F8A44EF66384CCFAB737C8D7ADB4BB8_2012-11	5
BIN_Drowor_worm_0f015bb8e2f93fd7076f8d178df2450d_2013-04	1
BIN_Enfal_Lurid_0fb1b0833f723682346041d72ed112f9_2013-01	7
BIN_Gh0st-gif_f4d4076dff760eb92e4ae559c2dc4525	5
BIN_Gh0st_variant-v2010_B1D09374006E20FA795B2E70BF566C6D_2012-08	53
BIN_Googledocs_macadocs_2012-12	7
BIN_Imaut_823e9bab188ad8cb30c14adc7e67066d	5
BIN_IXESHE_0F88D9B0D237B5FCDC0F985A548254F2-2013-05	5
BIN_LetsGo_yahoosb_b21ba443726385c11802a8ad731771c0_2011-07-19	2
BIN_Mediana_0AE47E3261EA0A2DBCE471B28DFFE007_2012-10	62
BIN_Nettravler_1f26e5f9b44c28b37b6cd13283838366	19
BIN_njRAT-Backdoor.LV_660709324acb88ef11f71782af28a1f0	1
BIN_NJRat-BackdoorLV_6fd868e68037040c94215566852230ab_CNtiananmensquare	1
BIN_Ponyloader-Zeus_B10393BE747143F3B4622E9E5277FFCE	58
BIN_Ramnitpcap_2012-01	2158
BIN_Reedum_0ca4f93a848cf01348336a8c6ff22daf_2013-03	2
BIN_RssFeeder_68EE5FDA371E4AC48DAD7FCB2C94BAC7-2012-06	20
BIN_sality_CEAF4D9E1F408299144E75D7F29C1810	13
BIN_Sanny-Daws_338D0B855421867732E05399A2D56670_2012-10	2
BIN_Stabuniq_F31B797831B36A4877AA0FD173A7A4A2_2012-12	5
BIN_Taidoor_40D79D1120638688AC7D9497CC819462_2012-10	33
BIN_Taidoor_46ef9b0f1419e26f2f37d9d3495c499f	176
BIN_Taleret.E_5328cfcb46ef18ecf7ba0d21a7adc02c	13

BIN_Tapaoux_60AF79FB0BD2C9F33375035609C931CB_winver_2011-08-23	3
BIN_Tbot_23AAB9C1C462F3FDFDDD98181E963230_2012-12	44
BIN_Tbot_2E1814CCCF0C3BB2CC32E0A0671C0891_2012-12	99
BIN_Tbot_5375FB5E867680FFB8E72D29DB9ABBD5_2012-12	70
BIN_Tbot_FC7C3E087789824F34A9309DA2388CE5_2012-12	218
BIN_Tinba_2012-06	5
BIN_torpigminiloader_011C1CA6030EE091CE7C20CD3AAECFA0	15
BIN_TrojanCookies_840BD11343D140916F45223BA05ABACB_2012_01	739
BIN_TrojanPage_86893886C7CBC7310F7675F4EFDE0A29	11
BIN_UStealD_2b796f11f15e8c73f8f69180cf74b39d	3
BIN_Vobfus_634AA845F5B0B519B6D8A8670B994906_2012-12	4
BIN_Wordpress_Mutopy_Symmi_20A6EBF61243B760DD65F897236B6AD3-DeepEndR	170
BIN_Wordpress_Mutopy_Symmi_20A6EBF61243B760DD65F897236B6AD3-ShortRun	325
BIN_Xpaj_2012-05	4
BIN_ZeroAccess_3169969E91F5FE5446909BBAB6E14D5D_2012-10	240
BIN_ZeroAccess_Sirefef_C2A9CCC8C6A6DF1CA1725F955F991940_2013-08	903
BIN_ZeusGameover_2012-02	109
BitcoinMiner_F865C199024105A2FFDF5FA98F391D74	11
cryptolocker_9CBB128E8211A7CD00729C159815CB1C	5
EK_BIN_Blackhole_leadingto_Medfos_0512E73000BCCCE5AFD2E9329972208A_2013-04	11
EK_Blackhole_55A60EBB5EC6079C52CEDB6CB1DC48AD	1
EK_Smokekt150(Malwaredontneedcoffee)_2012-09	41
iMesh_7.1.0.x(IMWeb.dll_7.0.0.x)_Remote_Heap_Overflow_Exploit	1
Kelihos_C94DC5C9BB7B99658C275B7337C64B33	1833
Microsoft_SQL_Server_Distributed_Management_Objects_BoF_Exploit	1

Mswab_Yayih_FD1BE09E499E8E380424B3835FC973A8_2012-03	11
OSX_DocksterTrojan	15
PassAlert_B4A1368515C6C39ACEF63A4BC368EDB2-2013-05-13	3
purplehaze	7126
RTF_Mongall_Dropper_Cve-2012-0158_C6F01A6AD70DA7A554D48BDBF7C7E065_2013-01	7
Tijcont_845B0945D5FE0E0AAA16234DC21484E0	51
Yahoo_Music_Jukebox_2.2-AddImage()_ActiveX_Remote_BOF_Exploit(2)	1
Skupno v učni množici	19810
Testa množica (32 primerov)	
BIN_9002_D4ED654BCDA42576FDDFE03361608CAA_2013-01-30	1
BIN_ChePro_2A5E5D3C536DA346849750A4B8C8613A-1	2
BIN_CitadelUnpacked_2012-05	1
BIN_Cutwail_284Fb18Fab33C93Bc69Ce392D08Fd250_2012-10	109
BIN_Cutwail-Pushdo(1)_582DE032477E099EB1024D84C73E98C1	174
BIN_DNSChanger_2011-12	4
BIN_Gypthoy_3EE49121300384FF3C82EB9A1F06F288	4
BIN_IRCbot_c6716a417f82ccedf0f860b735ac0187_2013-04	2
BIN_Kelihos_aka_Nap_0feaaa4adc31728e54b006ab9a7e6afa	5
BIN_Kuluoz-Asprox_9F842AD20C50AD1AAB41F20B321BF84B	15966
BIN_Lader-dlGameoverZeus_12cfe1caa12991102d79a366d3aa79e9	27
BIN_Likseput_E019E37F19040059AB5662563F06B609_2012-10	4
BIN_LURK_AF4E8D4BE4481D0420CCF1C00792F484_20120-10	156
BIN_MatsnuMBRwiping_1B2D2A4B97C7C2727D571BBF9376F54F	9
BIN_Nitedrem_508af8c499102ad2ebc1a83fdbcefecb	10
BIN_PlugX_2ff2d518313475a612f095dd863c8aea	5
BIN_PowerLoader_4497A231DA9BD0EEA327DDEC4B31DA12_2013-05	5
BIN_SpyEye_2010-02	25
BIN_torpigminiloader_C3366B6006ACC1F8DF875EAA114796F0	14

BIN_ZeroAccess_Sirefef_29A35124ABEAD63CD8DB2BBB469CBC7A_2013-05	102
BIN_Zeus_b1551c676a54e9127cd0e7ea283b92cc-2012-04	17
Darkcomet_DC98ABBA995771480AECF4769A88756E	1
EK_Blackhole_Java_CVE-2012-4681_2012-08	4
EK_Blackholev1_2012-03	20
EK_Blackholev1_2012-08	47
EK_Blackholev2_2012-09	78
EK_popads_109.236.80.170_2013-08-13	3
HorstProxy_EFE5529D697174914938F4ABF115F762-2013-05-13	6
NUVICO_DVR_NVDV4__PdvrAtl_Module_(PdvrAt.DLL_1.0.1.25)_BoF_Exploit	1
PDF_CVE-2011-2462_Pdf_2011-12	57
RealPlayer_rmoc3260.dll_ActiveX_Control_Remote_Code_Execution_Exploit	1
Sejoong_Namo_ActiveSquare_6_NamoInstaller.dll-ActiveX_BoF_Exploit	1
Skupno v testni množici	16861
Skupaj	36671

Dodatek B

Rezultati eksperimentov na testni množici

Naslednja tabela prikazuje uspešnost klasifikatorjev v odvisnosti od parametrov algoritma naključnih gozdov, izmerjeno z eksperimenti na testni množici, ki so opisani v poglavju 4.5.2. Parameter število dreves je označen s črko n, y označuje minimalno število primerov v listih dreves, stolpec p2p pa pomeni prisotnost dodatnega parametra p2p pri klasifikaciji. Razlaga vrednosti matrike zmot in opisi mer uspešnosti so predstavljeni v poglavju 4.5.1. Klasifikatorji so v tabeli razporejeni po uspešnosti, merjeni z mero $F_{0,5}$.

n	y	p2p	TP	FP	FN	TN	*T*	*Senz*	*Spec*	*Prec*	*F*	$F_{0,5}$
40	2	Ne	16370	106	491	393447	99,85%	97,09%	99,973%	99,36%	98,21%	98,89%
40	3	Ne	16338	116	523	393437	99,84%	96,90%	99,971%	99,30%	98,08%	98,81%
60	3	Ne	16336	118	525	393435	99,84%	96,89%	99,970%	99,28%	98,07%	98,79%
50	3	Ne	16335	118	526	393435	99,84%	96,88%	99,970%	99,28%	98,07%	98,79%
45	3	Ne	16337	124	524	393429	99,84%	96,89%	99,968%	99,25%	98,06%	98,77%
50	1	Ne	16381	155	480	393398	99,85%	97,15%	99,961%	99,06%	98,10%	98,67%
40	1	Ne	16381	161	480	393392	99,84%	97,15%	99,959%	99,03%	98,08%	98,65%
50	3	Da	15991	86	870	393467	99,77%	94,84%	99,978%	99,47%	97,10%	98,50%
45	3	Da	15993	94	868	393459	99,77%	94,85%	99,976%	99,42%	97,08%	98,47%
40	3	Da	15981	92	880	393461	99,76%	94,78%	99,977%	99,43%	97,05%	98,46%

n	y	p2p	TP	FP	FN	TN	T	$Senz$	$Spec$	$Prec$	F	$F_{0,5}$
60	1	Ne	16239	163	622	393390	99,81%	96,31%	99,959%	99,01%	97,64%	98,46%
80	3	Da	15979	94	882	393459	99,76%	94,77%	99,976%	99,42%	97,04%	98,45%
40	10	Da	15969	92	892	393461	99,76%	94,71%	99,977%	99,43%	97,01%	98,45%
100	3	Da	15957	93	904	393460	99,76%	94,64%	99,976%	99,42%	96,97%	98,43%
50	5	Da	15924	87	937	393466	99,75%	94,44%	99,978%	99,46%	96,88%	98,41%
40	4	Da	15905	82	956	393471	99,75%	94,33%	99,979%	99,49%	96,84%	98,41%
30	3	Da	15947	95	914	393458	99,75%	94,58%	99,976%	99,41%	96,93%	98,40%
55	5	Da	15927	90	934	393463	99,75%	94,46%	99,977%	99,44%	96,89%	98,40%
45	5	Da	15929	91	932	393462	99,75%	94,47%	99,977%	99,43%	96,89%	98,40%
70	5	Da	15917	88	944	393465	99,75%	94,40%	99,978%	99,45%	96,86%	98,40%
60	5	Da	15916	89	945	393464	99,75%	94,40%	99,977%	99,44%	96,85%	98,39%
40	7	Da	15922	91	939	393462	99,75%	94,43%	99,977%	99,43%	96,87%	98,39%
200	5	Da	15900	86	961	393467	99,74%	94,30%	99,978%	99,46%	96,81%	98,39%
40	5	Da	15910	89	951	393464	99,75%	94,36%	99,977%	99,44%	96,84%	98,38%
40	5	Da	15910	89	951	393464	99,75%	94,36%	99,977%	99,44%	96,84%	98,38%
12	5	Da	15894	90	967	393463	99,74%	94,26%	99,977%	99,44%	96,78%	98,36%
40	2	Da	15860	81	1001	393472	99,74%	94,06%	99,979%	99,49%	96,70%	98,36%
30	5	Da	15901	93	960	393460	99,74%	94,31%	99,976%	99,42%	96,80%	98,35%
10	5	Da	15859	93	1002	393460	99,73%	94,06%	99,976%	99,42%	96,66%	98,30%
7	5	Da	15905	121	956	393432	99,74%	94,33%	99,969%	99,24%	96,73%	98,22%
3	5	Da	15905	127	956	393426	99,74%	94,33%	99,968%	99,21%	96,71%	98,19%
2	5	Da	15674	78	1187	393475	99,69%	92,96%	99,980%	99,50%	96,12%	98,12%
40	15	Da	15822	119	1039	393434	99,72%	93,84%	99,970%	99,25%	96,47%	98,12%
40	20	Da	15676	125	1185	393428	99,68%	92,97%	99,968%	99,21%	95,99%	97,90%
40	50	Da	15848	233	1013	393320	99,70%	93,99%	99,941%	98,55%	96,22%	97,60%
100	5	Da	15162	56	1699	393497	99,57%	89,92%	99,986%	99,63%	94,53%	97,53%
15	5	Da	15057	49	1804	393504	99,55%	89,30%	99,988%	99,68%	94,20%	97,41%
40	5	Da	15010	37	1851	393516	99,54%	89,02%	99,991%	99,75%	94,08%	97,41%
20	5	Da	14870	36	1991	393517	99,51%	88,19%	99,991%	99,76%	93,62%	97,21%
40	100	Da	15894	339	967	393214	99,68%	94,26%	99,914%	97,91%	96,05%	97,16%

n	y	p2p	TP	FP	FN	TN	T	$Senz$	$Spec$	$Prec$	F	$F_{0,5}$
50	1	Da	14476	134	2385	393419	99,39%	85,85%	99,966%	99,08%	92,00%	96,12%
40	1	Da	14416	136	2445	393417	99,37%	85,50%	99,965%	99,07%	91,78%	96,02%
30	1	Da	14404	137	2457	393416	99,37%	85,43%	99,965%	99,06%	91,74%	95,99%
40	4	Ne	14201	101	2660	393452	99,33%	84,22%	99,974%	99,29%	91,14%	95,86%
40	10	Ne	14199	133	2662	393420	99,32%	84,21%	99,966%	99,07%	91,04%	95,69%
20	5	Ne	13887	43	2974	393510	99,26%	82,36%	99,989%	99,69%	90,20%	95,67%
5	5	Da	13934	78	2927	393475	99,27%	82,64%	99,980%	99,44%	90,27%	95,56%
100	5	Ne	13813	42	3048	393511	99,25%	81,92%	99,989%	99,70%	89,94%	95,55%
40	5	Ne	14041	123	2820	393430	99,28%	83,28%	99,969%	99,13%	90,51%	95,49%
50	5	Ne	14024	123	2837	393430	99,28%	83,17%	99,969%	99,13%	90,45%	95,47%
5	5	Ne	13954	148	2907	393405	99,26%	82,76%	99,962%	98,95%	90,13%	95,22%
15	5	Ne	13814	122	3047	393431	99,23%	81,93%	99,969%	99,12%	89,71%	95,13%
40	500	Da	15900	858	961	392695	99,56%	94,30%	99,782%	94,88%	94,59%	94,76%
7	5	Ne	13560	140	3301	393413	99,16%	80,42%	99,964%	98,98%	88,74%	94,61%
200	5	Ne	13237	46	3624	393507	99,11%	78,51%	99,988%	99,65%	87,83%	94,56%
30	5	Ne	13005	47	3856	393506	99,05%	77,13%	99,988%	99,64%	86,95%	94,14%
12	5	Ne	12322	41	4539	393512	98,88%	73,08%	99,990%	99,67%	84,33%	92,91%
10	5	Ne	12236	34	4625	393519	98,86%	72,57%	99,991%	99,72%	84,01%	92,78%
3	5	Ne	12280	91	4581	393462	98,86%	72,83%	99,977%	99,26%	84,02%	92,55%