REPUBLIKA E SHQIPËRISË
UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT
DEPARTAMENTI I INXHINIERISË INFORMATIKE

MARIN ARANITASI

PËR MARJEN E GRADËS
"DOKTOR"
Në "TEKNOLOGJITË E INFORMACIONIT DHE KOMUNIKIMIT"
DREJTIMI "INXHINIERI INFORMATIKE"

DISERTACION

REALIZIMI NJË SHTRESE PERMANENTE NË KERNELIN E SISTEMIT OPERATIV ANDROID E CILA RREGJISTRON THIRRJET SISTEM TË APLIKACIONEVE.

Udhëheqës Shkencor

Prof. Asoc. Genti Daci

Tiranë
2015

REALIZIMI NJË SHTRESE PERMANENTE NË KERNELIN E SISTEMIT OPERATIV ANDROID E CILA RREGJISTRON THIRRJET SISTEM TË APLIKACIONEVE.

Disertacion

I paraqitur në Universitetin Politeknik të Tiranës

Për marrjen e gradës

"Doktor"

Në

"Teknologjitë e Informacionit dhe Komunikimit"

Drejtimi Inxhinieri Informatike

Nga

Z. Marin Aranitasi

2015

Disertacioni i shkruar nga

Marin Aranitasi

Master Shkencor, Universiteti Politeknik i Tiranës, Shqipëri, 2011

Diplomë e nivelit të parë, Universiteti Politeknik i Tiranës, Shqipëri, 2009

I aprovuar nga

_____ , Kryetari, i Jurisë së Disertacionit të Doktoraturës

_____ , Anëtarët i Jurisë së Disertacionit të Doktoraturës

_____ , Anëtarët i Jurisë së Disertacionit të Doktoraturës

_____, Anëtarët i Jurisë së Disertacionit të Doktoraturës

_____ , Anëtarët i Jurisë së Disertacionit të Doktoraturës

I pranuar nga

_____ , Dekan, Fakulteti i Teknologjisë së Informacionit

TABELA E PËRMBAJTJES

LISTA E FIGURAVE

LISTA E TABELAVE

Abstrakt

Në vitet e fundit përdorimi i paisjeve inteligjente mobile (të ashtëquajtura smartphone) është rritur në mënyrë dramatike. Çdo njeri mund të ketë një smartphone dhe ta përdori atë për nevojat e veta. Për shkak se firmat prodhuese të paisjeve mobile janë më shumë të shqetësuara ti bëjnë ato sa më shumë user friendly, çdo klient i tyre mendon se mund të përdori këto paisje thjeshtësisht. Rritja e përdorimit të smartphone mund të sjellë këto paisje dhe përdoruesit e tyrë në qëndër të vëmendjes së personave me qëllime të këqia. Numri i prodhuesve të viruseve për paisjet mobile, të qualtura malware, është rritur në mënyrë eksponenciale.

Në këtë disertacion ne propozojmë shtimin e një shtrese në sistemin operativ Android. Qëllimi i kësaj pune është rritja e sigurisë së paisjeve të lëvizsshme. Kjo do të realizohet duke bërë kapjen, memorizimin dhe dërgimin e thirrjeve sistem në një server të jashtëm. Zgjedhja e thirrjeve sistem si element kyç në zgjidhjen e propozuar është bërë duke studiuar karakteristikat e një numri shumë të madh të malware-ve që janë detektuar deri në ditët e sotme. Nga ky studim kemi parë që malwaret sa vijnë e sofistikohen dhe antiviruset dhe teknikat normale të detektimit të tyre nuk po japin rrezultat. Kjo vjen edhe si shkak i veçorive të paisjeve mobile të cilat nuk lejojnë skanime të tepërta apo programe shumë të rënda për analizmin e viruseve, për shkak të resurseve të limituara. Por sado të sofistikuara të jenë në nivel programimi këto malware, ato nuk mund të shmangin përdorimin e thirrjeve sistem për arritjen e qëllimeve të tyre të këqia.

Këtu na lindi idea e shtimit të një shtrese shtesë në sistemin operativ e cila do të bëjë vetëm kapjen dhe dërgimin e thirrjeve sistem dhe analiza e këtyre thirrjeve do të bëhët në një paisje të jashtme. Kështu eleminohet edhe përpunimi i të dhënave në paisjet e lëvizsshme.

Testimet janë bërë me malware realë të dhe nga rrezultate shihet që diferenca e përdorimit të thirrjeve sistem nga aplikacionet e virusuara në krahasim me ato zyrtare është shumë e madhe dhe si pasojë detektimi i tyre sipas kësaj teknike është i lartë.

Fjalë kyçe: Thirrje sistem, Sistem operatv, Android, Malware, Smarthphone

Falenderime

Në rradhë të parë dua të falenderoj ZOTIN, pa të Cilin nuk do të realizoja asgjë. Pa mbështetjen e TIJ gjithcka do të ishte e pamundur.

Në rradhë të dytë dua të falenderoj familjen time e cila është edhe shtylla e dytë më e fortë, pa mbështetjen e të cilës realizimi i këtij disertacioni nuk do të mund të realizohej. Mbështetja e tyre në të gjitha ciklet e studimit ka qenë pika ime më e fortë.

Dua të falenderoj përzemërsisht dhe të shpreh mirënjohjen time për udhëheqësin shkencor, Prof.Asoc Genti Daci, për ndihmën e madhe dhe mbështetjen e çmuar që më ofroi përgjatë gjithë studimit tim, produkt i shumë konsultimeve, këshillimeve praktike dhe mbështetje nga ana e tij.

Falenderoj dekanen e Fakultetit të Teknologjisë së Informacionit, Prof.Dr. Rozeta Miho, për mbështetjen dhe ndjekjen e mbarëvajtjes së ciklit të studimeve të doktoraturës.

Falenderoj përgjegjësin e Qëndrës për Kërkim dhe Zhvillim në TI, për interesimin e vazhdueshëm dhe mbështetjen konstante në realizimin e këtij punimi.

Gjithashtu dua të falenderoj të gjithë kolegët e Departamentit të Inxhinierisë Informatike, dhe në veçanti përgjegjësen e departamentit, Prof.Asoc Elinda Mece, për pozitivitetin e theksuar, komunikimin dashamirës dhe mbështetjen e vazhdueshme nga fillimi i ciklit të doktoraturës deri në fundin e tij.

Shpreh mirënjohjen time për rektorin e Universitetit Politeknik të Tiranës, Akad.Prof.Dr. Jorgaq Kaçani për përkrahjen dhe mbështetjen që ka dhënë, për zhvillimin e stafit akademik në përgjithësi dhe për mua në veçanti.

Marin Aranitasi
Tiranë, 2015

KAPITULLI 1

Hyrje

Në këtë kapitull përshkruhet në terma të përgjithshëm disa nga elementët kryesorë të këtij punimi. Në të do të jepën motivimi për realizimin e këtij disertacioni si dhe qëllimet e tij. Gjithashtu do të tregohet edhe struktura e disertacionit.

1.1 Motivimi i punimit

Raportet e fundit të rreziqeve (thread reports) të antiviruseve kryesore, japin alarmin që situata e sigurisë në paisje e lëvizsshme është kritike. Numri i njerzve të cilët përdorin telefonat inteligjentë (ose smartphone) është rritur shumë në vitet e fundit. Ndryshe nga fillmet e tyre, përdorimi i këtyre smartphone ka kaluar nga një mjet të cilin e kishin vetem pak persona, pjesa më e madhe e të cilëve ose ishin inxhiniera shumë të apasionuar të fushës, ose ishin pasanikë që e donin paisjen vetëm për arsye ego, në mjetin më të përdorur edhe për gjërat më elemntare të jetës tonë të përditshme. Të gjithë ne jemi koshient për lehtësirat në komunikim, dhe më gjërë, që na kanë sjellë këto paisje. Gjithashtu rritja e konkurencës në këtë sektor midis firmave gjigande botërore ka sjellë edhe një përmirësim të ndjeshëm të kapaciteteve dhe funksionaliteteve të tyre. Por si çdo gjë e mirë ka edhe anën e "errët" të saj. Rritja e popullaritetit të këtyre paisjeve sjell edhe rritjen e vëmendjes te to. Por jo gjithmonë kjo rritje vëmendjë vjen nga personat e duhur. Siç e thonë edhe raportet e përvitshme të antiviruse të ndryshme kemi një rritje eksponenciale të personave dashakeqës të cilët shkruajnë dhe injektojnë viruse nga më të ndryshmet. Qëllimet janë te ndryshme duke filluar nga vjedhja e informacioneve personale, vjedhja e të dhënave të paisjes edhe deri te më e keqja që është përfitimi financiar për sulmuesin duke vjedhur kredencialet e bankave ose duke ja konsumuar të gjitha kreditet përdoruesit nëpërmjet mashtrimeve kompjuterike të ndryshme [1]. Duke parë këtë gjëndjë mendova që tema

kryesore e ketij punimi do të ishte dhënia e një kontributi në rritjen e sigurisë së sistemeve të lëvizsshme, duke prezantuar një teknikë të re në detektimin e viruseve.

1.2 Qëllimi dhe kontributi i këtij punimi

Qëllimi kryesor i këtij punimi është prezantimi i një teknike të re e cila do të synojë rritjen e sigurisë së sistemeve operative mobile.

Qëllimet të tjera të këtij punimi janë:

- Vlerësimi i situatës aktuale të sigurisë së sistemeve operative mobile. Ky vlerësim do të bazohet në të dhëna të konfirmuara si nga kompanitë më të mëdhaja që lidhen me sigurinë kompjuterike si edhe nga firmat zotëruese të marketeve kryesore për shkarkimin e aplikacioneve.
- Studim i literaturës shkencore dhe teknike në lidhje me sistemet operative mobile.
- Njohja e teknikave të sulmeve të viruseve të viteve të fundit.
- Evidentimi i teknikave të deritanishme të mbrojtjes së sistemeve operative të smartphoneve

Kontributi i këtij disertacioni është :

- Propozim i një zgjidhje të re në lidhje rritjen e sigurisë së sisteme operative Android.
- Evidentimi i thirrjeve sistem si element kyç në detektimin e viruseve.
- Implementimi dhe integrimi në kernelin e sistemit operativ Android i një shtrese të re e cila do të shërbejë për monitorimin e thirrjeve sistem të aplikacioneve të ndryshme.
- Testimi i zhgjidhjes së dhënë me anë të përdorimit të viruseve dhe situatave reale.

1.3 Struktura e disertacionit

Kapitulli 1 jep në terma të përgjithsëm disa nga elementët kryesorë të këtij punimi. Në të prezantohet motivimi, qëllimet si dhe kontributet kryesore që janë dhënë në këtë disertacion.

Kapitulli 2 mbulon disa nga konceptet, termat, dhe metodologjitë e përdorura për realizimin e sigurisë kompjuterike. Këtu tegohet që njeriu është një element kyç si për implementimin e teknikave kryesore të sigurisë, si për prishjen totale të tyre, duke qenë një nga hallkat më të dobëta në zinxhirin e sigurisë. Gjithashtu në të prezantohen edhe disa nga grupet e lejeve të sistemit operativ android.

Kapitulli 3 prezanton punimet dhe realizimet konkrete kryesore në fushën e sigurisë së sistemeve operative mobile të viteve të fundit. Aty tegohen punimet e realizuar për mbrojten në nivel kerneli si dhe për mbrojtjen e të dhënave të përdoruesit.

Kapitulli 4 merret me malwaret, dmth me viruset e paisjeve mobile. Në fillim jepet një përshkrim i përgjithshëm mbi malwaret dhe pastaj tregohen qëllimet kryesore të sulmeve të tyre. Gjthashti në këtë kapitull tregohet edhe historia e evolimit të malwareve nga i pari që u detektua deri tek malwaret e sotme. Gjithasthu jepet edhe një parashikim sesi mund të jenë malwaret e së ardhmes.

Kapitulli 5 fokusohet në sulmet në sistemin operativ android. Aty jepen klasat e vektorëve të sulmeve si dhe metodologjitë kryesore të sulmeve që janë realizuar. Gjithashtu fokus kanë edhe qëllimet dashakeqëse të këtyre sulmeve, të cilët mund të shërbejnë si element gjatë analizës së ndërtimit të sistemeve të sigurisë për këto paisje.

Kapitulli 6 merret me sistemet dhe teknikat e mbrojtjes së sistemeve operative mobile. Në të paraqiten mekanizma ekzistues të cilët janë zhvilluar për të shmangur lloje të ndryshme të kërcënimeve ndaj smartphoneve. Aty paraqiten disa Intrusion Detection Systems dhe teknikat e imlementimit të tyre.

Kapitulli 7 tregon zgjidhje e propozuar dhe testimin e saj. Këtu tregohet llogjika dhe elementët kryesorë të zgjidhjes së propozuar. Jepet një bllokskemë e algoritmit të propozuar si dhe detaje teknike të implementimit. Gjithashtu në këtë kapitull realizohen eksperimente të shumta me malware reale për validimin e zgjidhjes tonë. Rrezultatet eksperimentale shafqen si në formë tabelore ashtu edhe në atë grafike, në mënyrë që të evidentohet efiçenca e zgjidhjes tonë.

Kapitulli 8 prezanton përfundimet dhe konkluzionet e arritura mbas realizimit të këtij punimi.

KAPITULLI 2

Bazat teorike të sigurisë

Siguria është e rëndësishme, dhe mungesa e saj rrezikon implikime financiare. Kjo pjesë mbulon disa nga konceptet, termat, dhe metodologjitë e përdorura për realizimin e sigurisë kompjuterike. Kur meren në konsiderate rrjetet, ata mund të shikohen nga përspektiva të ndryshme. Për shëmbull, menaxheri i lartë mund shikoj rrjetin si një mjet biznesi për të realizuar qëllimet e kompanisë. Teknikët e rrjetit (të paktën disa) mund të konsideronin rrjetin të jetë qëndra e universit. Përdoruesit mund të konsiderojnë rrjetin një mjet për për të kryer detyrat e tyre, Jo të gjithë përdoruesit e vlerësojmë rolin e tyre në mbajtjen e të dhënave të sigurta, dhe për fat të keq përdoruesit e rrjetit përfaqësojnë një dobësi të madhe, ata kanë username dhe passworde (ose kredencialet e tjera) që i lejojë atyre aksesin në rrjet. Nëse një përdorues është kompromet uar ose një individ i paautorizuar fiton akses, siguria e rrjetit mund te dështoj. Pra, një pikë e rëndësishme është se përdoruesit vetë paraqësin një rrezik për sigurinë dhe që trajnimi i përdoruesit është një pjesë e rëndësishme e politikave gjithëpërfshirëse të sigurisë [2].

2.1 Konfidencialiteti Integriteti dhe Disponueshmëria

Objektivat e sigurisë zakonisht përfshijnë tri konceptet themelore [2]:

1. *Konfidencialiteti*: Ka dy lloje të të dhënave: të dhënat në lëvizje dmth ato që lëvizin nëpër rrjet; dhe të dhëna të tjetra, kur të dhënat janë në media storage (server, lokal workstation, në një cloud, dhe kështu me radhë). Konfidencialiteti do të thotë se vetëm individët/ sistemet e autorizuar mund të shikjojnë informacione të klasifikuara. Kjo gjithashtu nënkupton se individët e paautorizuar nuk duhet të ketë asnjë lloj të aksesi në të dhëna. Sa i përket të dhënave në lëvizje, mënyra primare për të mbrojtur të dhënat është duke i enkriptuar ato para se të

dërgohen në rrjet. Një tjetër opsion është përdorimi i rrjeteve të ndara për transmetimin e të dhënave konfidenciale.

2. *Integriteti*: Integriteti i të dhënave do të thotë se ndryshimet e bëra në të dhëna bëhen vetëm nga individët/ sistemet e autorizuar. Korruptimi i të dhënave është një dështim për të ruajtur të dhënat e sigurta.

3. *Disponueshmëria*: Kjo vlen për sistemet dhe të dhënat. Nëse rrjeti ose të dhënat e tij nuk janë në dispozicion për përdoruesit kjo mund të ndodhë për shkak të një denial-of-service (DoS) attack ose ndoshta për shkak të një dështimi të përgjithshëm të rrjetit. Ndikimi mund të jetë i rëndësishëm për të kompanitë dhe përdoruesit të cilët mbështeten në këtë rrjet si një mjet biznesi. Dështimi i një rrjeti zakonisht barazohet me humbje të të dhënave.

2.2 Analiza e kostos së sigurisë

Inxhinierët që merren me sigurinë duhet të kuptojnë jo vetëm atë që ata mbrojnë, por edhe nga kush po e mbrojnë përdoruesin. Menaxhimi i rrezikut është i bazuar në parimet dhe konceptet specifike të lidhura me të dyja mbrojtjen dhe menaxhimin e sigurisë [2].

Çfarë është një asset? Kjo është diçka që është e vlefshme për një organizatë. Këto mund të jenë sende të prekshme (njërëz, kompjutera, dhe kështu me radhë), ose artikuj të paprekshme (të pronës intelektuale, informacioni i bazës së të dhënave, listat e kontaktit, te dhëna te kontabilitetit). Duke ditur asetet që jeni duke u përpjekur të mbroni dhe vlerat e tyre, vendndodhjen, dhe ekspozimin mund tju ndihmojë më shumë në mënyrë efektive te përcaktoni kohën dhe paratë që duhen shpenzuar për sigurimin e këtyre aseteve. [2]

Vulnerabiliteti (vulnerability) është një dobësi në një sistem apo në dizenjimin e tij. Dobësitë mund të gjenden në protokollet, sistemet operative, aplikacionet, dhe dizajne të sistemit. Dobësi të shumta zbulohen çdo ditë .[2]

Një kërcënim (threat) është çdo rrezik potencial i një aseti. Nëse një dobësi ekziston por nuk është shfrytëzuar ende, kërcënim është i fshehur dhe ende i pa realizuar. Nëse dikush është duke nisur një sulm kundër sistemit tuaj dhe me sukses akseson diçka ose komprometon sigurisë tuaj, kërcënim realizohet. Njësia që shfrytezon avantazhin e ndjeshmërisë se sistemit është i njohur si the threat agent ose threat vector. [2]

Një kundërmasë (countermeasure) është një garanci që në njëfarë mënyre e zbut një rrezik potencial. Ai e bën këtë duke reduktuar ose eliminuar vulnerabilitetin, ose të paktën zvogëlon gjasat që një agjent kërcënimi të shfrytëzoj rrezikun. Për shëmbull, ju mund të keni një makinë jo të lidhur në rrjetin tuaj, duke e bërë atë shumë të prekshme. Nëse kjo makinë është e palidhur në rrjet dhe pushon të ketë ndonjë ndërveprim të shkëmbimit të të dhënave me pajisje të tjera, ju keni zbutur me sukses të gjitha këto dobësi. Ka të ngjarë që makina nuk është më një aset, edhe pse; por ajo është më e sigurt. Vini re se limitime zbatohet për mënyrën se si ne të klasifikojë gjërat. Ne nuk shpenzojnë më shumë se vlera e asetit për ta mbrojtur atë. Nëse identifikohen të dhënat me vlerën më të madhe, automatikisht identifikohet ku do të jenë përpjekjet më të mëdha për ta bërë më të sigurtë këtë informacion. Përtej qëllim të caktuar të kompanisë në lidhje me vlerën e ndonjë të dhëne, entitete rregullatorë gjithashtu mund të jenë të përfshirë (rregulloret e qëverisë apo ligje, marrëveshje partnere biznesi, marrëveshjet kontraktuale, dhe kështu me radhë). Pranimi i rrezikut të plotë (the all-or-nothing approach) nuk është gjithmonë e pranueshme. Të njëjtat pajisje të sigurisë, të tilla si firewalls and intrusion prevention systems (IPS), mund të mbrojnë pajisje të shumta të njëjtën kohë, duke ofruar një përfitim kosto. Kurrë nuk mund të eleminohet plotësisht rreziku, kështu që duhet të gjendet një ekuiliber. [2]

Tabela 1. Përkufizimet kryesore të sigurisë

Asset	Një aset është një çështje që duhet të mbrohet dhe mund të përfshijnë pronën, njërëz, dhe informacion / të dhëna që kanë vlerë për të kompanisë/përdoruesit. Kjo përfshin artikuj të vlefshem të tilla si informacion mbi sekrete të tregtisë dhe reputacionin e kompanisë. Të dhënat mund të përfshijnë të dhënat e kompanisë, informacione klient, software, dhe kështu me radhë.
Vulnerability	Një vulnerability është një dobësi e shfrytëzueshme. Shfrytëzimi i saj mund të rezultojë në një sulm me qëllim të keq, ose ajo mund të jetë shkaktuar aksidentalisht për shkak të dështimit ose dobësive në politikën e sistemit, ose softwareve që ekzekutohen në rrjet.
Threat	Kjo është ajo nga e cila duhet të mbrohemi. Një kërcënim është çdo gjë që përpiqët të fitoj akses të paautorizuar që të ndërhyj, të shkatërroj, ose të dëmtojë një aset. Kërcënimet janë realizuar shpesh nëpërmjet një sulmi që përdor avantazhin e një dobësie ekzistuese. Kërcënimet sot vijnë në shumë varieteteve dhe të përhapet më shpejt se kurrë më parë.
Risk	Rreziku është potenciali për akses të paautorizuar të nderhyj, shkatërroj, apo dëmtimtoj e një aset. Nëse një kërcënim ekziston, por kundërmasat e duhura dhe mbrojtjet janë vendosur, reduktohet potenciali për kërcënimin të suksesshëm (duke zvogëluar riskun e përgjithshëm).
Countermeasure	Një kundërmasë është një pajisje ose proces që është zbatohet për të luftuar një kërcënim potencial, i cili në këtë mënyrë zvogëlon rrezikun.

2.3 Grupet e autorizimeve

Grupet e autorizimeve (Permissions groups) janë të dizejnuara për të treguar atë që një aplikacion do të jetë në gjendje për të aksesuar në paisjen tonë. Me Permissions groups, ne mund të shohim se çfarë aftësish apo informacioni një aplikacion mund të përdorë para se ta shkarkojmë atë.[5]

Është shumë e rëndëssishme që të shikojmë me detaje përmissions groups para se të shkarkojmë një aplikacion. Pasi ne i lejojmë një aplikacioni të aksesoj një permission groups, ai mund të përdorë të gjitha lejet individuale që janë pjesë e atij grupi. Ne nuk i miratojmë manualisht lejet individuale që i përkasin një përmission group që e keni pranuar tashmë. Këtu është një listë e përmissions groups që një aplikacion mund të përdori.[5]

Përdoruesit që dëshirojnë të kenë kontroll të plotë mbi lejet individuale në një app mund të rishikojnë lejet individuale për një aplikacioni në çdo kohë, ose mund të marrin parasysh fikjen e auto-updates. Me kalimin e kohës, sistemi opërativ Android ka prezantuar aftësitë e reja dhe karakteristika të reja. Permissions groups mund të ndryshojë gjatë kohës që aftësi të reja shtohen në Android. [5]

Tabela 2 Grupet e lejeve të sistemit operativ Android

GRUPET E LEJEVE	LEJET
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION

MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

KAPITULLI 3

RELATED WORK

3.1 Siguria në Sistemet Operative

Siguria në Sistemet Operative (SO) është një fushë shumë e gjerë e shkencës kompjuterike dhe ështe trajtuar nga disa studiuese. Enck ne punimin e tij [3] ka trajtuar në detaje punët më kryesore në këtë fushë. Shumë nga konceptet themelore të sigurisë së sistemeve operative u zhvilluan gjatë dizenjimit të Multics në vitet 1960-'70. Në 1974, Salter paraqiti nëntë fusha kërkimi për sigurinë e sistemeve operative duke përfshirë: system penetration, studime mbi ndërfaqën e përdoruesit, modele matematikordhe mekanizma mbrojtës. Një vëzhgim i literaturave të sotme të sistemeve operative tregon praninë e këtyre fushave në sistemet operative modern. Jaeger na paraqët një përshkrim të parimeve bazë të sigurisë në sistemet operative duke flluar nga Multics deri në platformat e sotme UNIX dhe Microsoft Windows. [3]

Në sistemet operativ tradicional aplikacionet ekzekutohen si *procese*. Çdo proces ka një *protection domain*, i përkufizuar si informacioni dhe burimet që një proces mund të aksesoj. Mbrojtja (protection) në sistemet opetative mund të paraqitet si një matricë e kontrollit të access-it (*access control matrix*). Matrica është e përbërë nga një grup subjektesh (p.sh. procese) dhe objektesh (p.sh. file) dhe me elemente matrice vepimet (actions) (p.sh. read, write) të cilët subjektet mund t'i përdorin mbi objektet. Matricat e kontrollit të access-it zakonisht nuk janë shumë të populluara ateherë politikat e mbrojtjes marin formën e listave të kontrollit të access-it (*access control list* (ACL)). Në politikat e mbrojtjes çdo objekti i jepet një ACL në të cilën janë përcaktuar subjektet dhe veprimet që këto subjekte mund të kryejnë. Një paraqitje tjetër e politikave të mbrojtjes është *capability list* (C-List). Në këtë model subjekteve u caktohet një C-List duke përcaktuar në të objektet dhe veprimet që subjekti mund të kryej mbi objektin. [3]

Politikat e kontrollit të access-it mund të jenë: *discretionary (të lirë për të vepruar)* ose *mandatory (të detyruar).* Politika discretionary menaxhohet nga subjektet. Zakonisht subjekte janë userat ose procese që po ekzekutojnë në autoritetin e userit. Si rrjedhojë politika e kontollit të access-it të lirë për të vepruar zakonisht përcaktohet si politikë e menaxhuar nga useri. Nëse politikat e tranzicionit nuk limitohen me kujdes mund te shkaktohen cënime të sigurisë. Përcaktimi nëse kjo politikë mund të arrij ose jo një gjendje jo të sigurtë nuk është e ditur, ky kusht njihet si problemi i sigurisë (safety problem). Politika e kontrollit të access-it të derytuar (*Mandatory access control policy* (MAC policy) ja kufizon politikat e menaxhimit entitetit administrativ, kështu që nuk është subjekt i problemeve me sigurinë. Pjesa më e madhe e literaturave për sigurinë e sistemeve operative fokusohen në politikën MAC sepse ajo mund te ofroj garanci të provuar sigurinë. [3]

Në bazë të politikës MAC është një reference monitor [3]. Një reference monitor kërkon:

1. Të ndërhyjë në të gjitha veprimet që mund të cënojnë sigurinë.
2. Pacënueshmërinë e mekanizmave mbrojtës.
3. Të verifikoj pacenueshmërinë dhe të gjitha ndërhyrjet.

Pavarësisht kërkesave të reference monitor-it vetem disa sisteme janë të verifikueshëm në mënyrë rigoroze për shkak të kufizimeve praktike.

3.2 Mbrojtja në nivel Kerneli

Në vitet 1970 – ’80 në fokus të sigurisë së sistemeve operative ishte verifikueshmëria. Që të jetë i verifikueshëm një sistem kompjuterik duhet të ishte i vogël. Ky vëzhgim çoj në krijimin e konceptit *security kernel,* i cili mund të përcaktohet si hardware-i dhe software-i i nevojshëm për të arritur një reference monitor. Shembuj të security kernel të hershëm janë Scomp [9] dhe GEMSOS [10]. Scomp ka të theksuar praninë e hardware-ve të përshtatur për të minimizuar softwaret e besueshëm. Stresat mbrojtëse për të izoluar software jo të sigurtë menaxhohen tërësisht nga hardware, gjithashtu një modul sigurie ndërmjeteson aksesin direkt të memorjes (direct memory access (DMA)) në busin I/O

[3]. Në të kundërt, GEMSOS u dizenjua që të opëroj mbi hardware x86. Edhe pse arkitektura x86 ofron shtresa mbrojtëse ajo nuk mund të ndërhyj në DMA. Pavarësisht këtij limiti, GEMSOS u vlerësua dhe verifikua me Trusted Computer System Evaluation Criteria's level A1 [10] [3].

Ndërsa kërkimet për security kernels po përparonin, Rushby [11] vërejti se qëllimet e sigurimit të verifikueshmërisë (në këtë rast, të multilevel security) ishin shpesh në kundërshtim me realitetet praktike, duke rezultuar në shumë procese të besuara jashtë kernelit. Ky vëzhgim çoj ne lindjen e konceptit *separation kernel* [12]. Një separation kernel është, në fakt, një security kernel, ai ka një synim shumë të veçantë ndaj sigurisë, të arritj ndarje të barasvlershme të hosteve në një sistem të shpërndarë. Për këtë arsye, verifikueshmëria e separation kernel thjeshton punën për të siguruar izolimin midis "regjimeve". Në këtë model, regjimet nuk duhet të jetë plotësisht i izoluar, por kanalet e komunikimit duhet të jenë të përcaktuara mirë, dhe me aftësinë për t'i "prerë" ata nëse është e nevojshme [3].

Koncepti i një separation kernel çoj në krijimin e microkernelave , të cilët hoqën të gjitha funksionalitetet e panevojshme nga kerneli, duke i vendosur ato në proceset e user-space. Mikrokerneli Mach [13] aplikon këtë përafrim për arkitekturën UNIX, duke siguruar ndarje sipas privilegjit të funksionaliteteve të kernelit. Megjithëse ky dizajn rrit IPC (interprocess communication) ai ka një përformancë të dobët. Mikrokerneli L4 [14, 15] tregoi se përformanca e microkernelit nuk ka nevojë të jetë dukshëm më e keqë se arkitekturat macrokernel që përdorin implementime me procesor specifikë për abstraksionet me procesor të pavarur [3].

Në përshkrimin e tij të modelit separation kernel, Rushby [11] vuri në dukje ngjashmërinë e separation kernel me virtual machine monitors (VMMs) [16, 17], p.sh., the IBM VM/370 [18]. VMM simulon "pjesë hardware" për të lejuar makina të shumta virtuale të ekzekutohen në të njëjtën kohë në të njëjtën makinë fizike. Kelem and

Reiertag [19] zyrtarisht modeluan VMM-të. Kohët e fundit, sistemet VMM tilla si VMware [20] dhe XEN [21] janë shfrytëzuar që të ofrojnë një përimetër sigurie. Terra [22] demonstron një arkitekturë ku aplikacionet desktop ekzekutohen në VM të vecanta. sHype [23] implementon mandatory access control për komunikim mes VM-ve [3].

3.3 Information flow

Historikisht, mandatory access control ka qënë sinonim me *information flow control* (IFC). Ngjashëm meaccess control matrix të Lampson [7], IFC modelon një grup subjektesh S dhenjëgrup tëobjekteveO. Veprimet read dhe write paraqiten si rrjedha (flows) (→). Për shëmbull, kurnjë subjekt s∈S lexon nga një objekt o∈O, atëherë s←o. Në anën tjetër, kur shkruan në o, atëherë s→o. Këto opëracione formojnë një grafi krrjedhë-informacion G= (V, E) ku V është bashkimi i subjekteve dhe objekteve (V =S∪O) dhe E janë rrjedhat (flows). Matrica e Lampson-s mund të konvertohet nënjëgrafik të rrjedhës së informacion duke shënuar çdo llojveprimi njërin nga rastet: read, write, read-write. [3]

Modelet information flow security etiketojnë çdo subjekt dhe objekt me nga një klasë sigurie. Denning [24] i organizoj klasat e sigurisë në një listë për të përcaktuar parimet e sigurisë. Lista specifikon rrjedhat(flows) e lejueshme mes klasave të sigurisë. Bell dhe LaPadula [25] përcaktuan modelin multi-level security (MLS) duke përdorur lista për konfidencialitet. Në MLS, kodohen politikat simple-security (no read up) dhe ★ -security (no write down) mbi klasat e sigurisë hierarkike (p.sh., top-secret, secret, confidential, dhe unclassified). Biba [26] propozoi integritetin e dyfishtë për kodimin e politikave simple-integrity (no read down) dhe ★-integrity (no write up) brenda listave. Ndërsa modelet e konfidencialitetit dhe integritetit te information flow mund të zbatohet në të njëjtën kohë, subjektet vetëm mund të lexojnë dhe shkruajnë brenda klasave të sigurisë së tyre. Për shëmbull, ndërsa MLS përdor klasa strikte të sigurisë, një model i integritetit mund të përdorin klasat e sigurisë të tilla si: trusted, system, application, user, and untrusted. [3]

Edhe pse modelet e MLS dhe Biba japin garanci matematikisht të vërtetueshme, zbatimet praktike mbështeten në sigurinë e proceseve të besuar. Një proces i besuar është

lejuar të veprojnë jashtë kufijve të listës të sigurisë dhe për këtë arsye përfaqëson një rrezik të mundshem. Modeli i integritetit i Clark-Wilson [27] kufizon se si mund të ekzistojnë proceset e besuar.Ai siguron verifikueshmërinë përmes certifikimit e të gjitha proceseve të besuar.Modeli CW-lite [28] relakson Clark-Wilson duke mos kërkuar certifikimin e të gjithë proceseve, por mbështetet në ndërfaqë të përcaktuara për të pranuar apo injoruar inputet nga proceset me integritet të ulët. Usable Mandatory Integrity Protection (UMIP) [29]gjithashtu relakson modelet e rrepta të integritetit dhe përcakton besimin në aspektin e llojeve të rrjedhave te informacionit (P.sh., IPC, network).UMIP merret me përjashtime të një modeli duke përdorur rregulla të përcaktuara nga konfigurimi i accessit të kontrollit të sistemit. Një model tjetër praktik , Practical Proactive Integrity (PPI) [30],përdor një kombinim të integritetit të etiketave dhe politikave në mënyrë fleksible për të garantuar integritetin e sistemit. [3]

Decentralized information flow control (DIFC) [31, 32, 33] është një mënyrë tjetër për të tejkaluar kufizimet e proceseve të besuar. DIFC është një application i “the decentralized label model” (DLM) [34]për sistemet operative.Në vend të përcaktimit të përjashtimeve në një MLS të përcaktuara në nivel qëndror apo në një liste Biba, DIFC ndan besim në shumë klasa sigurie jo të krahasueshme.Këto klasa, të quajtura *tags*, krijohen dhe menaxhohen nga aplikacionet.Etiketat e subjekteve dhe objekteve përbëhen nga grupe të tageve, dhe lista përcaktohet nga një rregull praktik i grupit të këtyre tageve. Një subjekt thuhet që "zoteron" një tag nëse mund të shtoj dhe heq nga etiketa e tij. Për më tepër, subjektet mund të delegojnë shtimin dhe heqjen e aftësive të tagut ndaj subjekteve të tjera për të menaxhuar besimin. Themelore për këtë përafrim është vëzhgimi se një sistem ka shumë lloje të informacioneve jo të krahasueshme, dhe shpesh, vetëm zhvilluesi i aplikacionit ka kontekstin e mjaftueshëm për të administruar politikën e mbrojtjes. [3]

3.4 Modelet e tjera MAC

Jo të gjitha sistemet e mbrojtjes MAC përqëndrohet në information flow [3]. Për shëmbull, Domain Type Enforcement (DTE) [35] parashikon ndarjen e privilegjeve për

proceset në pronësi të root në sistemet UNIX. DTE është e ngjashme me Type Enforcement (TE) [36]; megjithatë, në DTE, vetëm objektet janë emërtuar me privilegje. Subjekteve u janë të caktuar një domain, i cili përmban 1) "pikë hyrje (entry point) " program (p.sh., një binary path), 2) të drejtat e aksesit (access rights) (read, write, execute) mbi objektet , dhe 3) të drejtat e aksesit (p.sh. , sinjale) në domain të tjere. SELinux [37](Një implementim i Flask [38] për Linux) është i bazuar edhe në privilegje. Ashtu si DTE, SELinux vetëm përdor privilegjet për objektet; megjithatë, subjekteve u janë të caktuar një rol. Rolet e SELinux janë të ngjashme, por të dallueshme nga, rolet në RBAC [39]. Politika SELinux specifikon se si rolet mund veprojë në tipet, si dhe rregullat e tranzicionit mes roleve. Politika e kontrollit të aksesit në SELinux është shumë komplekse dhe shpesh e prish sistemin prandaj caktohet një politikë në bazë të objektivit. Ngjashëm me DTE, politika SELinux siguron ndarje sipas privilegjit për proceset root. Në përgjigje të kompleksitetit të politikës SELinux, AppArmor [40] (i njohur më parë si SubDomain [41]) përkufizon politikën MAC duke përdorur path names për të arritur semantikë të ngjashme me SELinux [3].

Aftësitë (*Capabilities*) kanë qënë gjithashtu të përdorura për mbrojtjen e MAC. Një *Capabilities* është e ngjashme me një celes fizik që i jepet një subjekti për të fituar akses në disa nderfaqë. Në një model të *Capabilities*, politika e mbrojtjes menaxhohet përmes delegacionit të *Capabilities* nga një proces në një tjetër. *Capabilities* natyrisht përfshijnë*the principle of least privilege* [42], si politikë aksesi në runtime. Kjo karakteristikë është e njohur më mirë si një zgjidhje për *theconfused deputy problem* [43]. Ky problem ndodh kur një proces jo i besueshëm bind sistemin për të kryen një veprim në emër të tij. Në një model të *Capabilities*, proceset e sistemit duhet vetëm të kryejnë veprime duke përdorur grupin e *Capabilities* të caktuara për proceset jo të besueshëm. Ndërsa delegimi i *Capabilities* shfaqet si diskret, ai ka qenë përdorur për sigurine e MAC në sistemet operative. Sistemet operative të hershem më të njohur janë bazuar mbi *Capabilities* HYDRA [44]. Boebert [45] dhe Karger [46] theksojne se sistemet operative tradicionale nuk sigurojne cilesite e . -Security (no write down) të kërkuar nga sistemet MAC. Dështimi lind nga aftësia që ka një proces i lartë për të lexuar të dhëna për një

proces të ulët dhe pastaj të përdorin ato të dhëna për të shkruar në një ndërfaqë të ulët. Sistemet operative SCAP [47] dhe EROS [48] e kane kapërcyer këtë cilësi duke përdorur aftësitë *read-only* dhe aftësitë e *weak*, respektivisht. [3]

3.5 Mbrojtja e informacionit të përdoruesit

Trojanë dhe malware të ngjashëm hyjnë në sistem si file të shkarkuara. Shumë teknika të bazuara në sandboxing janë propozuar për të parandaluar malware që dëmtojne përdoruesit.Janus [49] përcakton profile të sigurta për aplikacionet ndihmëse (p.sh.,document viewers dhe media players) për të kufizuar ndikimin e shfrytëzimit të fileve të dëmshme të shkarkuara me Email dhe Web browsers.Jaeger et al. [50] garandoj akses të siguruar për të shkarkuar përmbajtje të ekzekutueshme (p.sh., scripts) të bazuar në politikat e specifikuara. MAPbox [51] përshkruan politika të paracaktuara të sandbox bazuar në etiketat e aplikacioneve. Lai dhe Gray [52] përkufizuan aplikacionet të bazuara në argumente të programit dhe në nivelin besueshmërisë. Të drejtat për të aksesuar file i paraqiten përdoruesit. [3]

Modele ku përdoruesi në mënyrë specifike përcakton një veprim si jo të besueshëm janë propozuar. TRON [53] përdor *Capabilities*për file dhe tree directory. *Capabilities*mund të caktohen në krijimin e procesit (fork) për të kufizuar aplikacionet jo të besueshme. Plash[54] ofron funksionalitet të ngjashëm për përdoruesit brenda një ndërfaqë command line shell. [3]

Dëmtimi nga malware dhe aplikacione të dëmshme gjithashtu mund të ulet nga ekzekutimi i aplikacioneve si identiteteve ndara.Për shëmbull, Polaris [55] përdor profilet e paracaktuara që automatikisht të ekzekutoj aplikacione specifike si një llogari e veçantë përdoruesi.Koncepti i identiteteve në nivel programi për aksesimin e fileve fillimisht u shfaq në PACLs [56], ku listat e kontrollit të aksesit të programit janë të specifikuara në file.Disa sisteme për mënyrën e trajtimit të programeve janë propozuar ,duke përfshirë PinUP [57, 58], Sub-Operating Systems [59], Sub-Identities [60], and FileMonster [61]. Vini re se duke ekzekutuar të gjitha aplikacionet si përdorues të veçant, kontrolli i aksesit të skedareve në Android gjithashtu bie në këtë kategori [3].

3.6 Mbrojtja e Sistemit Operativ të Smartphone

Smartphonet shpesh konsiderohen si pajisje me burime të kufizuara që ekzekutojnë sistemeve operative të plotë [3]. Si e tillë, kërkuesit kanë ri-aplikuar teknikat tradicionale me modifikime për të përmbushur kërkesat e mjedisit të ri.Integrity measurement and remote attestation [62] është një teknikëe tillë. Në teknologjinë smartphone ka shumë njërëz të interesuar. The Trusted Computing Group's (TCG) mobile specification [63]supozonkatërpalë të interesuara: device manufacturer, network opërator, third-party service provider, dhe end user.Ai përdorMobile Remote - Owner Trusted Module (MRTM) dhe Mobile Local-Owner Trusted Module (MLTM)për të kryer detyrate kryeratradicionalishtngaTrusted Platform Module (TPM).NjëarkitekturëtelefoniTCGka shumë domain të izoluar , një për secilin aktor, ku secili domain kaMRTMe vetlogjik, me përjashtim tëdomainitend-user, i cili përdor njëMLTM. Zhang et al. [64] realizoj specifikimet e telefonit celular të besuarTCG duke përdorurSELinuxpër të izoluardomainetopëracionale.Kohët e fundit, Zhanget al. [65] dizenjoj një framework për plarformën LiMo. Në mënyrë të ngjashme, Naumanet al. [66] zbatojë një testimpër Android, duke përfshirëmatjen eaplikacioneve.Matje e integritetitështëpërdorur gjithashtupër tëarritur qëllimetklasike.Për shëmbull, Muthukumaranet al. [67] përmirësoj platformën Linux-based Open moko me matje të integritetit dhe SE Linux për të zbatuarqëllimet e sigurisë të CW-lite nëinstalimin e softwareve.SHABTAIet al. [68] implement SELinux përAndroid për të forcuar sistemin Linux-based nën middleware. [3]

Virtualizimi është zbatuar në telefona për të siguruar ndarjen mes grupeve tëaplikacioneve. VMware mobile virtualization [69] i lejonnjë administratorit të instaloj hosted VM që përmbajne aplikacione biznesi .Megjithatë,ky modellejondobësitë e intalimeve përsonale të userit të ndërhyjne në aplikacionet e biznesit.Në të kundërt, OKLabs[70] përdorL4si njësupërvizorpër të ekzekutuarnë të njëjtën kohëAndroid, Symbian, dhe Windows.Së fundi, Leeet al. [71] zbatoj politikat MAC për komunikiminndërmjetVM-veqë ekzekutohen nënjë telefon.Arkitektura kangjashmëri me një konfiguraciontë ARM TrustZone for embedded Linux [72] [3].

Politika të sigurisë të Higher-layer frameworks janë marë në konsiderate gjithashtu. Mullineret al. [73] për të parandaluar sulmet etiketon proceset bazuar nën dërfaqëte sigurisëtë aksesit te tyre. Politikatparandalojnë sulmet cross-service duke parandaluarnjë procestë përdori kombinimete pasigurt atëndërfaqëve. Ionet al. [74] zgjerojJ2ME security frameworkpër të përfshirëpolitikaqë kufizojnëpërdorimine shërbimevegjatënjë përiudhe të caktuarkohore(p.sh., SMS, data).Sxc[75] i lejon përdoruesit(ose ekspërtët) të përcaktojnë "kontrata të sigurisë(security contracts)" për aplikacionet .NET që ekzekutohen në Windows Mobile phones.Kjo qasjelejon përdoruesittë instalojnë aplikime jo të besueshme,por të ruajnëgarancitë caktuararuntime. Në njëpërpjekjetë ngjashmenëplatformën Android, Apex [76] ndryshon modelin “all or nothing” , duke i lejuar përdoruesit tëzgjedhin në mënyrë individuale lojin e lejeve(përmissions) që do të kenë aplikacionet, si dhe kushtet kur lejetmund të përdoren. CRePE [77] gjithashtuzbatonpolitikakontekstualepërdoruesifine-grained (p.sh., location, time). Në të kundërt, Saint[78] rrit numrinepolitikave developër-defined tësigurisë përplatformën Android. Politikat Saint mund të kufizojn ëtë dyja instalimet e aplikacioneve dhe ndërveprimet runtime mes aplikacioneve. Porscha [79] zbaton politikënpër content owners, duke kufizuarsi SMS dhe Mesazhet Email të përdoren pasi kanë ardhur në telefon. [3]

Karlson et al. [80] intervistoi përdorues smartphonesh për të kuptuar më mirë gatishmërinë e tyre për të ndarë te dhënat me përdorues të tjerë. Zbulimet tregojnë se smartphonet duhet të sigurojnë mënyra më pak të privilegjuara për të lejuar përdoruesit "guest users" për të përdorur telefonat pa kompromentuar privatesinë e pronarit të smartphonit.Gjetjet të ngjashme janë raportuar nga Liu et al. [81], prototipi xShare i tyre për Windows Mobile demonstron se si duhet të veprohet në një rast të tillë. Diffuser [82] siguron mbështetje të ngjashme për Android. Ngjashëm me punën time, kërkuesit kanë studiuar kufizimet e application-level policy frameworks. Shin et al. [83] zyrtarisht modeloi Android’s security policy language për ndërveprimet mes aplikacioneve.Duke përdorur këtë model, ata identifikojnë një sulmi përshkallëzimit të privilegjit ku një aplikacion pa privelegj mund të kryejnë opëracione të ndjeshme ndaj sigurisë ose duke

shfrytëzuar dobësitë në aplikacionet e tjera ose marrëveshje të fshehtë [84]. Një zgjidhje për këtë problem kërkon ndjekjen kalimtare të përmission use.Chaudhuri [85] përcakton një afrim të bazuar nga gjuha e programimit për vlerësimin e komunikimit mes aplikacioneve.Ky model formal është përdorur për të ndërtuar ScanDroid framework [86] për certifikimin automatikisht të aplikacioneve bazuar në source kodin e tyre.Megjithatë, ky model kërkon një leje paraprake në bazë tëAndroid përmissions ekzistuese.Për fat të keq, shumica e Android përmissions janë jo të krahasueshme, e cila pengon zbatimin praktik të këtij modeli. [3]

3.7 Malware detection

Zbulimin dhe parandalimi i Trojanëve dhe malwareve të tjera ka qënë prej kohësh një temë e kërkimeve në lidhje me sigurinë. Në mjedise desktop dhe server, antiviruset software zakonisht scanojnë filet për të zbuluar malware në bazë të tiparëve të paracaktuara. Megjithatë, skanimi i tillë është tepër intensiv për smartphonet, në mënyrë specifike në lidhje me konsumin e energjisë. Prandaj, strategji alternative janë të nevojshme [3].

Venugopal dhe Hu [87] përshkruan një skaner efikas i optimizuar për telefonat celular. Si një alternativë, Bose et al. [88] propozoj përdorimin e analizës së sjelljes në bazë të opëracioneve security-sensitive. Andromaly [89] implementoi anomaly detection për Android bazuar në objekte runtime të tilla si ngarkesa e CPU, input events, dhe konsumi i energjisë. Nash et al. [90] konsideroj një sistem të zbulimit te ndërhyrjeve për sulmet që shkaktojne shkarkimin e baterisë; megjithatë, puna i tij vetëm diskuton algoritma të mundshme. Më në fund, Kim et al. [91] përdori një metodë të bazuar në zbulimin e anomalive të konsumit të energjisë për të zbuluar malware. Modeli i propozuar mbështet në modelin stand alone ku të gjitha analizat ndodhin në telefon, si dhe përdorimi i një serveri në distancë për përpunimin e të dhënave. [3]

Disa sistemet e propozuara dërgojnë shkrimet e ngjarjeve të sistemit në një server qëndror për analizë. Megjithatë, siç vuri në dukje Miettinen et al. [92], mekanizmi i logging duhet të jetë i kujdesshëm për të mbrojtur privatësinë e përdoruesit, duke siguruar që informacionet private të rëndësishëm kurrë të mos e lëne pajisjen. SmartSiren [1]

dërgon shkrimet e aktiviteteve te pajisjes të tilla si përdorimi i Bluetooth dhe SMS në një server qëndror. Serveri qëndror pastaj kryen analizat e sjelljes gjithë pajisjeve, e cila është e dobishme për zbulimin e Bluetooth worms. Në mënyrë të ngjashme, Schmidt et al. [93] ruan sasinë e RAM-it te lirë, aktivitetit të përdoruesit, process count, përdorimit të CPU, dhe numri i mesazheve SMS të dërguara për analiza nga një server qëndror.

Analiza Malware gjithashtu mund të kryhet duke ruajtur konsistencën ndërmjet një smartphone dhe një imazhi të përsëritur në një network server [94]. Paranoid Android [95] përdor këtë përafrim, por ruan energji duke përdorur " loose synchronization " për të dërguar të dhëna vetëm kur përdoruesi është duke përdorur pajisjen. Oberheide et al. [96] siguroj një model alternativ për skanim antivirusi në server side. Këtu, autorët krijojnë një klient celular për arkitekturën CloudAV [97] që ngarkon file në një server për skanim nëse një skedar me njëjtë kriptografi hash nuk është skanuar. [3]

3.8 Gjurmimi i informacionit

Kontroli i information flow i sistemit operativ e ndjek informacionin deri në detaje. DEFCon [98] përdor një logjikë të ngjashme me sistemet operative DIFC, por fokusohet në ngjarjet dhe modifikon Java Runtime me izolimi të lehtë. Lidhur me këto përafrime, PRECIP [99] etiketon të dyja proceset dhe objektet shared kernel të tilla si clipboard dhe display buffer. Megjithatë, këto modele tëprocess-level information flow janë të dobët dhe nuk mund të gjejnë informata të ndjeshme brenda aplikacioneve jo të besueshme.
Information flow security [100] të bazuar në gjuhë programimi, shtojnëgjuhën e programimit ekzistues duke etiketuar variablat me atributet e sigurisë.Kompilatorët përdorni etiketat e sigurisë për të gjeneruar prova të sigurisë, p.sh., Jif [101, 34] dhe SLam[102]. Laminar[103] siguron garanci DIFC të bazuar rajone të sigurisë të caktuara nga programuesi. Megjithatë, këto gjuhë kërkojnë zhvillim të kujdesshëm dhe shpesh janë të papajtueshme me legacy software designs [104] [3].

Analiza dynamic taint (e njohur edhe si " taint tracking ") jep informacion për legacy programs. Në taint tracking, variablat apo regjistrat janë shënuar në një taint source ku semantikat e sigurisë së informacionit janë të njohura. Këto taint markings futem ne mënyrë dinamike gjatë ekzekutimit dhe përfundimisht vërehen si një taint sink, ku

veprimet e duhura thërriten në bazë të shënimit të tyre. Taint tracking tradicionalisht përdor semantika të përkufizuara data flow, të njohura dhe si explicit flows. Kjo teknikë mbështetet në një përkufizim induktiv për çdo instruksion që ndikon rrjedhën e informacionit. Për shëmbull, instuksioni c = a + b cakton rezultatin e shumës së a dhe b tek c. Megjithatë, semantika data flow nuk i kap të gjitha information flows. Flukset e nënkuptuara, i njohur gjithashtu si control flows, rezultojnë nga instruksione branch dhe loop. Për shëmbull, një variabel y mund të vendosen në 1 nëse një variabël i monitoruar x është i njëjtë me një vlerë të caktuar. Këtu, ka një rrjedhje e informacionit nga x në y pa një instruksion të qartë. Sistemet dynamic taint tracking japin llogari për disa lloje të flukseve eksplicite. Për shëmbull, indekset në një vektor futen shpesh që të japin llogari për translation tables (p.sh., konvertimi ASCII ne UNICODE). Taint scopes gjithashtu janë përdorur dikur kur adresat e instruksioneve start dhe end janë të njohur për branche dhe loop. Këtu, shenjat mbi variablat e kushtëzuara është prezent për të gjitha variablat e caktuara brenda branch-it ose loop-it. Megjithatë, ky është një përafrim dhe mund të çoj në përfundime të rëndësishme të gabuara. Së fundi, për të kapur për të gjitha implicit flows, janë të nevojshme disa analiza statike [105] [3].

Dynamic taint tracking ka qënë përdorur për të rritur integritetin e sistemit (p.sh., të mbrojkundër sulmeve software [106, 107, 108]) dhe konfidencialitetit (p.sh., të zbuloj ekspozimin e të dhënave private [109, 110, 111]), si dhe të identifikoj Internet worms [112]. Kur përmiresojm integritetin e sistemit, network interface është përdorur si taint source, duke shënuar të gjitha informatat përbrenda si jo të besueshme. Nëse informacioni jo i besueshem nuk është i pastruar para përdorimit në një pikë kontrolli (p.sh., stack return address). Kur përdoret për të monitoruar privatesinë, network interface është taint sink. Këtu, trafiku nga jashtë inspektohet për taint markings caktuar në burimet e ndjeshme të privatësisë. Burimet e ndjeshme të privatësisë nuk janë gjithmonë përcaktuar mirë. Për shëmbull, është e veshtire për të programuar ndarjen mes karaktere të futura për një fjalëkalim ose një kartë krediti nga përmbajtjet e një mesazhi të futur në ndërfaqën e tastierës. Megjithatë, smartphonet kanë të përcaktuara mirë taint sources me semantikë të qart për të gjithë informacionet te lexuara nga nderfaqja [3].

Dynamic tracking approache shtrihen nga whole-system analysis duke përdorur hardware[113, 114, 115] dhe emulation environments [116, 109] për për-process tracking duke përdorur përkthim dinamik binar (DBT) [117, 107, 108, 111]. Përformanca dhe memory overhead të lidhur me dynamic tracking ka rezultuar në një sërë optimizimesh,duke përfshirë context switches të optimizuar [107], on-demand tracking [118] bazuar në hypërvisor introspection, dhe funksione përmbledhjesh për kode me information flow propërties të njohura [111]. Nëse source kodi është në dispozicion, përmiresime të dukshme të përformancës mund të arrihen duke programuar programet e certifikuara me dynamic tracking functionality [119, 120]. Orkestrim Automatik është gjithashtu kryer mbi x86 binaries [121], duke siguruar një kompromis në mes përkthimit të source kodit dhe DBT. Designi TaintDroid është frymëzuar nga këto vepra të mëparshme, por adreson sfida të ndryshme për telefonat mobile. TaintDroid është sistemi i parë dynamic taint analysis për një telefon celular që ka arritur analizë praktike të sistemit përmes integrimit të tracking multiple data object granularities [3].

Së fundi, dynamic taint analysis ka qënë e aplikuar në makina virtuale dhe interpretuesit. Haldar et al. [122] instrumentoi klasën Java String me taint tracking për të parandaluar sulmet SQL injëksion. WASP [123] ka motive të ngjashme; megjithatë, ai përdor positive tainting e karaktereve individuale për të siguruar që SQL query përmban vetëm nënstringje të integritetit të lartë. Chandra dhe Franz [124] propozojë fine-grained information flow tracking brenda JVM dhe instrument Java byte-code për të ndihmuar analizat control flow. Në mënyrë të ngjashme, Nair et al. [125] krijoi Kaffe JVM. Vogt et al. [126] krijoi një përkthyes Javascript për të parandaluar sulmet cross-site scripting. Xu et al. [119] instrumentoi përkthyes të PHP source kode me dynamic information tracking për të parandaluar sulmet e SQL injëksion. Së fundi, the Resin [127] environment for PHP and Python përdor data flow trackin për të parandaluar një shumëllojshmëri sulmesh të aplikacioneve Web. Kur të dhënat lëne mjedisin interpretues, Resin zbaton filtra për file dhe bazave të të dhënave SQL për të serializuar dhe deserializuar objekte dhe politika në nivel byte. Kodi i interpretuar i TaintDroid mbart ngjashmëri me disa nga këto

vepra. Megjithatë, TaintDroid zbaton system-wide information flow tracking, duke lidhur interpreter taint tracking me një gamë të mekanizmave system sharing të sistemeve operative [3]

3.9 Analiza e sigurisë dhe privatësisë

Shumë paisje dhe teknika janë projektuar për të identifikuar shqëtësimet e sigurisë në software. Ky seksion përshkruan këto teknika në "real code", si qëllim të rrisim sigurinë e sistemit. Fillojmë analizen duke pasur në plan te parë analizën e cënueshmërisë. [3]

3.9.1 Analiza e vulnerabilitetit

[3] Software të shkruar në C janë veçanërisht të ndjeshëm ndaj gabimeve të programimit që rezultojnë në problem. Ashcraft dhe Engler [128] përdorën extensions të kompilatorëve për të identifikuar gabimet në range checks. Modifikimete kompilatorëve janë përdorurpër të identifikuar mbi 100 dobësinë Linux dhe kernellin OpenBSD. Problemet më të përgjithshme janë identifikua rduke përdorur MOPS[129], një tool për të kontrolluar programet e cila përcakton dobësitë si finite state automata (FSA).Chenet al. [129] përdoriMOPS për të analizuar tetë aplikacione Linux të përhapura, të përbërë nga më shumë se njëmilion rreshta kod.Në disa aplikacione, ai identifikoj dështimet që të heqin dorë nga privilegjet root.Shvarcet al. [130] gjithashtu përdorinMOP, duke kontrolluar të gjitha applikacionet nënjë Linux distribution (60.000.000 rreshta kod), duke zbuluar108buge.Në mënyrë të ngjashme ,BalldheRajamani[131] përdorenSLAM [132] për të zbuluargabimet nëdriverat e pajisjeve të Windows XP [3].

Aplikacionet Java janë në thelb më të sigurta se aplikacionet në C dhe shmangin dobësitë e thjeshta si buffer overflows . Ware dhe Fox [133] krahasuan tetë open source and commercially available Java source code analysis tools .Ata arritën në përfundimin se asnjë tool nuk i zbulon të gjitha dobësitë.Hovemeyer dhe Pugh [134] studiuan gjashtë aplikacione Java dhe libraritë më popullore duke përdorur FindBugs të zgjeruar me kontrolle shtesë.Ndërsa analiza përfshinte non-security bugs, rezultatet motivuan një nevojë të fortë për analizë të automatizuar nga të gjithë zhvilluesit.Livshits dhe Lam [135] u fokusuan në Web aplikacione të bazuara në Java.Në Web server environment,

inputet kontrollohen lehtësisht nga një kundërshtar, dhe mos kontrollimi mund të çojë në SQL injëksion, cross-site scripting, HTTP response splitting, path traversal , dhe command injëction.Duke përdorur static taint analysis, nëntë server side aplikacione Java janë studiuar, duke gjetur 41 shkelje potenciale të sigurisë, nga të cilat 29 ishin gabime të sigurisë. Felmetsger et al. [136] gjithashtu studioj web aplikacione Java-based; megjithatë, ai përparoj në analizën e cënueshmërisë, duke siguruar zbulimin automatik të gabimeve specifike logjike të aplikacioneve.Tool i tij, Wailer, përdor tool ekzistuese analizuese për të identifikuar ndërhyrje të mundshme.Raste të gabuara pastaj hiqën duke përdorur model checking. Asnjë annotation manual nuk është i nevojshëm në të dyja fazat. Wailer është vleresuar duke studiuar katër Web aplikacione popullor, duke identifikuar 30 dobësi [3].

Web aplikacionet shpesh shkruhen dhe në PHP.Jovanovicet al. [137] dizenjoiPixy për të identifikuarSQL injëction dhe dobësitëcross-site scriptingduke përdorurtaint analysispër PHP.Ai përdoriPixy për të studiuar shtatë aplikacione popullore PHP duke identifikuar mbi 200 dobësi. Balzarottiet al. [138] dizenjoi Sanerpër të zbuluar dobësitë input validation në aplikacione PHP.Saner përdor si analizën statike dhe dinamike.Së pari, analiza statike modelon modifikime të inputeve në të gjitha pathet nga burimi në një destinacion. Pastaj, analiza dinamike largon lidhjet gabim duke përcaktuar se cilat kode paths janë përdorur nga aplikacioni. Saner është përdorurpër të studiuar pesë aplikacione popullore PHP dhe janë identifikuar 13 dobësi të pazbuluara më pare [3].

3.9.2 Analiza e privatësisë

Klasa spyware e malware kërkon të nxjerrë informacion privat të userit. Browser Helpër Objektet (BHOs) dhe Web browser toolbars janë burime të zakonshme të spywareve. Kirda et al. [139] konsideroj tiparët e sjelljes së BHOs dhe toolbareve. Ai vërejti se për të arritur qëllimin e tij, BHO dhe toolbar spyware duhet edhe të monitoroj sjelljen e përdoruesit dhe të kontrolloj Windows API calls që potencialisht nxjerrin informacion. Duke përdorur static analysis të API calls, me analizë dinamike që të përsosin kodin e ekzekutueshem, u studiuan 33 probleme sigurie dhe 18 malware BHO dhe toolbars. Duke

përdorur analizën e kombinuar, vetëm dy komponente të studiuar u identifikuan si spyware. Egele et al. [110] gjente informacionin që rrjedh nga spyware browser-based duke përdorur në mënyrë eksplicite analizën dynamic taint. Përafrimi i tij modifikon QËMU për të kryer vetëm taint tracking të BHOs. Studim heton 21 komponentë të njohur spyware dhe 14 malware BHO. Përveç komponentëve spyware, studimi gjeti dy malware BHO që nxjerrin informacione private. [3]

Yin et al. [109] e konsideroj privacy-breaching malware në përgjithësi me Panorama, një tool i projektuar për të whole-system , fine-grained taint tracking. Panorama krijon " taint graphs " që janë analizuar nga politikat e malware detection. Panorama është përdorur për të studiuar 42 lloje malwaresh të vërtetë dhe 56 aplikacione problematike. Nga këto lloje, vetëm tre janë identifikuar në mënyrë të gabuar si malware për shkak të prezencës së sjelljeve të keqpërdorimit të informacionit. Autorët vërejnë se këto rezultate të gabuara positive rezultojë nga një kufizim i përdorimit të informacionit : domethënë, paaftësia për të kapur qëllimin [3].

Jung et al. [140] përdor testimin diferencial blackbox fuzz në hartimin e Privacy Oracle . Kjo teknikë kombinon parametrat kontrolluar në input (p.sh., një grup i përdoruesve) me testimin tradicionale fuzz . Korrelacionet midis inputeve të kontrolluara dhe të dhënave të qëndrueshme të trafikut të rrjetit përdoren si tregues të rrjedhjeve të privatësisë. Privacy Oracle është përdorur për të studiuar 20 aplikacionet me te vleresuara nga download.com dhe 6 klientët më të njohur të IM, duke identifikuar shumë nga problemet që ishin zbuluar më parë. Yumerefendi et al. [180] propozojë një përafrim të ngjashëm për TightLip; megjithatë, nuk ka studim për prova të kryera [3].

Së fundi, Egele et al. përdori PiOS për të kryer analiza statike në aplikacione iOS. PiOS rindërton rrjedhen e informacionit në Objective-C binaries. PiOS përdoret për të studiuar përdorimin e ID, location, address book, phone number, browser history, dhe fotove në mbi 1,400 iPhone aplikacione nga të dy tregjet e aplikacioneve Apple App Store dhe Cydia. Studimi gjeti se shumicës së aplikacioneve u mungon devide ID. Studimi gjeti gjithashtu se mbi gjysma e aplikacioneve përfshijnë librari reklamimi dhe analizimi, të

cilat shkaktojnë vështirësi kur vendosen flukset e informacionit. Për të mënjanuar këtë, u implementuan librari të flukseve të njohur në reklamim dhe analizë [3].

KAPITULLI 4

Malwaret

4.1 Përshkrim i përgjithshëm

[6] Në fund të vitit 2010, u zbulua një malware i ri për platformën Android i quajtur 'Geinimi' dhe menjeherë u klasifikua si " Android malware më i sofistikuar deri më atehëre". Ky lajm zuri shumë shpejt faqet e para të gazetave e revistave duke rritur kështu shqetësimin se cfarë maleware njihen deri më tani, cili është rreziku që i kanoset një përdoruesi të Android, dhe cfarë pritet të ndodhi në të ardhmen. Ketu do te trajtohet i gjithe evolucioni qe kanë kaluar malware duke filluar që nga SMS Trojan i parë i zbuluar në treg në 2010 dhe duke vazhduar me malware më të sofistikuara që u zbuluan në marketin zyrtar te Android ne vitin 2011 si DroidDream, DroidKungFu, dhe Plankton. Gjithashtu trajtohen dhe metodologjitë dhe mjetet e nevojshme për analizimin e malware. Në këtë pjesë flitet dhe për pritshmeritë në të ardhmen e malware [6].

Me 3 Prill ,1973 Martin Cooper, Menaxher i Përgjithshem i Motorola Communication Systems Division në atë kohë, bëri telefonatën e parë publike me një telefon celular. [142] Tanime rreth 38 vite pas atij momenti , telefonët celulare jane pjesë e një industrie të madhe të cilat prodhojnë paisje jo vetëm tv afta per të berë ose marrë thirrje telefonike , por edhe per të dërguar mesazhe, të navigojnë në internet, të kapin fotografi, të regjistrojnë video dhe të dërgojnë, marrin , dhe ruajnë informacione konfidenciale si kontaktet, email, foto, dhe video. Informacioni konfidencial është dicka shumë tërheqese për kriminelet e internetit, por shumica e këtyre paisjeve janë shumë të limituara në konfigurimet e tyre hardware dhe software, dhe shumica prej tyre nuk janë të lidhura me internetin vazhdimisht, pra perhapja e kercenimeve është e veshtire gjithashtu dhe fitimet potenciale të ketyre 'kriminelëve' janë të ulta [6]

Megjithatë , ekziston një klasë paisjesh celulare e cila përputhet me karakteristikat e përmendura më lart, paisjet Smartphone. Smartphonet janë paisje celulare high-end të

cilat ofrojnë fuqi përpunimi dhe lidhje në rrjet më te avancuara se një paisje celulare e thjeshtë. Për këtë shkak, ato janë shndërruar në një ndër mjetet kryesore me anë të të cilave njerzit aksesojnë rrjetet sociale, gjithashtu ato jane prane shndërrimit në një "portofol inteligjent" me anë të një marreveshjeje midis operatorëve telefonik, pra po shkojmë drejt një të ardhme ku keto paisje do të sillen si "celësa makine, para, bileta, dhe karta udhetimi" [142]. Kohët e fundit dicka e re u shfaq në këtë industri dhe po konsumon marketet dhe të netbooks dhe laptopëve, Tabletat. Te dyja këto lloje paisjesh (Smartphone dhe Tabletat) punojnë me Sisteme Operative të lehta të cilat janë optimizuara të punojnë me konsumim të ulët energjie. Një nga këto sisteme me popullore ne industri është Android [143]. Android është nje sistem operativ me bazë të sistemit Linux [144] dhe u hodh ne treg ne vitin 2007 [145] si nje projekt i Open Handset Alliance me Google si sponsor kryesor. Ne ate kohe kur doli lajmerimi, Mikko Hypponen i F-Secure ishte shumë skeptik mbi ketë platformë të re open source, duke ngritur pyetjet si "A do të sjelli një standart i hapur problem me malware?". Duheshin me shumë se tre vite, por mesa duket pasi Android terhoqi interesin e personave me qëllime te keqija, 2011 u shndërrua dhe në vitin e malware për Android [6].

4.2 Kategorizimi i malwareve në bazë të qëllimit

Në varësi te sjelljes dhe qëllimit të krijimit të tyre, malware ndahen në këto kategori [3]:

- *Proof-of-concept*: Një malware i tillë zakonisht shfaqet kur vektore të ri infektimi eksplorohen nga krijuesit e malware dhe zakonisht keto malware kanë pasoja të pa menduara më parë. Për shembull, Cabir demostronte shpërndarje në bazë të Bluetooth dhe pa dëshirë shkarkonte baterite e paisjes. Me ardhjen ne treg te platformave te reja si Android dhe Iphone, do të vazhdohet te shikohen malware proof-of-concept si RootStrap.

- *Destructive*: Janë ato malware të cilat kanë qëllime destruktive dhe më të perhapura. Edhe pse besohet që malware-t te cillat sjellin fitime monetare do të kalojnë maware-t me qëllime destruktive, ky trend pritet të vazhdoje. Malware e të ardhmes mund të infektojnë më shumë se integritetin e paisjes. Sisteme operative të përdorura sot dhe aplikacione të ndryshme varen shume tek cloud computing per memorje dhe backup. Neqofte se malware, për shembull fshin të

dhëna nga kontaktet e telefonit, të dhënat e humbura do të përhapen tek sinkronizimi i ardhshëm i cloud dhe rrjedhimisht do të ndikojë tek të gjithë përdoruesit e tjerë.

- *Premediatated spyware*: Këto malware krijonë mundesinë e përgjimit. Këto malware zakonisht përdoren për spiunime indrustriale. Fillimisht këto duhet te shkarkoheshin dhe instaloheshin nga një person, me zhvillimin e tyre ato mund të shkarkohen dhe instalohen nga një tjetër malware i cili vepron në paisje.

- *Information scavengers*: Këto malware zakonisht përkojne për të dhëna vulnerabel si librin e kontakteve dhe te dhenat e llogarive.

- *Botnet*: Të quajtura mobots janë afershisht të njëjtë me botnet-et në PC (si DoS dhe spam). Paisjet telefonike pritet te jenë subjekte te Dos dhe te spam-eve SMS.

4.3 Historiku i Malware-ve

Koded e këqija për paisjet celulare shpeshherë konsiderohen si mit për shkak të limitimeve në hardware dhe software të këtyre paisjeve. Megjithatë, historia ka treguar që paisjet celulare janë të gjithashtu të ekspozuara ndaj këtyre rreziqeve. Një nga rreziqet më të përhapura ishte dhe Cabir, një malware për sistemet Symbian i cili përdorte Bluetooth si kanal kryesor shpërndarjeje. Kodi i ketij malware u lëshua për herë të parë ne Internet nga grupi 29A ne vitin 2004 duke cuar kështu drejt varianteve të tjera të ketij malware dhe duke nisur kështu periudhën e pasigurisë së paisjeve celulare. Cabir u shfaq kur Bluetooth ishte teknologjia principale e përdorur për transmetimin e informacionit midis paisjeve të ndryshme [6].

Në vitin 2007, Apple prezantoi gjeneraten e parë të smartphone, një paisje celulare e dizenjuar për të qenë e lidhur me internetin gjate gjithë kohes, me një user interface tepër intuitive dhe një ekran i cili ndryshoi mënyren se si njerezit ndervepronin me smartphone. Një teknologji tjetër e rëndesishme e prezantuar nga Apple ishte dhe App Store, një mjedis virtual i dizenjuar për të marrë dhe instaluar applikacione direkt nga interneti në

smartphone. Të dy këto faktorë rriten popullaritetin e smarphone duke cuar keshtu në më shumë konkurrentë në këtë industri. U shfaq Google Android. Versioni i parë i këtij sistemi operativ u lëshua nga Google me Shtator te 2008 [150]. Një muaj më vonë, Google dhe HTC lëshuan dhe HTC Dream, smartphone i parë në market i cili përdorte sistemin operativ Android [146]. Në fillim , platforma nuk ishte shumë popullore pasi ishte e kufizuar vetëm në disa paisje. Megjithatë, duke filluar nga versioni i parë, zhvillues te sigurisë, akademikë , madje dhe persona me qellime te këqija u interesuan në prodhimin e aplikacoioneve të këqija për këtë mjedis te ri software [6]

Në Shtator të 2008, grupi i kërkimit mbi sigurinë Blitz Force Massada nga Universiteti i Shkencave te Elektronikës dhe Teknologjisë në Kinë, lajmëruan mbi sulmin e parë të mixuar mbi platformën Android. Ishte një koleksion i më shumë se 30 sulmeve të ekzekutuare në katër module të ndryshme [147] [6].

- Android/CallAccepter.A [148] – Detyronte paisjen të pranonte çdo thirrje
- Android/Radiocutter.A [149]– Fikte radion, duke ndaluar thirrjet e dërguara/marra
- Android/SilentMutter.A [150]– Detyronte paisjen të fikte të gjitha thirrjet
- Android/StiffWeather.A [151] – Mblidhte informacione sensitive dhe ja dërgonte sulmuesit

Ka informacion shumë të limituar mbi këto aplikacione në internet. Megjithate, qellimi i ketyre proof-of-concept nuk ishte shkaktimi i dëmit. Përkundrazi, objektivi kryesor ishte demonstrimi, në fazat fillestare te Android, se ky sistem operativ ishte vulnerabël ndaj kodeve te këqija. Ashtu si me paisjet jo të lëvizshme, një nga objektivat kryesore të kodeve të këqija ishte të mblidhte sa më shumë informacion privat mbi perdoruesin pa djeninë e tyre. Këto tipe malware quhen ndryshe dhe spyware dhe sigurisht, Android nuk është imun ndaj tyre [6].

Në Nentor të 2009, Retina-X Studios shfaq spyware e parë të botes për Android [152], Mobile Spy [153], i cili monitoron në mënyrë të fshehur paisjet duke përdorur web

browserat, thirrjet telefonike, mesazhet, fotot, videot, GPS, madje dhe URL e vizituara nga paisja. Një karakteristikë e rëndesishme e ketij aplikacioni ishte metoda e instalimit: sulmuesi duhet të ketë akses fizik mbi paisje në mënyrë që të instalojë këtë aplikacion pasi aplikacioni duhet të kopjohet në mënyrë manuale në paisje [154]. Një karakteristikë tjetër e Mobile Spy është ekzekutimi në "stealth mode" , në background , pa nje ikonë të dukshme, pra përdoruesi nuk është në djeni që po monitorohet nga ky aplikacion [6].
Pak muaj më vonë në Janar te 2010, First Tech Credit Uninon dhe një entitet tjetër financiar si Travis Credit Union [155] raportuan ekzitencen e disa aplikacioneve që mund te ishin të dëmshme në Android Market. Në Dhjetor të 2009, zhvilluesi Droid09 postoi një aplikacion bankar jo zyrtar duke shënjestruar banka të ndryshme [156]. Qëllimi kryesor i atyre aplikacioneve ishte marrja e infromacioneve personale mbi llogaritë e përdorueseve të bankave, por kur çështja u bë publike, Google i hoqi keto aplikacione të dyshimta, duke lëne kështu kërkuesit e sigurisë dhe kompanitë e Antiviruseve pa një mostër për të analizuar. [156,157] Një konkluzion i rëndësishem u arrit nga Graham Cluley nga Sophos: "Android Market, nuk është aq i monitoruar sa ekuivalenti i Apple. Kjo kombinuar me levizjen dhe zërat mbi smartphone e ri të cilet përdorin Android si Motorola Droid dhe Google Nexus One, mund që në të ardhmen, ta shndërrojnë platformen me atraktive për personat me qëllime të keqija" [157] [6].

Nga spyware tek phishing attacks, evolucioni i malwareve të Android bëri një hap gjigand: rootkits. Ne Qershor te 2010, Christian Papathanasiou dhe Nicholas J. Percoco nga Trustwave prezantuan artikullin " Ky nuk është Droid qe po kerkoni" [158]. Ne kete artikull ata zhvilluan nje rootkit ne nivek Kerneli i cili ishte i aftë të dërgonte një "reverse TCP" mbi 3G/Wi-Fi tek sulmuesi pas marrjes se nje thirrje nga një "trigger number" [158]. Ky rootkit mund te shpërndahej në ajër ose mund të instalohet bashkë me një aplikacion të dëmshëm, mënyrë kryesore e përhapjes në ditët e sotme e malwareve të Android. Pavarësisht kompleksitetit të rrezikut, deri me tani, ky lloj sulmi nuk është pare në qarkullim [6].

Në Gusht të 2010, Dennis Maslennikov i Kaspersky raportoi zbulimin e të parit SMS Trojan për Android. [159] I njohur si “FakePlayer”, ky aplikacion i dëmshem hiqej si një aplikacion “media player” me një ikonë fallco te Microsoft Windows Media Player dhe titullin “ Movie Player” mbi imazhin. Në fakt, ky aplikacion kopjonte sjelljen tipike të zbuluar tek malware i Symbian si SymbOs.Mosquit.[160] Ai dërgon SMS drejt shërbimeve premium pa dijeninë e përdoruesit. Resultati ishte një faturë e majme mbi perdoruesin, në bazë të SMS-ve të dërguara. Në këte rast janë dy numra premium të perfshira, 3353 dhe 3354, dhe kostoja e cdo SMS ishte afersisht 5$. Megjithatë, për momentin, megjithëse cdo paisje mund të infektohet, ky Trojan funksionin ploteshisht vetëm për përdoruesit nga Rusia dhe nuk po përhapet më tej në Android Market.[159] Po në Gusht 2010 Symantec zbuloi një verison të modifikuar të lojes “snake” [161]. Ky aplikacion kishte aftesine të mblidhte (cdo 15 minuta) dhe dërgonte koordinatat GPS drejt një serveri të largët dhe të monitoronte vendndodhjen e një paisjeje pa djeninë e përdoruesit. Informacioni mund te shfaqej duke përdorur një tjetër aplikacion komercial , GPS Spy, i cili shkarkonte të dhënat që do të shfaqeshin në Google Maps. Ashtu si me Mobile Spy, sulmuesi duhet të kishte akses fizik tek paisja e viktimes në mënyrë që të instalonte aplikacionin. Gjithashtu ky nuk ishte një rrezik tepër i lartë pasi nuk ishte shumë i përhapur dhe impakti i tij ishte minimal [6].

Në Nentor, dy aplikacione interesante u shfaqën në market. I pari ishte “Angry Birds Bonus Levels”. Ky aplikacion u krijua nga kërkuesi Jon Oberheide i cili zbuloi disa brishtësi në sigurinë e Android. [162] Ky proof-of-concept konsistonte në një aplikacion i cili ishte i afte të shkarkonte dhe instaklonte aplikacione te tjera nga Android Market. E vetmja konsekuencë e këtij demostrimi ishte fshirja e këtij aplikacioni nga Androd Market. Megjithatë , nuk kishte asnjë lajmërim zyrtar nga Google mbi këtë brishtësi , dhe në baze të këtij fakti as sot nuk është e qarte nëse kjo brishtësi është rregulluar dhe se si [6].

Aplikacioni tjeter i lidhur me mungesën e sigurisë në platformën Android u shfaqe në Tetor të 2010 dhe ishte “SMS Replicator”. Në fakt, ky ishte një aplikacion spyware klasik

i cili në mënyrë të heshtur dërgonte SMS e marra nga përdoruesi drejt një numri tjëter. Pjesa interesante e këtij aplikacioni ishte ekzistenda e një “dere të pasme”. Sulmuesi mund të konfigurojë numrin tek i cili dërgohen SMS e marra nga përdoruesi i infektuar duke dërguar një mesazh të shkurtër me një password të konfiguruar më parë. Megjithatë, ishte një password i fshehur universal (red4life) i cili lejonte konfigurimin e cdo numri në paisjen e infektuar [6].

Në fund të 2010, Lookout lajmëroi zbulimin e Geinimi [163], i cili ishte në atë kohë Android malware më i sofistikuar i gjetur në treg. Deri në atë kohë, vetëm një Trojan (i cili sulmonte vetëm në rusi) dhe disa spyware ishin gjetur në Android Market zyrtar dhe jozyrtar [164] [6].

- Innovative distribution method – Gemini ishte i futur brenda aplikacioneve legjitime si Monkey Jump 2, President Versus Aliens, City Defense dhe Baseball Superstars 2010. Kodi konsistonte në disa klasa të cilat ishin shtuar brenda aplikacionit ekzistues për të përfshirë një funksionalitet “backdoor”. Megjithatë, këto aplikacione u uploaduan në markete Android jozyrtare në Kinë, dhe nuk u gjetën në Marketin zyrtar të Android, pasi emri i paketës finale të infektuar ishte i njëjti me atë të versionit të legjitimuar të aplikacionit gjë që çonte në konflikt në Google Application Market [165].
- Dual anti-analysis method – Encryption dhe obfuscation. Dekriptimi i komunikimit midis serverit te largët dhe paisjes së infektuar nuk ishte dicka e veshtirë pasi përdoreshin algoritma të dobët. Megjithatë, Geinimi komplikonte analizën statike të kodit të dëmshëm sepse përdorej gjithashtu për të turbulluar(obfuscate) source code-n.
- Botnet like capabilities – Në fakt , ai kishte një liste me më shume se 20 komanda të inplementuara në source code. Megjithate, Lookout dhe disa kompani të tjera sigurie nuk ishin të afta të dallonin një server të dërgonte komanda drejt paisjej së infektuar. Për këtë arsye, nuk u konfirmua që qellimi i ketij malware ishte te krijonte një botnet.

Geinimi demonstroi mundësinë për të infektuar një aplikacion legjitim duke i bërë repack me kodin e dëmshem. Kjo teknikë shpejt do të inplementohej nga variante të tjerë të Geinimi si Trojan ADRD(ose HongTouTou [166]), i zbuluar nga Aegis Lab [167] në Shkurt. Ndryshe nga Geinimi, ky malware infektonte wallpapera-t, pasi keto lloje aplikacionesh nuk rapresentohen nga një ikonë në panelin kryesor të aplikacioneve, gjë që e bënte më të fshehtë se Geinimi. Diferenca tjetër midis tyre ishte se ADRD kishte funksionalitete të limituara. Ai dërgonte vetëm IMEI/IMSI e paisjese së infektuar tek një servër i largët dhe kërkonte të dërgonte një kërkesë drejt një search engine popullor në Kinë , duke shkaktuar konsumimin e bandwidth-it.

Një tjetër shembull i aplikacioneve legjitime te "Trojanizuara" është Android.Pjapps, i cili ishte i aftë të instalonte aplikacione, navigonte në website , të shtonte bookmark-e në browser, të dërgonte mesazhe, dhe të bllokonte dërgimin e mesazheve. [168] Pjapps gjithashtu kishte dhe funksionalitet te tipit botnet, duke suportuar komanda te derguara nga nje server te ciliat tregonin se cfarë aksioni do të performohej nga paisja e infektuar. Megjithate, ashtu si Geinimi, të gjitha komandat dhe serverat e kontrollit nuk ishin aktiv gjatë momentit të analizimit. Pra, përseri është e vështirë të konfirmosh që ke të bësh me një botnet [168].

Masa e sigurisë primare e rekomanduar përdorueseve te Android ishte të " perdornin vetem Androit Market zyretare për të shkarkuar dhe instaluar aplikacionet " dhë të përdorin" opsionin e ndalimit te instalimit te aplikacioneve te Android Market jo zyretare" [168]. Mgjithate , kjo ishte gati të ndryshonte me zbulimin e DroidDream [169], një malware i gjendur në më shumë se 50 aplikacione të ndryshme në Android Market zyretar [6].

Në Mars, një user Lompolo raportori në reddit, [170] një faqe sociale lajmesh, ekzistencën e versioneve pirate të aplikacioneve legjitime të ngarkuara në Android Market nga zhvilluesi "Myournet". Detjajet e postit të cilat ngriten alarmet ishin mundesia e perdorimit të "rageagainsthecage" , i cili deri ne ate kohe ishte perdorur per te marre akses ne nje paisje pa qëllime të këqia. Një pasojë tjetër e dushkme e ketij malware ishte që "aplikacionet dukeshin sikur postonin kodet IMEI/IMSI tek

http://184.105.245.17:8080/GMServer/GMServlet “. Në fund dhe jo më pak e rëndesishme, ky malware ishte i aftë të instalonte një aplikacion që ishte brenda .apk (DownloadProvidersManager.apk) po i fshehur (sqlite.db) [170]. Pavarësisht faktit që përdorte të njëjtën teknikë ripaketimi të aplikacioneve legjitime si vektor shpërndarës, dallueshëm nuk ishte varjant tjetër i Geinimi. DroidDream prezantoi përdorimin e “publicity disclosed exploits” në mënyrë që të merrte të drejta administratori në një paisje dhe të merrte kontrollon total të smarphone. Por më e rëndësishme ishte fakti që ky malware arriti në Android Market një numër shkarkimesh nga 50,000-200,000 [171] [6].

Reagimi i Google ndaj ketij fakti ishte fshirja e të gjithe aplikacioneve të këqija dhe largim i perhershëm i zhvillueseve që i hodhen ato. Megjithatë, Google bëri më shumë se fshirjen e ambjentit, Google hoqi aplikacionet e këqija nga paisjet e prekura duke përdorur “Android Market Security Tool March 2011”, e cila eliminonte aplikacionet e këqija të instaluara në paisje. [172] Disa dite më vonë, Symantec raportoi zbulimin e një versioni të këtij aplikacioni [173] por tek i cili brenda ishte ripaketuar një malware. Ky kod u zbulua nga AegisLab dhe u quajt Fake10086 [174] pasi një nga qellimet e ktij malware ishte te komunikonte në fshehtesi me një server të largët dhe te bllokonte disa SMS hyrese që vinin nga 10086, te cilat paralajmeronin viktimën [6].

Më vonë atë muaj, një aplikacion popullor për Android u modifikua dhe ngarkua në disa website “file sharing”. Versioni 1.3.6 i “Walk and Text” lejon përdouresin të shkruaj në paisje ndërsa po ecën. Versioni i ligë i Walk and Text “ është malware i parë i zbuluar në treg i cili mundohet të disiplinojë përdoruesit të cilët shkarkojnë aplikacione në mënyre ilegale nga site të pa autorizuara” [175] duke u treguar një mesazh fals i cili indikonte që aplikacioni legjitim po krakohej, ndërkohë që dërgonte SMS tek të gjithë personat në listën e kontakteve. SMS thoshte “ Hey, just downloaded a pirated App off the Internet, Walk and Text for Android. Im [sic] stupid and cheap, it costed only 1 buck. Don’t steal like I did!” Mesa duket, objektivi i vetëm i aplikacionit ishte të dënonte përdoruesin për

shkarkimin e një versioni pirat të aplikacionit nga një market jozyrtar, dhe gjeneronte një faturë të majme për SMS e derguara. Megjithatë, aplikacioni gjithashtu mbledh informacione personal nga paisja , si numrin e paisjes dhe IMEI, dhe të dhënat i dërgonte në një server të largët. [175] Në fund, versioni i keq i Walk and Text shfaq një mesazh, duke i sugjeruar përdoruesit të kontrollojë faturën dhe duke i rekomanduar që të blejene aplikacionin zyretar nga Android Market. [176]

Pak kohe më vonë u zbulua një malware i ri në Android Market zyretar. AegisLab zbuloi "zsone", i cili është username i zhvilluesit i cili postoi "10" ne Android Market. [176] Sjellja e këtij malware ishte tepër e ngjashme me atë të Fake Player pasi ai dërgonte vetëm SMS drejt shërbimeve premium pa autorizim [6].

Muaji Maj ishte dhe muaji i SMS Trojan në Android pasi një malware i ngjashëm u shfaq disa ditë më vonë : Android.Smspacem. Një verison i piratuar i aplikacionit të ligjshëm "The Holy F***ing Bible", u zbulua nga Symantex, dhe kishte si qellim kryesor të dërgonte një "the end of the world" spam drejt të gjithë kontakteve të ruajtura në paisje kur data në paisje ishte 21 Maj 2011.[177] Mesazhet e dërguara zgjidheshin në mënyrë të cfardoshme nga kjo listë:

- Cannot talk right now, the world is about to end
- Jebus is way over due for a come back
- Is the Raptures, praise Jesus
- Prepare to meet thy maker, make sure to hedge your bet juss in case the Muslims were right
- Just saw the four horsemen of the apocalypse and man did they have the worst case of road rage
- Es el fin del mundo

Ky malware gjithashtu ndryshonte wallpaper-in e paisjes dhe vendoste një imazh të Stephen Colbert(Komedian Amerikan). Këto ishin shenja të dukshme të infektimit. Ndër aftesitë e tjera të ketij malware, kur paisja restartohej, aplikacioni i keq mundohej të

kontaktonte një “command and control server” cdo 33 minuta në mënyre që të dërgonte informacione private nga paisja dhe gjithashtu të merrte një nga këto komanda: [179] [6]

- Pacem — Regjistronte cdo kontakt të telefonit tek “ColbertPAC”
- Formula401 — Dërgonte “download links” drejt cdo kontaktio.
- Health — Paisja kthente “I am infected and alive ver 1.00”

Pra asnjë nga këto nuk është gjë e re. Megjithatë, strategjia e komunikimit e cila përdorej për të krijuar një botnet ndryshonte nga malware e mëparshëm si psh Geinimi. Ky malware procesonte komandat nëpërmjet një “web service” duke përdorur protokollin “simple object access protocol(SOAP)”, duke krijuar një objekt me një “XML namespace and command”, e cila ishte metoda me të cilën malware do të komunikonte me serverin. Pasi objekti SOAP krijohej, dy opsione do të shtoheshin (numri i telefonit i paisjes së infektuar dhe emri i operatorit) për të dërguar më pas objektin nëpërmjet HTTP drejt serverit të largët [6] [178]

Disa ditë më vonë, para Smspacem, në fund të muajit Maj, një malware i ri u gjet në Android Market: DoridDreamLight. Të paktën 26 aplikacioneve të infektuara u ngarkuan duke përdorur gjashtë llogari të ndryshme. Komponentet e keqija të DroidDreamLight aktivizoheshin sapo gjendja e paisjes ndryshonte (kur vinte një thirrje etj…), pra ky malware mund të ekzekutohej pa ndërhyrjen e përdoruesit. Ashtu si me shumicën e aplikacioneve të infektuara të para deri më tani në Android, DoridDreamLight mblidhte informacione të lidhura me paisjen (IMEI, IMSI, APN, Model, SDK version) dhe i dergonte të dhenat drejt një serveri. URL e serverit ishte e enkriptuar me DES dhe e ruajtur në skedarin “prefer.dat” së bashku me apk origjinale. Menjëhere pasi paisja identifikohej, ky malware tentonte të shkrarkonte dhe instalonte aplikacione të reja. Ndryshe nga DroidDream origjinal,ky malware nuk ishte i aftë ta bente këtë gje pa nderhyrjen e përdoruesit [6] [180].

Pothuajse në të njëjtën ditë, NetQin lajmeroi zbulimin e një malware të pranishëm në më shumë se 20 aplikacione, BaseBridge. [181] Strategjia e ketij malware ishte ajo e regjistrimit tek një shërbim SMS premium, me qëllimin e gjenerimit të faturave të majme

ndaj përdourueseve. Sipas NetQin, “Kjo ishte hera e parë që një malware “auto-dialing” i cili shkaktonte tarifa të larta për përdoruesit u gjet në një paisje Andoroid (edhe pse malware i tillë është gjetur më parë në paisjet Symbian) gjë që tregon që malwaret në Android po diversifikohen” [181]. Në fakt, BaseBridge ishte i aftë të përgjigjej ndaj thirrjeve hyrese pa nderhyrjen e përdoruesit dhe kishte mekanizma të cilat e conin volumin në “0”, gjë që mund të përdorej për të spiunuar viktimën [182]. Një karakteristikë interesante e BaseBridge ishte aftesia nëse programi i sigurisë “360 Safeguard” po ekzekutohej në paisje. Nëse po , një mesazh falco do të shfaqej, duke treguar që aplikacioni ka ndaluar punën për shkak të një errori,por , në fakt, 360 Safeguard vazhdonte të ekzekutohej në fshehtësi. BaseBridge përfshinte dhe ekzekutimin e “ragaisntthecage”, enkriptimin e “command and control server URLs” duke perdorur AES, dhe disa mekanizma që fshihnin operacionet e kryera nga paisja si fshirja e “call log”, bllokimi i mesazheve nga “moblie provider” në mënyrë të tillë që të shmangte mesazhet mbi faturimin në fshehtësi [6] [182]

Në fund të muaijt Maj , një malware i ri u zbulua në Adroid Market zyretar, DoridKungFu. Ky malware ishte i aftë të fshihej në rrënjet e paisjeve Android dhe përdorte “RageAgainsTheCage” dhe “CVE-2009-1185” fillimisht të implementuara nga DroidDream . Por, ndryshe nga paraardhesit, DroidKungFu perdote AES për të enkriptuar “RageAgainsTheCage” dhe “CVE-2009-1185” , në mënyre të tillë që të shmangte detektimin e tyre nga antivirusi i paisjes. [183] Përvec diferencave, sjellja e këtij malware ishte e njëjtë si e DroidDream. Pra mblidhte informacione mbi paisjen, dhe instalonte një aplikacion të dytë i cili mund të shkarkonte malware të tjera në paisje.

Në këtë pjesë u fol për të shkuarën dhe zhvillimin e malware. Në seksione e mëposhtme do të prezantohen metodologji dhe mjete për analizimin e malware [6].

4.4 Struktura e Sistemit Operativ Android

Analiza e Android malware kërkon dhe njohuri mbi platformën Android dhe se si aplikacionet ekzekutohen në këtë platformë. Arkitektura e Android jepet më poshtë [6]:

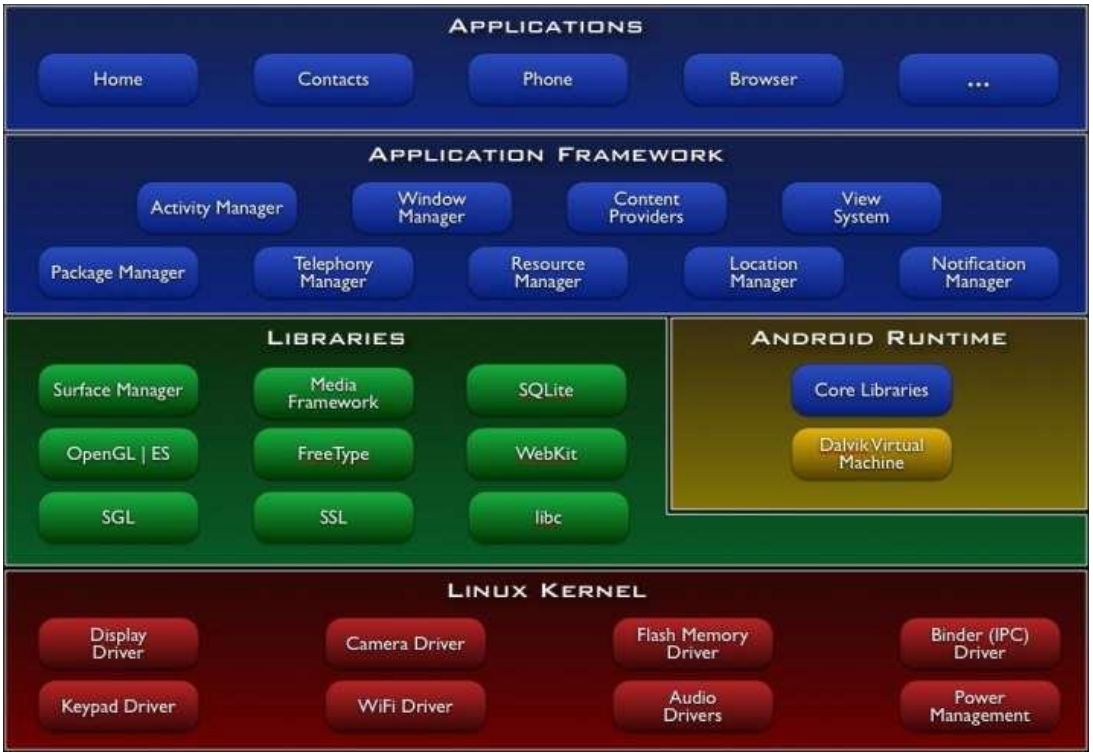


Fig. 1 Struktura e Android

Komponenti qëndror i Android është një kernel me bazë Linux i cili shërben si urë midis hardware të paisjes dhe komponentëve software të sistemit. Për këtë arsye, është e mundur të aksesos një paisje Android duke përdorur një shell me baze Linux dhe të ekzekutosh komanda për të listuar përmbajtjen e një direktorie të sistemit. Gjithashtu, është e mundur të marresh hua disa "tools" të përdorura në Linux dhe ti ekzekutosh në Android por, që të arrishe këtë gjë, është e nevojshme të gjenerohet një kod i ekzekutueshëm për një platformë ARM, i cili është procesori kryesor për paisjet Android. Mbi shtresën e Kernel, jane disa librari open source c/c++ te cilat mundësojne ndërveprim ne nivel të ulët midis framework të aplikacionit dhe komponenteve të kernel. Një komponent i rëndësishem në këtë shtresë është SQLite, i cili është një librari software që inplementon "një transactional SQL database engine". [184] Aplikacionet

Android zakonisht përdorin databazat SQLite për të ruajtur të dhëna të përhershme në paisje duke perfshirë, informacione personale të cilat mund të aksesohen për qellime të keqija nëse nuk janë të mbrojtura sic duhet. Një shembull e kësaj është një version vulnerabel si Skype per Android i zbuluar nga Android Police ne Prill të 2011. [185] Vulnerabiliteti qendronte një përdorimin "pa leje" të databazes SQLite për të ruajtur informacione personale të përdoruesit , si informacionet e profilit dhe mesazhet. Për më tepër, të dhënat ruheshin në "plain text" gjë që lejonte cdo person ose aplikacion të mblidhte informacione private të lidhura me llogarine Skype të përdoruesit. Pavarësisht rregullimit nga shitesi i Skype të këtij problem [186], ky incident demonstron që databazat DQLite janë një objektiv kryesor për malware të ndryshëm [6].

Një komponent tjetër kryesor në arkitekturën e Android është Dalvik Virtual Machine. Pavarësisht faktit që shumica e aplikacioneve në Android janë shkruar duke përdorur gjuhen e programimit Java, Dalvik VM (register - based) nuk është nje Java Virtual Machine (stack - based). Për këtë arsye, kur një aplikacion për Android kompilohet, source code është, në fakt, kopiluar në file .class , por menjeherë "dx tool" përdoret për të konvertuar këto .class në një dex file.

Shtresa tjetër në arkitekturën e Android është Application Framework, e cila është projektuar per të ofruar funksionalitete të përgjithshme , për ti siguruar zhvillueseve një mjet për krijimin e aplikacioneve. Sebashku me këto komponente software, Android mundëson një API (applications programming interface) të dokumentuar, e cila është e nevojshme për kuptimin e Java source code të shkruajtur nga zhvilluesi. Për analizimin e malware, është e rendesishme të kuptohen komponentët e ndryshëm që mund të gjendën në aplikacionet Android: [6] [187]

- Activities – Përfaqëson një ekran të thjeshtë me një ndërfaqe për përdoruesin. Zakonisht ky lloj komponenti implementohet nga aplikacionet origjinale të cilat po infektohen nga kode të keqija.
- Services – Karakteristika kryesore e ketij komponenti është që ekzekutohet në background. Për këtë arsye, përdoret shpesh nga kodet e këqija të cilat po paketohen brenda aplikacioneve të ligjshme.

- Content providers – Qëllimi i këtij komponenti është të mundesojë aftësine për të shkëmbyer të dhena midis aplikacioneve. Ky lloj komponenti zakonisht nuk përdoret nga malware.
- Broadcast recivers – Ky komponent "përgjigjet ndaj lajmerimeve System-Wide ". Zakonisht përdoret nga malware për të "ndezur" një funksionalitet të caktuar kur një event ndodh në paisje.

Shtresa e fundit e arkitektures së Android është vendi ku aplikacioni final kompilohet dhe është gati te instalohet dhe ekzekutohet në sistem. Formati i këtij file quhet Android Package (.apk), qe është, në fakt, një file i kompresuar që përmban komponentët e meposhtëm:

- Manifest – Qëllimi i këtij skedari është të deklarojë të gjithë komponentët, kërkesat, dhe lejet e kërkuara nga aplikacioni që të ekzekutohet në sistem. Ky skedar shkruhet duke përdorur strukturën standarte XML dhe duhet të jetë prezent në rrenjë të direktorise së aplikacionit.
- Classes.dex – Brenda këtij skedari ndodhet Dalvik bytecode i cili duhet dekompiluar në mënyrë që të performohet një analizë statikë mbi aplikacionin.
- Resources – Përmban të gjithë informacionin e lidhur me resorset e përdorura nga aplikacioni.

Duke u bazuar në këte pershkrim të Android, seksioni i mëposhtem flet për metodat dhe mjete e përdorura për të analizuar një malware [6].

4.5 Metodologjitë dhe mjetet për analizimin e Android Malware

Malware-t mund të analizohen duke përdorur dy teknika të ndryshme dhe konplementare: analizën dinamike dhe analizën statike. E para konsiston në ekzekutimin e një mostre në një mjedis të kontrolluar për të monitoruar sjelljen dhe për të vendosur nëse është i dëmshem dhe cfarë modifikimesh ka në sistem. Analiza dinamike përdoret shpesh për të

studiuar malware e kompjuterave desktop pasi sasia e mostrave të malware të gjeneruara cdo ditë është shumë herë më e madhe se sasia e fileve qe një njeri mund të analizojë me anë të analizës statike. Për këtë arsye, analiza dinamike zakonisht bëhet nëpermjet një procesi të automatizuar. Megjithatë , ajo nuk mundëson të njëjtën kualitet të dhenash si një analizë e detajuar statike [6]

Me paisjet celulare, analiza dinamike nuk është shumë e pëlqyeshme pasi, në mënyrë që të replikosh një malware, është e nevojshme të përgatitësh mjedise të ndryshme për shkak të ekzistences së platformave dhe sistemeve operative të ndryshme. Për me tepër, disa vite më pare shumica e malware (në vecanti ato per Symbian) përdornin lidhjet wireless si një vektor kryesor përhapjeje. Në ditët e sotme, vektori i perhapjes është i ndryshëm. Megjithatë, kur një malware replikohet në një paisje reale, masa sigurie duhet të merren në mënyrë që të parandalohet një infektim aksidental [6]

Analiza statike është me e përshtatshme për paisjet celulare pasi nuk ka një numer shumë të madh mostrash (krahasuar me ato për desktop) , dhe zakonisht kjo lloj analize mundëson të dhëna më të detajuara sesa replikimi i malware në mjedise testimi.

Në analizen statike, aplikacioni i infektuar që do të analizohet vjen si në file apk, e cila ka karakteristika të njëjta me një file jar: të dyja janë të ndërtuara në formatin .zip. Në mënyrë që të ekstraktosh te gjithe komponentet brenda nje file apk, disa mjete mund te merdoren qe te dekompresosh file kryesor si psh: Winzip, Winrar, dhe 7-zip midis të tjerave. Pasi të gjithë file dhe folderat ekstraktohen, analiza statike do të jetë e fokusuar në dy komponente: AndroidManifest.xml dhe classes.dex [6]

AndroidManifest.xml vjen në një format binar që nuk është zyrtarisht i dokumentuar nga Google. Megjithatë, ekzistojnë disa mjete të ideuara për të transformuar XML nga formati binar në një file text të qartë (AXMLPrinter2.jar [188]). Pasi file dekriptohet, është e rëndesishme ta analizosh atë për të parë nëse “permissions” e aplikacionit korrespondojnë me ato që supozohet që kërkohen (në varësi të qëllimit kryesor të aplikacionit). Për shembull, nuk është zakonshme për një lojë që të kërkojë “permission” për të dërguar mesazhe SMS pasi kjo nuk është e lidhur me një funksionalitet të aplikacionit. Një komponent tjetër i Manifest është seksioni ku definohen komponentet e

aplikacionit. Zakonisht, nëse aplikacioni nuk është tërësisht i infektuar, aktiviteti i definuar në Manifest korrespondon me atë të aplikacionit të pastër origjinal. Megjithatë, shumica e malware të sotëm modifikojnë Manifest në mënyrë që të shtojnë shërbime të infektuara të cilat ekzekutohen në fshehtesi pa djeninë e përdoruesit [6].

Pasi AndroidManifest.xml është analizuar dhe "permissions", komponentet dhe "entry points" të aplikacionit janë identifikuar, komponenti tjeter i apk që do të analizohet është classes.dex. Ajo mund të analizohet duke përdorur dy metoda të ndryshme por komplementare: Dekompilimi dhe/ose Deasemblimi i file dex. Mënyra e parë konsiston në performimin e operacionit të kundërt të kompiluesit për të përkthyer programin e ekzekutueshëm (Dex Bytecode) në source code java origjinal. Hapi i pare është të konvertosh dex file në një Java Archive (JAR). Ky proces mund të kryhet me Dex2Jar.rar [189]. Pasi merret JAR file, një dekompilues Java (si JD-GUI [190]) mund të përdoret që të marrësh një përafrim të source code origjinal të aplikacionit. Megjithatë Dext2Jar nuk është perfekte. Eshtë e rëndësishme të kuptosh që kodi Java i fituar nuk është ekzaktesisht i njëjtë si ai origjinali dhe disa here output-i i dekompiluesit mund të këtë mangësi në logjikë dhe mund të tregojë një kod i cili nuk mund të kthehet plotësisht në një kod Java të kuptueshëm. Në këtë pikë, mjetet e deasemblimit janë të nevojshme për të analizuar pjesë të kodit të cilat nuk mund të perktheheshin nga Dex2Jar.

Android ka një deasemblues vetjak ëe quajtur dexdump, i cili mund të gjendet në folderin "Android/platform-tools". Megjithatë, outputi i ketij mjeti nuk është shumë user-frendly pasi kodi nuk deasemblohet plotësisht. [191] Disa mjete të tjera janë zhvilluar për të dhënë një output më të lexueshëm si dedexer baskmali [192].

Pasi outputi gjenerohet nga një nga mjetet e permendura më lart, disa njohuri bazike të Java dhe Dalvik bytecode janë të nevojshme në mënyrë që të kuptohet qëllimi i kodit [6].

4.6 E ardhmja e Malware

Android malware ka evoluar nga një Trojan i thjeshtë që dërgon SMS drejt sherbimeve premium pa dijeninë dhe autorizimin e përdoruesit, në malware të sofistikuar të aftë të infektojnë aplikacione të ligjshme, enkriptojnë të dhëna duke përdorur "symmetric-key encryption (AES), të përhapet nëpërmjet Android Market zyrtar, të marrë të drejta

administratori në paisje duke përdorur dy exploite të ndryshëm, të marrë dhe ekzekutojë komanda në largësi, të sillet si një shkarkues duke instaluar aplikacione të tjerë pa dijeninë e përdoruesit, dhe në fund, të ngarkojë një “payload” një mënyrë dinamike nga një server i largët [6].

Sot , të gjithë shembujt e Android malware janë klasifikuar si Trojan Horse pasi të gjitha familjet e para deri më tani në treg nuk vetë-replikohen (kanë nevojë për një veprim nga përdoruesi) . Duke marrë parasysh që shumica e paisjeve Android kanë lidhje të përkohshme në rrjet, me shumë mundësi hapi tjetër në evoulimin e malware mund të jetë një Android worm i cili shfrytëzon nga distanca një “zero-day vulnerability” në sistem për të instaluar payload-in dhe të fillojë të kërkojë për paisje të tjera vulnerabel në internet.

Një step tjetër në këtë evolucion mund të jetë një Android rootkit. Ky lloj rreziku në kompjuterat tradicional mund të mundësojë akses të privilegjuar të vazhdueshëm duke fshehur prezencën e tij. Duke marrë parasysh që (1) sot disa malware janë të aftë të marrin të drejta administratori në paisje dhe (2) tashmë një kernel-level Android rootkit me këto karakteristika është shumë pranë realitetit. Për më tepër, Android rootkits mund të jenë më të vështirë për tu fshirë , në vecanti nëse ata janë rootkite në nivel kerneli pasi ata mund të shtojnë ose zëvendësojnë kode direkt në bërthamën e sistemit operativ.

Si përfundim, për tju shmangur metodave tradicionale të detektimit, malware polimorfike dhe metamorfike janë një tjetër mundësi. Këto veti kanë si qëllim ndryshimin e formës të cdo instance të malware duke përdorur enkriptimin ose “appended/pre-pended data”. Strategjia e parë ësht parë tek DroidKungFu, i cili përdor AES për të enkriptuar dy exploit-et të cilat më vonë lancohen për të marrë priviligje administratori në paisje. Një mekanizëm i avancuar mund të ndryshojë kodin e malware në mënyrë automatike cdo herë që ekzekutohet. Të dyja karakteristikat do të shtonin levelin e kompleksitetit të detektimit dhe fshirje së këtij lloj rreziku. [6]

KAPITULLI 5

Sulmet në SO Android

Në këtë kapitull, do të diskutojmë disa lloje sulmesh që i bëhen smartphone-ve. Në fillim do të përshkruajmë metodologjitë e mundshme për të kryer një sulm dhe për secilin prej tyre do të jap edhe shembuj realë. Së dyti, do të tregojmë se si këto metodologjji mund të shfrytëzohen për të arritur qëllime të ndryshme [7].

5.1 Metodologjitë e sulmeve

Metodologjitë e ndryshme që përdoren kundra smartphone-ve kategorizohen duke përdorur klasat në vijim [7]:

1. Wireless;
2. Break-in (me vjedhje);
3. Infrastructure-based;
4. Worm-based;
5. Botnet;
6. User-based;

1) *Sulmet Wireless*: Janë disa lloje sulmesh të ndryshme wireless për smartphone-t, veçanërisht ato synojnë të dhëna përsonale dhe sensitive. Sulmi më i zakonshëm është vjedhja e transmetimit wireless për të nxjerrë informacione konfidenciale, siç janë username-et dhe password-et. Sulmet wireless gjithashtu mund të përdorin identifikimin unik të hardware-it(p.sh. adresën wireless të LAN MAC adress) për të gjurmuar ose profilizuar pronarin e pajisjes. Së fundmi, malware-t zakonisht shfrytëzojnë mjedisin e Bluetooth për të përshejtuar përhapjen.
   [197] diskuton problemet e sigurisë në mjediset ëireless dhe tregon aktivitetet e kërkimet të fundme. Një pasqyrë më gjithpërfshirëse për sulmet me Bluetooth që bëhet në smartphone-t mund ta gjejmë te [198]. Disa studime se si ti parandalojmë këto lloj sulmesh mund t'i gjejmë në [197,199, 200, 201].

Shembull-Cabir: Cabir është një virus i cili përhapet përmes Bluetooth. Ky virus konsiston në një mesazh i cili përmban një file aplikacion, cabir.sis i cili duket si një pjesë e Menaxherit të Sigurisë. Në qoftë se instalohet, virusi përdor funksionalitetet të pajisjes së Bluetooth-it për të kërkuar për pajisje të tjera Bluetooth. Pastaj, virusi gjithashtu tenton të dërgojë file-in e infektuar SIS te pajisja e zbuluar .

2) *Sulmet Break-in*: Një sulmet break-in i mundëson agresorit të marri kontroll mbi pajisjen e synuar duke shfrytëzuar gabimet e programeve, (p.sh. duke shkaktuar mbingarkimin e buffer-ve). Më së shumti këto sulme janë sulme paraprijës të sulmeve të mëvonshme si vjedhjen e identieteve/ të dhënave ose mbifaturim. Disa studime për parandalimin e këtyre sulmeve mund t'i gjeni në [202,203]. Shembull – Doomboot.A: Ky Trojan instalon file-a sistem binare të korruptuara në drive-in C:\ të pajisjes. File-et binare të korruputuara përmbajnë Trojanë të tjera, si Commwarrior, të cilat gjithashtu instalohen në pajisje.

3) *Sulmet Infrastructure-based*: Duke qënë se shërbimi që ofrohet nga infrastruktura janë bazike për funksionalitet bazë të smartphone-ve, si vendosja e një thirrje/marrja e një thirrjeje telefonike, sherbimet SMS dhe e-mail, impakti ekonomik dhe social i këtyre sulmeve është shumë i madh, aq sa ai që përshkruhet edhe te [204]. [205] vlerëson impaktin e sigurisë në ndërfaqën e SMS-ve në disponueshmërinë e rrjetit telefonik celular. Për shembull, në qoftë se një agresor është në gjendje që vazhdimisht të dërgojë mesazhe përmes disa portaleve në rrjetin SMS, grumbullimi i ngarkesës mund të mbingarkojë kanalin e kontrollit dhe, më pas, bllokimin e thirrjeve dhe komunikimit me SMS. Autori demostron që një agresor që mund të injëktojë mesazhe text nga Interneti mund të bllokojë shërbimin voice (zanor) në një zonë metroplolitane duke përdorur hit-lists që përmbajën jo më pak se 2,500 shënjëstra në jo më shumë se një kabëll modemi.

a) GPRS: Duke qënë se arkitektura e GPRS është e ngritur mbi infrastukturën e GSM-së, ajo përdor një arkitekturë sigurie bazuar mbi sigurinë e përshtatur për GSM (për një pasqyrë më të mirë mbi sulmet dhe kërcënimet në rrjetin GSM, shikoni [53]). Sulmet kundra GPRS mund të synojnë pajisjen, aksesin e rrjetit radio, rrjetin backbone, dhe ndërfaqën që lidh rrjetin GPRS me njëri-tjetrin ose me Internetin. Rezultati i këtyre sulmeve mund të kompromentojë sigurnië e end-user-it, mbifaturin të përdoruesit, zbulimin ose modifikim e informacioneve kritike, shërbim jo i disponueshëm, ose prishjen e rrjetit. Ne gjithashtu duhet të konsiderojmë që GPRS është më shumë e eksopozuar nga agresorët se sa GSM sepse ajo përdor teknologjinë IP, e cila është më tepër e cënuar.

   Sulmet kundra GPRS mund të jenë aktive dhe pasive: sulmet aktive kërkojnë ndërhyrje aktive të agresorit për të dëgjuar, modifikuar, dhe injëktimin e të dhënave në kanalin e komunikimit. Gjithashtu, nëse agresori nuk është pjesë e GPRS, agresori mund të përcaktohet si i jashtëm; në të kundërt, agresori përcaktohet si i brendshëm. Një agresor pasiv kemi kur një agresor ndërhyn në kanalin e komunikimit ndërmejet dy node-ve pa shqëtësuar komunikimin për të zbuluar informacione të vlefshme rreth të dhënave ose mesazheve të kontrollit.
   Sic përshkruhet në [206], janë 5 zona sensitive në sigurinë e GPRS që mund të ndërhyet për të kryer një sulm:

   1- stacionit mobile (MS) dhe kartës SIM: rezultati i këtyre sulmeve mund të jetë monitorimi i përdorimit të MS, shkarkimi i skedarëve të padëshiruara, kryerja e thirrjeve të padëshiruara. Agresorët në kartën SIM fillimisht bazohen në një çelës secret, i cili ruhet në kartën SIM të MS. Kur një agresor arrin ta

marrë këtë çelës, ai mund të ndërpresë shkëmbimin e të dhënave, ose klonimin e kartës origjinale SIM;
2- Ndërfaqia midis MS dhe SGSN (Serving GPRS Support Node): një agresor mund të kryejë sulme si DoS ose Man-in-the-Middle1. Në Dos një palë e tretë me qëllim të keq mund të bllokojë të dhënat e përdoruesit dhe sinjalin e trafikut duke përdorur pajisje special të quajtur bllokues, ose nxitjen e dështimit të protokolleve specifikë, ose falsifikim i të qënurit element i rrjetit;

3- Rrjeti backbone GPRS: ata i referohen teknologjise IP dhe Sistemit te Sinjalit 7 (SS7), të cilët përçojnë të dhënat e përdoruesti dhe informacionet e sinjalizimit. Sapo një keqdashës merr akses të rrjetit GPRS mund të kryej sulme të ndryshme, siç janë DoS, IP spoofing, kompromentimin e privatësisë ose dërgimin e sasive të mëdha të të dhënave te përdoruesi;

4- Rrjeti i paketave që lidh opëratorë të ndryshëm: ndërfaqa Gp që mundëson lidhjen midis rrjetit GPRS që i përket opëratorëve të ndryshëm dhe i ofron përdoruesve roaming. Qëllimet e sigurisë janë disponueshmëria e resurseve dhe e shërbimeve, dhe integriteti dhe konfidencialiteti i transferimit të të dhënave;

5- Interneti: ndërfaqa Gi lidh rrjetin GPRS me Internetin dhe ofruesit e shërbimeve që ofrojnë shërbime me përsonin marrës në mobile. Për arsye se ndërfaqa Gi mund të përmbajë çdo lloj trafiku, elementët e rrjetit GPRS dhe marrësi në mobile mund të ekspozohen ndaj një shumëllojshmërie të kërcënimeve që gjenden në Internet;

6- Një tjetër tip sulmesh kundra kartave SIM të GSM/GPRS është sulmi në side-channel që i lejon agresorit të marrë informacione sensitive nga side-channel, siç përshkruhet edhe në [207].

b) UMTS: Arkitektura e sigurisë së UMTS-së përcakton një bashkësi proçedurash gjatë komunikimit të tyre. Në kernel-in e saj të arkitekturës së sigurisë ndodhet mekanizmi i autentikimit të përdoruesit, i quajtur Ki, i cili ruhet në UICC (user's tampër-resistant Universal Integrated Circuit Card) dhe në rregjisterin korrespondues HLR (Home Location Register) të përdoruesit HN (Home Network).
[208] përshkruan disa dobësi në arkitekturën e sigurinë së UMTS-së që mund të shfrytëzohen nga agresorët për të lëshuar sulme DoS. Zakonisht, një agresor tenton të aksesojë mesazhet e pambrojtura të kontrollit në mënyrë që të manipulojë proçedurat specifike. Rezultati i pritur varjon nga kualitet i ulët i shërbimit e deri në DoS. Disa shembuj të këtyre sulmeve janë:

- Hedhjen e ACK-ve të sinjalit: një agresor monitoron për mesazhet e komandave të alokimit të TMSI-së që vazhdimisht forcon krijimin e një TMSI-i të re që, eventualisht, do të shkaktojë DoS te të gjithë përdoruesit në atë zonë;

- Modifikimi i mesazheve të pambrojtura të kontrollit të burimit radio (RRC): një agresor zëvendëson një RRC të vlefshme Connection Setup Complete me një mesazh RRC Connection Reject për të shkaktuar uljen e kualitetit të shërbimit ose një DoS për end-user-at;

- Sulmet Man-in-the-Middle, një agresor mund të luaj rolin si një stacion bazë fals i një viktime MS dhe, në të njëjtën kohë, i viktimës të staticionit bazë.

- Modifikimin e aftësisë fillestare të sigurisë së MS: një agresor modifikon një mesazh RRC Connection Request për të nxitur përfundimin e koneksionit. Për shembull, ai mund të shkaktojë një dëmtim serioz duke krijuar një numër shumë të madh kërkesash të vazhdueshme për lidhje;

- Modifikimi i mesazheve përiodike të autentikimit: kjo ndodh në qoftë se RNC (Radio Network Controller), në një mesazh ngacmimi të marrë, lëshon koneksionin, duke e shkëputur nga MS;

- Sinkronizimi SQN: një agresor mund të kërkojë për procedurën e risinkronizimit që të ekzekutohet njëherësh për një numër të madh përdoruesish, dhe ta bëjë vazhdimisht, duke e mbingarkuar HLR;

- EAP-ALA krijuar nga DoS: një agresor mund të imitojë një EAP-Respose/AKA-Client-Error message dhe ta dërgojë në EAP-Server për ta dëtyruar që të ndalojë protokollin ose ai mund të imitojë një njoftim EAP-Resopnse/AKA-Synchronization-Failure për ta detyruar serverin që të nxisë risinkronizimin e kushtueshëm të proçedurës.

Disa studime për dedektimin e sulmeve në infrastucures-based janë paraqitur në [7], [209], [210], [211].

Shembuj: [212] përshkruan një sulm ku një përdorues me qëllime të këqija shndërrohet si një stacion bazë GSM i vlefshëm të një përdoruesi UMTS dhe, si rezultat, ai mund të përgjojë në të gjithë stacionin mobile të iniciuar. [213] shqyrton realizueshmërinë e sulmeve DoS duke shfrytëzuar avantazhet e një rrjedhjeje specifike të gjetur në arkitekturën e sigurisë së UMTS-së. Sulmi në fjalë përfshinë modifikimin e RRC connection Request Message që përfshin aftësitë e pajisjeve të sigurisë së përdoruesit. Ky mesazh nuk mbron integritetin: në rast të një mospërputhjeje, koneksioni do të përfundojë, por gjatë këtij procesi burime të mjaftueshme do të konsumohen në të dyja anët. [7]

[204] karakterizon impaktin e kompromentimit në një shkallë më të gjerë dhe kordinimin e telefonave mobile në sulmet kundra rrjetit baze, duke demonstruar që një botnet i përbërë prej rreth 12,000 nyje të kompromentuara mund të degradojnë shërbimin e një zone area-code rreth 93%. Sulmi më i thjeshtë tenton që të ndalojë një përdorues të një rrjeti celular në dërgimin ose marrjen e telefonatave ose mesazheve text: sulmi kryhet nga një agresor që mund të kontrollojë një bashkësi smartphone-ësh të kompromentuar dhe mbingarkimin e një HLR-je specifike me sasi të mëdha të trafikut të të dhënave, në këtë mënyrë kërkesat e përdoruesit për të marrë shërbim nga HLR-ja hidhen poshtë.[7]

[214] diskuton tipet e dëmtimeve që mund t'ju shkaktohen smartphone-ve, si dhunim i privatësisë, vjedhje të identitetit dhe sulmet e shpërndara DoS.

4) *Sulmet worm-Based*: Karakteristika kryesore që i karakterizon sulmet e bazuara në worms janë:

- Kanali i transmetimit;
- Përhapjen e parametrave;
- Modelet e lëvizhmërisë së përdoruesit.

a) *Kanali i transmetimit*: Smartphone-t zakonisht janë pajisje me opsione të ndryshme koneksioni dhe, prandaj, ofron mënyra të ndryshme të rrugëzimeve të mundshme për vektorë të infektuar, siç janë:

1. Shkarkimi i skedarëve të infektuar gjatë lundrimit në Internet;
2. Transferimin e skedarëve malicious ndërmjet smartphone-ve duke përdorur ndërfaqën e Bluetooth-it;
3. Sinkronizimin e smartphone-it me një kompjuter të infektuar;
4. Aksesimin e një kartë memorje të infektuar;
5. Hapja e një skedari të infektuar bashkangjitur një mesazhi MMS.

Në vitet e fundit, Bluetooth është bërë një ndër protokollet wireless më të famshem dhe klasa e malware-ve që përdorin koneksionin Bluetooth për të infektuar pajisjet po rritet. worms-et e Bluetooth-it janë ndryshe nga klasat e tjera të worm-ve, një infekimi me Bluetooth kërkon që burimi i infektimit dhe viktima të jenë pranë njëra-tjetrës, psh. në një diametër rreth 20/30 metra [7]

b) *Shpërndarja e parametrave*: Përveç infektimit të pajisjes, worm-et gjithashtu mund të sulmojnë edhe rrjetin e komunikimit. Në këtë skenar, worm-et jo vetëm që kompromentojnë aftësinë e përdoruesit të përdorin smartphone-in e tij, por edhe rrjetin e tij gjithashtu. worm-et që shfrytëzojnë shërbimin e mesazheve (SMS/MMS), si rrugën e tyre të preferuar të infektimit, potencialisht janë më tepër të fortë, në terma

të shpejtësisë dhe në hapësirë përhapjeje, se sa me Bluetooth. Në fakt, këto worm-e gjithashtu dërgohen shumë thjeshtë duke përdorur një klik dhe mund të infektojë çfarëdo smartphone-i në çdo cep të botës me mundësi më të madhe suksesi të përhapjes.

c) *Modelet e lëvizhmërisë së përdoruesit*: Krahasuar me internetin, rrjetet mobile të telefonave kanë shumë karakteristika të ndryshme në lidhje me topologjinë, shërbimet, sigurinë dhe kapacitetin, pajisjeve dhe mënyrës së komunikimit. Këto karakteristika gjithashtu karakterizojnë edhe mënyrën se si modelet e reja të ëorm-ve përhapen: më e rëndësishja është se ato nuk kërkojnë lidhje me Internetin për përhapjen e tyre dhe, prandaj, mund të përhapen pa u dedektuar nga sistemet aktuale të sigurisë. Megjithatë, ëorm-et e mobile-ve mund të infektojnë disa pajisje duke përdorur sulmet proximity kundra pajisjve të cënueshme që gjenden fizikisht pranë. Për të modeluar përhapjen e këtyre worm-ve, kërkohen dy hapa:

    1. Ndërtimin e një modeli që përshkruan përpikmërisht se si pajisjet takojnë njëra-tjetrën;
    2. Kuptimin se si kodi i virusit shfrytëzon të dyja lëvizshmërinë e përdoruesit dhe kapacitetin e rrjetit.

Dinamika e përhapjes në afërsi varet nga dinamika e lëvizshmërisë të një popullsie në një zonë specifike gjeografike. Fatkeqësisht, një metodologji ideale për modelimin e lëvizshmërisë së një përdoruesi nuk ekziston: gjurmimi i kontakteve mobile të një përdoruesi reflekton sjelljen aktuale, por është e vështirë të përgjithësohet dhe kapet vetëm një nënbashkësie e të gjitha kontakteve për shkak të mbulimit gjeografik.

Për të modeluar një përhapje epidemike të malware-ve me proximity-based, lidhjet ëireless point-to-point, Micken and Noble [215] prezantojnë një frameëork i quajtur “probabilistic queuing” i cili merret me lëvizhmëritë e node-ve. Të kapësh shpërndarjen tërthore të rrjteve mobile, modeli prezanton nivele konektiviteti të ndryshme si rradhë të veçanta. Secila radhë paraqët një popullsi epidemologjike të ndarë. Një model

probabilistik radhe propozohet për të ndarë account-et për shpejtësitë e nyjeve dhe një rrjeti koneksioni jo homogje i detyruar nga pajisjet e lëvizshme.

Problemi i sulmeve të afërta i smartphone-ve diskutohet gjithashtu edhe në [216] ku autori sjell një model individual dhe ndërton një shprehje analitike për llogaritjet e vlerës së kontakteve dhe transmetimin e worm-ve. Në [217], një simulim event-driven prezantohet që kap karakteristikat dhe kufizimiet e rrjetit mobile. Simulatori modelon topologjitë reale dhe përgatit kapacitetin e infrastrukturës së rrjetit. Qëllimet e këtij modeli janë [7]:

- Përhapjet e modelve malware në rrjet për raste reale që të përcaktohet shpejtësia dhe ashpërisa;
- Të kuptohet se si impakti i sigurisë së rrjetit përhapet në rrjet;
- Theksimin e mbrojtjes së rrjetit kundra malware-ve.

Disa zgjidhje të tjera për t'ju bërë ballë sulmeve worm, të cilët bazohen në modelet e lëvizshme për përhapjen e worm-ve me anë të Bluetooth-it, studiohen gjithashtu edhe në [218,219,220]. Shikoni gjithashtu [221] për modelet analitike për epidemitë në rrjetet mobile.

Shembuj: [222] dhe [223] investigojnë nëse një shpërthim në shkallë të gjerë i worm-ve në Bluetooth është i zbatueshëm në praktikë. Autori përdor simulimet trace-driven për të ekzaminuar përhapjen dinamike të worm-ve të Bluetooth-it në popullsi të gjerë, duke treguar se koha fillestare e shpërthimit të këtyre worm-ve është shumë e rëndësishme. Rezultatet e tyre sugjerojnë që një worm i cili shfrytëzon Bluetooth-in mund të përhapet shumë shpejt. Si zgjidhje për mbrojtje, autori sugjeron lokalizimin e pikave të monitorimit në vendodhjet me shumë trafik.

[224] investigon përpjekjet që kërkohen për të strehuar një worm në telefonat windows. Autori ishte në gjendje të ndërtonte një prototip të worm-ve që mund të përhapen në mënyrë të pavarur nëse gjenden pika të dobëta në shërbimet mobile të windows-it që arrihen në rrjet. Autori citon që kufiri i poshtëm kohor kërkon ndërtimin e një worm toolkit-i dhe gjetja e dobësive në stack-un e protokolleve të rrjetit të telefonave windows kërkon afërsisht 14 javë punë të plotë [7].

[225] analizon në detaje worm-in e parë polimorfik që prek punimin e platformes windows CE në procesorët ARM, që njihen edhe si winCE.Pmcryptic.A. worm-i përhapet duke gjeneruar kopje të reja polimorfike të vetes së tyre çdo kohë dhe mund të ekzekutojnë disa veprime të padëshiruara në një smartphone të kompromentuar, duke përfshirë thirrje të numrave telefonikë [7].

5) *Botnet-et*: Deri vonë, rrjetet mobile kanë qënë relativisht të izoluara nga Internet-i, kështu ka qënë pak e nevojshme mbrojtja kundra sulmeve që mundohen të krijojnë botnet-e. gjithsesi, kjo situatë po ndryshon shumë shpjet që kur rrjetet mobile janë tani të mirë integruara me Internet-in. Prandaj, kërcënimet në Internet do të përhapen nga rrjeti mobile (ose anasjelltas), duke përfshirë botnet-et, duke qënë se smartphone-t mund të infektohen nga malware-t ato mund të shndërrohen lehtësishtë në botclient [226].
    a) *Command-and-Control (C&C*): Rrjeti C&C, përdort përhapjen e shpërndarë të mesazheve, detyrave, update-in e payload-ve midis bot-ve dhe botmasters (dhe anasjelltas), mund të ndërtohen duke përdorur Bluetooth, mesazhet SMS, Internet-in (p.sh. HTTP), peer-to-peer(P2P) ose cilido nga kombinimet e tyre.

    b) *Bluetooth C&C*: [227] investigon sfidat e ndërtimit dhe mirëmbajtjen e bodnet-ve mobile që komunikojnë me Bluetooth. Duke përdorur simulimet në gjurmët Bluetooth të mundshme publike, autori demostron që mesazhet C&C muund të përhapen afërsishtë në 66% të nyjeve të infektuara brenda 24 orëve të lëshuara nga botmaster-i. Për të reduktuar sasinë e trafikut të dukshëm nga ofruesi për të arritur dedektimin, në framework-et e zhvilluara vetëm një nënbashkësi e vogël bot-esh (atyre me shkallë me të lartë) komunikojnë direct me botmaster-in përmes kanalit celular (p.sh. SMS, cellular data). Këto node zgjidhen përmes shpeshtësië relative të kontaktit të tyre me pajisjet e tjera të infektuara: kur ndodh që smartphone-i i infektuar kalon përmes zonës së njëri-tjetrit, ata regjistrojnë identitetin e pajisjes tjetër. Pasi arrihen disa bashkësi pragjesh nga botmaster-i, nyjet me përioritet më të lartë të konektivitetit dërgojnë contact log-et e tyre te boatmaster-i i cili informohet për (i) se cilat pajisje janë nën kontrollin e tij (ii) se cila nyje mund të ndihmojë në shpërndarjen e menjëhershme të komandave. Botmaster-i gjithashtu shpërndan komandat dhe update-on payload-et përmes kësaj strukture hierarkike duke kontaktuar nyjet burim. Për shkak të prioritetit të madh të

konektivitetit, këto nyje mund të shpërndajnë payload-in ne një numër të madh nyjesh të infektuara direkt dhe pa kërkuar asnjë ndërveprim me botmaster-in.

SMS C&C: [228] propozon dizanjin e bonet-ve mobile proof-of-concept elastike edhe ndaj përcarjeve. Botnet-i përbëhet nga tre komponentë:

1- Vektorë për përhapjen e kodit bot te smartphone-i;
2- Një kanal për lëshimin e komandave;
3- Një topologji për të organizuar botnet-in.

Të gjitha komunikimet C & C janë kryer duke përdorur mesazhet sms. Për të fshehur identitenin e botmaster-it, nuk ka servera qëndrorë të dedikuar shpërndarjes së komandave, nga që ato lehtësisht mund të identifikohen dhe të hiqën. Në vend të tyre, një topologji P2P shfrytëzohet për ti lejuar botmaster-it dhe bot-ve të publikojnë dhe të kërkojnë për komandat në mënyrën P2P, duke bërë që dedektimi dhe prishja e tyre të jetë shumë e vështirë.

Hybrid C&C: [229] tregon që është shumë e lehtë të krijosh botnet totalisht funksional në iPhone-t e bërë jailbrake duke diskutuar dizajnin, implementimin dhe zhvillimin e botnet-ve në iPhone. Autori në fillim diskuton bonet-et e SMS-ve që më pas janë përmisuar me HTTP për të reduktuar numrin e SMS-ve që duhen dërguar për të kontrolluar bot-et. Së fundmi, autori tregon se sa e fuqishëm një bonet i tillë mund të jetë nëse agresorët kombinohen me P2P(Kademlia) me tipin hibrid SMS-HTTP.

Shembuj: Porras et al. [230], diku nga fundi i vitit 2009, disa përdorues të iPhone-ve jailbroken filluan të shikonin dirtare pop-up që i drejtonin viktimat në një website ku kërkohej një pagesë si dëmshpërblim për të hequr infektimin e malware-it. Kjo dobësi preku disa iPhone-a të bërë jailbrake të konfiguruar me shërbimin SSH me një password bazë të parazgjedhur. Duke skanuar disa adresa IP nga

Interneti për iPhone-a me SSH të aktivizuar, një agresor mund të upload-te një aplikancion të thjeshtë ransomware tek disa përdorues iPhone-i. Duke shfrytëzuar këtë dobësi, disa javë më vonë, një tjetër malware iPhone-sh (iKee.A) i konvertonte iPhone-ët në një worm që vetpërhapej për të infektuar iPhone-ë të tjerë. Këtë herë, ky virus arriti të infektonte më shumë se 20,000 viktima brenda një jave. Disa javë më vonë, një malware tjetër (iKee.B) u krijua, i ngjashëm me iKee.A: përpos vetëpërhapjes, aplikacioni iKee.B bot klient shtoi dhe shërbimin e C&C check-in që i lejonte botmaster-it të upload-te dhe të ekzekutonte komanda shell në të gjithë klientët bot të iPhone-ve. Ky shërbim i lejon bot-it të zhvillojë ose ta ridrejtojë iPhone-in e infektuar te një vendodhje e re C&C kudo qoftë në Internet. iKee.B gjithashtu inkorporon një veçori që tërheq të gjithë databazën e SMS-ve nga iPhone-i i viktimës.
[231] jep një përshkrim për Yxes, një nga malware-t e para për Symbian OS 9 dhe ishte hapi i parë drejt botnet-ve të telefonave. Sapo instalohet duke përdorur një çertifikatë të vlefshme, detyra kryesore e malware-it përfshin marrjen e IMEI-t dhe IMSI-n e telefonit, analizon kontaktet, vret aplikacionet e padëshiruara dhe përhapet. Si përfundim, malware-i fillon fazën e përhapjes duke dërguar një SMS te secila nga viktimat e reja me një link të një server-i infektues nga ku viktima mund të download-ë malware-in. Një nga problemet e Yxes është se kodi nuk përdor asnjë nga dobësitë e sistemit opërativ Symbian, por vetëm përdor funksionet e saja API në një mënyrë të zgjuar. Për këtë arsye, autori konkludon duke shënuar se koncepti i aftësisë, i futur nga Simbian, dështon për të ndaluar qëllimet keqdashëse, për dy arsye: (i) kriminelët kibernetikë arrijnë të kenë malware-in e tyre të shënjuar pavarësisht aftësisve që kërkojë; (ii)

mundësitë vetëm të garantojnë autorizim për veprime të caktuara por nuk mund të merren parasysh për një kontekst ose një qëllim.

Agresorët gjithashtu kanë marrë dhe aplikacione që janë shumë të përdorshëm dhe i kanë shtuar kod shtesë në të, si psh me Pjapps [232]. Disa përdorues vunë re se diçka nuk shkonte kur aplikacioni kërkonte më shumë leje nga ajo që ishte e nevojshme: Pjapps tentonte të krijonte një rrjet bot-sh nga pajisjet Android të kompromentuara. Ky sulm qartësisht demostron që agresorët i konsiderojnë pajisjet smartphone si platforma për krime kibernetike.

6) *Përdoruesi si një Vektor Sulmi*: sulmet e bazuara te përdoruesi përmbëhen nga çdo lloj ndërhyrje që nuk është e natyrës teknike. Shumica e malware-ve të smartphone-ve të sotëm nuk bazohen në dobësitë teknike, por mashtrojnë përdoruesin duke shkeluar mekanizmat teknikë të sigurisë [196]. Kjo është një nga klasat e rëndësishme të dobësive dhe shumica e studimeve janë kryer për të vlerësuar njohuritë e sigurisë nga mesatarja e përdoruesve; në veçanti diskutohet qëllimi i të gjithë mekanizmave të implementuara të sigurisë nga smartphone-t nëse mesatarja e përdoruesve nuk i njohin ato.
Shpesh, nëse përdoruesi qartësisht kupton se si të përdorë një mekanizëm specifik, ai mund të ketë vështirësi në kuptimin e të tjerave, mundësisht një mekanizëm i ri dhe i update-ar.
Sic [233] vë në dukje, shumica e sulmeve mashtrojnë përdouresin për të mbishkruar mekanizma e sigurisë, si në rastin e në abuzimit të një mardhënjëve të besuara, e cila mund të ndodhë kur një malware akseson librin e adresave të viktimës dhe dërgon vetë te kontaktet që ti besojnë përdoruesit të infektuar. Në një rast tjetër, një përdorues nuk mund të dallojë nëse një vecori është një funksionalitet legjitim ose një i imituar psh. në rastin e mesazheve të Bluetooth-it me përmbajtje keqdashëse.
Një sondazh i kryer nga Sophos[195] i pyeti përdouresit nëse smartphone-i i tyre ishte i enkriptuar: 26 % e përdoruesve u përgjigjën se të dhënat e tyre ishin të

enkriptuara, 50% e tyre thanë që nuk ishin të mbrojtur nesë telefoni vidhej ose humbej, dhe 24% e tyre nuk ishin të sigurtë nëse smartphone-i tyre ishte apo jo i enkriptuar. Këto rezultate treguan se duhet më shumë edukim në lidhje me rrezikshmërinë e sigurisë që kërkohet për smartphone-t.

Shembuj: Trojan-SMS.AndroidOS. FakePlayer.b është një Trojan për Android që i kërkon përdoruesit që manualisht të instalojë atë dhe i kërkon privilegje përdoruesit për të dërguar mesazhe SMS. Kështu që, përdoruesi në mënyrë aktive merr pjesë në proçesin e aktivizimit. Sapo instalimi ka përfunduar, nëse përdoruesi hap aplikacionin fallco, Trojan-i fillon të dërgojë mesazhe SMS te një numër premium pa dijeninë e përdoruesit.

Zeus MitMo[25] shfrytëzon inxhinerinë sociale për të infektuar smartphone-t dhe kryen opëracine online bankare. Si përfundim, një agresor dërgon një SMS me një link te një aplikacioni mobile të infektuar te përdoruesi: pastaj, nëse përdoruesi e instalon atë, agresori mund të lexojë mesazhet SMS të përdorura si një mënyrë e dytë identifikimi me shërbimin online bankar për të aksesuar llogarinë e përdoruesit.

5.2 Qëllimet e sulmeve

Në këtë pjesë, do të diskutoj në lidhje me qëllimet e agresorëve të cilat mund të jenë [7]:

- Privatësia;
- Përgjimi;
- Bllokimin e shërbimit;
- Mbi faturimi;

1) *Privatësia*: Sulmet e privatësië së smartphone-ve lidhen me situatat në të cilat vidhet integriteti dmth humbet konfidencialiteti, psh. kur telefonat humben ose vidhen. Në fakt, për shkak të madhësisë së vogël, smartphone-t mund të vidhen

ose humben më shumë se laptopët. Gjatë kohës që një pajisje nuk i përket pronarit, mund të jetë e mundur që dikush instalon një spyware në telefon; për më shumë, dikush mund të lexojë të dhënat përsonale, si listën e kontakteve ose mesazhet.

[234] prezanton se një kërkues ka detajuar një metodë proof-of-concept për të vjedhur të dhënat në Android duke përdorur kombinimn e e skriptimit cross-site dhe Javascript. [235] raporton se si Apple është paditur që gjoja lejon aplikacionet mobile të iPhone-it dhe iPad-it të dërgonë informacionet përsonale te rrjetet reklamuese pa pëlqimin e përdoruesit. Së fundmi [236] na jep një pamje të përgjithshme të çështjeve të fundme të sigurisë së iPhone-it dhe analizon pajisjet e pamodifikuara duke ekzaminuar cilat të dhëna sensitive mund të kompromentohen nga një aplikacion i shkarkuar nga App Store-i.

Një çështje më e gjerë, lidhur me mbrojtjen DRM (Digital Right Manager), diskutohet në [237]. Autori sugjeron një framework për DRM përsonale për smartphonet Motorola, ku përdoruesit ruajnë mbrojtjen DRM dhe kanë kontrollin mbi të dhënat përsonale. Së fundmi, sistemi DRM përsonal i lejon përdoruesve të përcaktojnë kontrollin dhe gjenerimin e liçensave për përmbajtjen e porosisë dhe transferimin në mënyrë të sigurtë të tyre te smartphone-t e tjerë. Një përdorues më pas është i aftë të përcaktojë dhe të kufizojë audiencën e menduar dhe sigurimin e skadimit të përmbajtjeve të kërkuara. Veç kësaj, pajisjet e përshtatshme DRM janë të afata që automatikisht të dedektojnë njëra-tjetrën dhe të shkëmbjenë kredencialet.

Një tjetër temë e rëndësishme lidhur me privatësinë është të qënurit i informuar për vendodhjen, psh. aftësia për të përcaktuar pozicionin gjeografik. Edhe pse kjo veçori ka përfitime të rëndësishme, ajo ngre disa implikime të rëndësishme të privatësisë për përdoruesit e smartphone-ve. Një studim i plotë mbi disa çështje të privatësisë lidhur me njohjen e pozicionit për smartphone-t prezantohet te [238].

Disa studime për parandalimin e këtyre sulmeve sugjerohen në [239, 240]. Një qasje për dizenjimin e zgjidhjeve kundra vjedhjeve të smartphone-ve përshkruhet në [241].

Shembuj: Objektivi i vjedhjes së kryer nga ZeuS MitMo [25] është informacioni rreth llogarisë bankare online. Duke shfrytëzuar disa teknika të inxhiniersë sociale dhe faktin se shumë kompani përdorin SMS si mjet i dytë identifikimi, ZeuS MitMo infekton smartphone-in dhe kryen opëracione bankare në vend të pronarit legal të llogarisë. Për të përmbushur detyrën e tij, pikë së pari, dergohet një SMS me një link te aplikacioni virus i zotëruesit të llogarisë bankare. Nëse përdoruesi e instalon aplikacionin, agresori merr kontrollin mbi telefonin e infektuar dhe mund të lexojë të gjitha SMS-të që janë shpërndarë. Nëse përdoruesi logohet në llogarinë bankare online, atij i kërkohet të vendosë numrin e tij të telefonit. Ky numër përdoret nga kompania për të përforcuar sistemin e verifikimit: në fakt, klienti merr në numrin e tij një mesazh text me të dhënat e transaksionit dhe një kod që të vendoset në website. Më pas, ndiqën hapat e mëposhtme nga agresori:

1) Agresori logohet me kredencialet e vjedhura duke përdorur telefonin e përdoruesit dhe kryen një opëracion specifik bankar që kërkohet autentikimi me SMS;
2) Një SMS me kodin e autentikimit dërgohet te pajisja mobile e përdoruesit nga sistemi i verifikimit. Software-i virus i cili është në punë në pajisjen e përdoruesit dërgon SMS-të te një sistem i kontrolluar nga agresori;
3) Agresori plotëson kodin e autentikimit dhe përmbush opëracionin.

[242] diskuton një sulm të shkarkueshëme drive-by, psh. kur një përdorues pa dashje shkarkon një malware (zakonisht një spyware) duke vizituar një website, te një iPhone 3GS që i lejon agresorit të vjedhë databazën e SMS-ve nga telefoni.

2. *Përgjimi*: Sulmet e përgjimit në smartphone bazohen tek sensorët si psh. mikrofoni, kamera, marrësi GPS. Këto sensorë lejojnë një shumëllojshmëri të aplikacioneve të reja, por ato gjithashtu mund të kompromentojnë seriozisht

privatësinë e përdoruesit. Nëse një smartphone kompromentohet, një agresor mund të aksesojë të dhënat e ruajtura në pajisje dhe gjithashtu përdorimin e sensorëve për përgjim dhe rregjistrim të të gjitha veprimeve të përdoruesit. Një sistem mbrojtje kundra sulmeve të përgjimeve tregohet në [243].

Shembuj: [244] përshkruan dizenjimin dhe implementimin e SVC (Steatlthy Video Captuer), një spyware që në mënyrë sekrete aktivizion kamerën e smartphone-it për të kompromentuar privatësinë e përdoruesit duke rregjistruar video private, me konsumim të paktë të energjisë dhe është i kujdesshëm ndaj antiviruseve komerçialë.

Soundminer [245] është një Trojan i pajisjeve mobile që është në gjendje të nxjerrë të dhëna përsonale nga sensori i audios. Të dhënat sensitive, si numri i kartës së kreditit, ose numri PIN, mund të nxirren duke përdorur ndërveprimin e të dy sistemeve me sistemin e menusë së telefonit. Si një shembull, Soundminer mund të nxjerrë numrin e telefonit destinacion duke analizuar audion dhe duke i dërguar këto të dhëna në remotë te një përson keqdashës. Disa privilegje kërkohen nga Trojan-i gjatë instalimit, si sigurimin e aksesimit të mikrofonit. Privilegje të tjera, në veçanti lidhje në rrjet ose ndërprerjen e thirrjeve telefonike, nuk kërkohen. Për këtë arsye, duke qënë se Soundminer-i nuk mund të ketë akses direkt në internet, transmetimi ka nevojë të bëhet përmes një aplikacioni të dytë, ose përmes një aplikacioni legjitim rrjeti ose përmes një programi me të drejta në rrjet. Kjo i lejon Soundminer-it t'ia hedhë mekanizmit që monitoron komunikimin midis dy aplikacioneve jo të besueshëm, siç propozohet në [246, 247].

3. *Bllokimin e shërbimit*: me DoS (Denial-of-Service), një agresor ndalon disponueshmërinë e shërbimit të një pajisje. Sulmet DoS ndaj smartphone-ve vij më shumë për konektivitet të fortë dhe reduktim të aftësive: për shkak të hardware-it të limituar, sulmimi i një smartphone-i mund të kryhet me pak

përpjekje; edhe pse trafiku i gjeneruar nga një agresor i vetëm mjafton që ta bëjë një pajisje të paqëndrueshme. DoS-e specifikë mund ta shkarkojnë shumë shpejt baterinë, ta fikin ose ose të limitojnë në mënyrë dramatike kohën e opërimit dhe përformimin e detyrave intensive të CPU-së që kërkojnë shumë energji ose e detyrojnë pajisjen të fiket. Një tjetër tip i DoS-it dërgon një sasi të madhe SMS/MMS te i njëjëti numër që ose ta bllokojë përdoruesin të kryej detyrën e tij ose të degradojë shërbimin e një zone. [248] tregon që duke përdorur vetëm komunikimiet SMS, psh. mesazhet që dërgohen ndërmjet smartphone-ve, telefonat e lirë mund të detyrohen që të fiken. Për këtë qëllim, protokolli i SMS-ve mund të përdoret për të dërguar programe të vegjël që hapen në smartphone. Opëratorët e rrjetit i përdorin këto file për të ndryshuar konfigurimet e pajisjes në largësi. E njëjta qasje shfrytëzohet për të sulmuar smartphone-t. Disa studime për parandalimin e kësaj klase sulmesh tregohen në [249, 211].

Shembuj: Në [250], autori ekzaminon mekanizmat e tanishëm të sigurisë në smartphone, duke identifikuar disa dobësi kritike të modeleve ekzistues të sigurisë. Për më tepër, ata tregojnë se si dobësi të tilla mund të shfrytëzohen për sulmet Distributed DoS në shërbimet e infrastrukturave publike duke devijuar thirrjet telefonike. Kjo arrihet duke injëktuar një kod të hartur shell përmes mbingarkimin e bufferave: në fakt, në shumë telefona me platformë Linux sistemi me një përdorues mund të aksesojë shumë lehtë privilegjet themelore duke thirrur ptrace() për të injëktuar kodin në procese të tjera. Autori demostron që, shfrytëzimi i 1% i sistemit mobile Linux, shërbimi i një qëndre shërbimi të emergjencave, në një zonë me miliona njërëz, mund të ndërpritet.

Sulmet e shkarkimit të baterisë kanë si shënjëstër një burim unik bottleneck në smartphone, e njohur si energjia e baterisë. [251] diskuton sulmet ndaj smartphone-ve që shkarkojnë baterinë e pajisjes deri në 22 herë më shpejt, duke e nxjerrë një pajisje të papërdorshme në një kohë të shkurtër. Për ta bërë këtë, si

fillim agresori duhet të ndërtojë një “hit-list” të të gjithë përdoruesve me rrjet Interneti aktiv duke shfrytëzuar avantazhin e protokollit të pasigurtë MMS, i cili automatikisht shkarkon mesazhe MMS duke marrë lajmërime përmes kërkesave të HTTP-së. Së dyti, agresori në mënyrë përiodike dërgon paketa UDP te smartphone-i i synuar dhe përdor ruajtjen e përmbajteve të PDP-së dhe paging channel. Autori tregon që nesë një telefon është i lidhur me Internet vazhdimisht, bateria do të harxhohet e gjitha në më pak se 7 orë.

Water torture attack [252] është një tjetër shembull i sulmeve të harxhimit të baterisë që bëhet nga shtresa PHY. Kjo arrihet duke e detyruar SS (subscriber station) të harxhojë baterinë, ose të konsumojë burimet e llogaritjeve, duke dërguar frama false.

Një shembull tjetër i sulmeve që synon disa smartphone Symbian S60 dhe i pengon viktimat të marrin mesazhet SMS njihet si Curse of Silence [253]. Ky sulm tenton të vendosë Messages Protocol Identifier te “Internet Electronic Mail” kështu që një SMS mund të përdoret për të dërguar e-mail. Ky sulm shfrytëzon dobësinë e disa smartphone-ve që nuk mund trajtojnë korrektësisht adresat e e-maileve me më shumë se 32 karaktere: duke shfrytëzuar këtë dobësi, agresori dërgon një e-mail fals në atë mënyrë që pajisja nuk është në gjendje të marrë më mesazhe të tjera SMS. Në fund të sulmit, smartphone-i shfaq një njoftim paralajmërues që memorja nuk është e mjaftueshme për të marrë mesazhe të tjera dhe disa të dhëna duhet të fshihen si fillim.

[254] prezanton një metodë të detajuar për analizimin e dobësive të implementuara nga SMS-të duke injëktuar mesazhe të shkurtra lokalisht brenda smartphone-it dhe analizimin dhe testimin e të gjithë shërbimeve të bazuara në SMS të implementuara nga stack-u i software-it të smartphone-it. Analizimi i dobësive kryhet duke bashkuar fusha të ndryshme në një mesazh standard SMS-je

duke përfshirë elemente siç janë adresa e dërguesit, të dhënat e user-it dhe flag-e të ndryshëm. Një test tjetër lidh mesazhe SMS të ndryshme në atë mënyrë që të detyrojë mesazhet të vijnë jashtë rendit ose dërgojë një payload të madh. Autori shpreh se përmes përdorimit të mjeteve testuese, ata ishin në gjendje të idendifikonin disa dobësi që mund të shfrytëzohen për të filluar sulmet DoS.

4. *Mbifaturimi*: Sulmet e mbifaturimit tarfiojnë pagesa shtesë te llogaria e viktimës dhe mund të transferojnë këtë pagesë shtesë nga viktima te llogaria e agresorit. Duke qënë se shumë nga shërbimet wireless rregullohen me kontrata pay-për-use, këto sulme janë shumë specifike te smartphone-t wireless.

   Shembull: Një karakterisitkë e rrjetiti GPRS është gjendja gjithmonë “on”: përdoruesit tarifohen për sasinë e trafikut në vend të kohës së përdorimit. Një shembull tipik i sulmeve të mbifaturimit në këtë rrjet është ai kur një agresor, në një korporatë me një server keqdashës të gjendur jashtë rrjetit GPRS, vjedh adresen IP të pajisjes së shënjëstruar dhe fillon sesionin e shkarkimeve në këtë server. Prandaj, user-i legjitim tarifohet për trafikun legjitim që nuk e ka kërkuar kurrë [255].

5.3 Karakterizimi i Malwareve

Në këtë paragraf, do të paraqës karakteristikat kryesore për viruset ekzistuese në sistemin operativ Android, siç tregohet në detaje tek [8], duke filluar nga instalimi, aktivizimi deri në transferimin e të dhënave keqbërëse.

5.3.1 Instalimi i malware-ve

Duke analizuar manualisht një model programi keqdashes në të dhënat që disponoj, unë kategorizoj mënyrat egzistuese që këto programe përdorin për tu instaluar në telefonat e përdoruesve dhe i përgjithësoj në tre teknika te bazuara ne inxhinieri sociale , p.sh: ripaketimi, sulm përditësimi (Update Attack), dhe infektimi gjatë shkarkimit (ose navigimit). Këto teknika nuk janë të njëjta kur vjen puna e infektimit

pasi variante të ndryshme të të njëjtit tip përdorin mënyra të ndryshme për të joshur përdoruesin për shkarkim [8].

5.3.1.1 Ripaketimi

Ripaketimi është një nga menyrat me te zakonshme që autorët e programeve keqdashese përdorin për te ngarkuar te dhëna te infektuara ne aplikacionet me popullore. Ne teresi ,keta autore pikasin dhe shkarkojne aplikacione popullore , i zberthejne ato , injëktojne te dhëna te keqija , me vone I bejne njësh me aplikacionin dhe I paraqësin ne Dyqanin Android zyrtar ose alternativat e tij. Përdoruesit mund te jene te cenueshme duke u terhequr për ti shkarkuar dhe ti instalojne keto aplikacione te infektuara. Për te përcaktuar sasine e përdorimit te teknikes se ripaketimit ndermjet koleksionit tone , ne ndjekim rrugen që vijon, n.q.s një model ndan te njëjtin emer pakete me një aplikacion ne dyqanin Android , atehere ne shkarkojme aplikacionin (nëse është falas) dhe manualisht krahasojme diferencen , e cila zakonisht përmban the dhëna te infektuara. Nëse aplikacioni original nuk është I disponueshem , ne zgjedhim te zberthejme modelin e infektuar dhe manualisht vendosim nëse pjesa e infektuar është pjese e funksionit kryesor te aplikacionit bartes. Nëse jo , ky program është konsideruar si i ripaketuar[8].
Ne total , ne mes te 1260 modelesh aplikacionesh keqdashese , 1083 prej tyre (86.0%) janë te ripaketuar. Duke I klasifikuar me tej bazuar mbi çdo lloj individualisht , hasim se Brenda 49 tipesh ne koleksionin tone , 25 prej tyre infektojne përdoruesit me aplikacionet e ripaketuara , nderkohe 25 prej tyre janë aplikacione me vete ku shumica e tyre janë te dizenjuara te spiunojne . Një tip infektimi p.sh : GoldDream , i përdor te dyja për te infektuar [8].

Midis 1083 aplikacioneve te ripaketuara , gjejme se autorët e infektimeve kane zgjedhur një shumellojshmeri aplikacionesh për te ripaketuar , përfshire aplikacione te paguara , lojerat me popullore , aplikacione te fuqishme për vegla (përfshire përditesimin e sigurise) , gjithashtu aplikacione referuar pornografise. Për shembull , një model infektues AnserverBot(SHA1: ef140ab1ad04bd9e52c8c5f2fb6440f3a9ebe8ea) ripaketoi

një aplikacion te paguar com.camelgames.mxmotor i disponueshem ne dyqanin zyrtar Android. Një tjeter infektues, BgServ[256] , ripaketoi një mjet sigurie leshuar nga Google për te hequr DroidDream nga telefonat e infektuar. Gjithashtu , mundesisht nga përpjekja për te fshehur te dhënat e infektuara , autorët janë te prirur te përdorin Emra Class-File te cilat duken te pranueshme dhe dashamirese. Për shembull , AnserverBot përdor një pakete te emeruar com.sec.android.provider.drm për ngarkimin e tij , I cili duket si një modul që siguron funksionin DRM te pranueshem. Versioni I parë I DroidKungFu zgjodhi te përdor com.google.ssearch për tu fshehur si një modul kerkimi Google dhe versionet pasardhese përdoren com.google.update për tu qasur si një përditesim zyrtar I Google. Është interesante për tu thene se një lloj infektuesi -jSMSHider –

Përdorte një çeles privat te disponueshem publikisht (serial number: b3998086d056cffa) i cili ishte shpërndare Android Open Source Projekt (AOSP Projekti i Burimeve te Hapura për Android). Modeli actual I sigurise për Android lejon aplikacionet shenjuar me te njëjtin celes platforme nga sistemi I telefonit , te kerkojne autorizim , i cili nuk është I lejueshem për aplikacione normale te paleve te treta. Një autorizim i tillë përfshin instalimin e aplikacioneve shtese pa nderhyrjen e përdoruesit. Fatkeqësisht, disa firmware te meparshem janë shenjuar nga një çelës default shpërndare ne AOSP. Kjo rezultoi që aplikacionet e infektuara jSMSHider mund te siguronin autorizime te privilegjuara për te ekzekutuar veprime pa dijeninë e përdoruesit [8].

Tabela 3 Malwaret e SO Android të ndara sipas Instalimit dhe Aktivizimit

	Installation				Activation								
	Repackaging	Update	Drive-by Download	Standalone	BOOT	SMS	NET	CALL	USB	PKG	BATT	SYS	MAIN
ADRD	√				√		√	√					
AnserverBot	√	√			√	√	√		√		√	√	
Asroot				√									
BaseBridge	√	√			√	√	√				√	√	
BeanBot	√					√		√					
BgServ	√				√	√							√
CoinPirate	√				√	√							
Crusewin				√	√	√							
DogWars	√												
DroidCoupon	√				√		√	√		√			
DroidDeluxe				√									
DroidDream	√												√
DroidDreamLight	√				√			√					
DroidKungFu1	√				√						√	√	
DroidKungFu2	√				√						√	√	
DroidKungFu3	√				√						√	√	
DroidKungFu4	√				√						√	√	
DroidKungFuSapp	√				√						√	√	
DroidKungFuUpdate	√	√											
Endofday	√				√	√							
FakeNetflix				√									
FakePlayer				√									
GamblerSMS				√	√								
Geinimi	√				√	√							
GGTracker			√	√	√	√					√		
GingerMaster	√				√								
GoldDream	√			√	√	√		√					
Gone60				√									
GPSSMSSpy				√		√							
HippoSMS	√				√	√							√
Jifake	√		√										
jSMSHider	√									√			√
KMin				√	√								
Lovetrap				√	√	√							
NickyBot				√	√	√							
Nickyspy				√	√								
Pjapps	√				√	√						√	
Plankton		√		√									
RogueLemon				√		√							
RogueSPPush				√		√							
SMSReplicator				√		√							
SndApps				√	√								
Spitmo			√	√		√		√					
TapSnake				√	√								
Walkinwat				√									
YZHC				√	√								
zHash				√	√								
Zitmo			√	√		√							
Zsone	√					√							√
number of families	*25*	*4*	*4*	*25*	*29*	*21*	*4*	*6*	*1*	*2*	*8*	*8*	*5*
number of samples	*1083*	*85*	*4*	*177*	*1050*	*398*	*288*	*112*	*187*	*17*	*725*	*782*	*56*

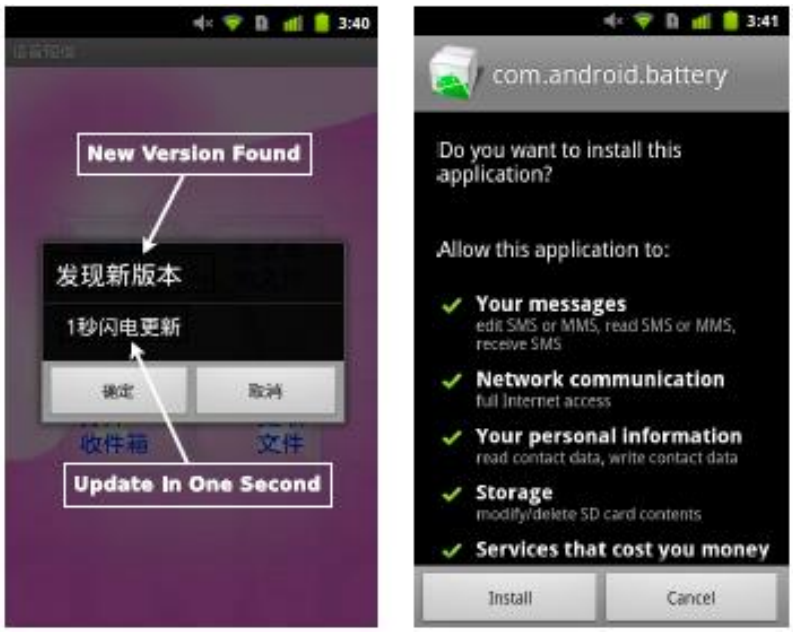
5.3.1.2 Sulmet gjatë update

Teknika e parë zakonisht parazitonte te dhënat e infektuara ne aplikacione bartese, e cila mund te zbulonte potencialisht pranine e tyre. Teknika e dyte e ben me te veshtire zbulimin. Specifikisht , ajo mund te ripaketoj përseri aplikacionet me te përdorura. Ne

vend që te përmbylle te dhënat e infektuara si një e tere , ajo përfshin një përditesim për një element I cili do te shkarkoje infektimin ne momentin e ekzekutimit. Si rezultat , një skanim I aplikacioneve bartese mund te deshtoje te dalloje te dhënat e infektuara. Ne te dhënat tona , janë kater klasifikime te medha programesh keqdashese : BaseBridge, DroidKungFuUpdate, AnserverBot, Plankton. Te cilët përdorin sulm përditesimi. BaseBridge ka një numer variantesh. Ndersa disa shfrytezojne rrenjët e programit I cili lejon instalimin e heshtur te aplikacioneve shtese pa nderhyrjen e përdoruesit, ne fokusohemi ne variantet e tjera që përdorin sulm përditesimi pa shfrytezimin e burimit te programit. Kur një aplikacion i infektuar BaseBridge ekzekutohet , ajo do te kontrolloje nëse një informacion përditesues do te shfaqët. Nëse po , duke thene se një version I ri është I disponueshem , përdoruesit do ti ofrohet te instaloje versionin e përditesuar. Versioni I ri aktualisht ruhet ne aplikacionin bartes sin je reserve. Nëse përdoruesi pranon, një version I "përditesuar" me te dhënat e infektuara do te instalohet. Kjo është një menyre me e fshehte se e para, për arsyen se t dhënat e infektuara janë ne versionin e përditesuar , jo ne aplikacionin original [8].

DroidKungFuUpdate është e ngjashme me BaseBridge , por ne vend që te përcjelle ose te mbylle versionin e "përditesuar" Brenda aplikacionit zyrtar, ai zgjedh që te shkarkoje ne distance nga rrjeti. Për me shume , ai zgjedh një rruge te fshehte duke njoftuar përdoruesin nga një library e pales se tretë [257] , e cila lejon funksionin e njoftimit. (funksioni është I njëjte me njoftimin automatik te Google Cloud ne strukturen e Mesazheve te pajisjes.) Pasi shkarkohet ,version I " përditesuar" del te jete infektuesi DroidKungFu , I cili është I aksesueshem ne dyqanin zyrtar dhe alternative te Android.

Dy sulmet e përditesimit te meparshme kerkonin aprovimin e përdoruesit për te shkarkuar dhe instaluar versionet e reja. Dy llojet e infektuesve te tjere , AnserverBot dhe Plankton , e çoi më tej sulmin përditesues duke përditesuar component te vecante ne aplikacionin bartes, jot e gjithe aplikacionin. Si rezultat , ai nuk kerkon lejen e përdoruesit. Ne vecanti , Plankton përveteson dhe ekzekuton një skede jar mbajtur ne një server ne distance ndersa AnserverBot gjen një hyrje publike blog, e cila mban te dhëna te infektuara për përditesimin [8].



a) Kërkesa e Update **b) Instalimi i një versioni të ri**

Fig. 2 Një sulm update nga BaseBridge

5.3.1.3 Infektimi gjatë shkarkimit

Mënyra e tretë aplikon sulmin tradicional gjatë shkarkimit në hapesiren mobile. Edhe pse ato nuk shfrytezojne direkt cenueshmerite e shfletuesit mobile , ato joshin përdoruesit te shkarkojne aplikacione interesante dhe te pasur me shume funksione. Ne te dhënat tona , kemi hasur kater tipe infektimesh si: GGTracker [258] , Jifake[259], Spitmo [260] , Zitmo[261]. Dy te fundit janë dizenjuar te vjedhin te dhënat bankare [8].

GGTracker fillon nga reklamimet Brenda aplikacionit. Vecanerisht , kur një përdorues klikon një adrese special reklame , ajo do e drejtoje përdoruesin ne një faqë keqdashese ,e cila sillet sikur analizon përdorimin e baterise se telefonit dhe e drejton përdoruesin ne

një dyqan Android jot e vertete për te shkarkuar një aplikacion që pretendon te përmiresoje efiçensën e baterise.

Fig. 3 Një sulm Update nga DroidKungfu

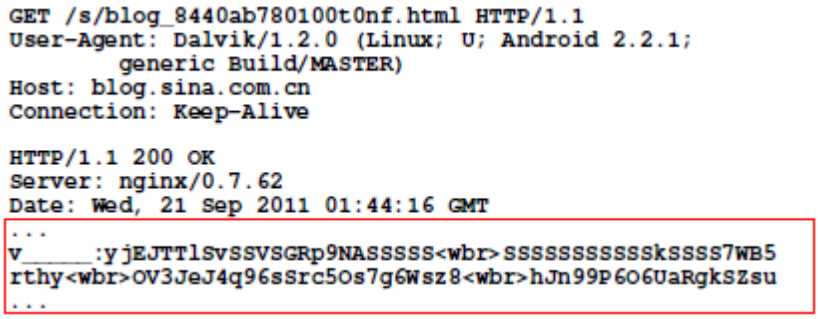


Fig. 4 Një sulm update nga AnserverBot

Fatkeqësisht , aplikacioni i shkarkuar nuk fokusohet ne përmiresimin e baterise , por një infektim I cili do te pajtohet ne një sherbim tarifor te larte pa dijenine e përdoruesit.

Jifake shkarkohet po njësoj ne një faqë dashakeqëse. Sidoqofte , ai nuk përdor reklamimin brenda aplikacionit për te terhequr dhe drejtuar përdoruesit. Ne te kundert , ai përdor skanimin e infektuar te kodit QR [262] , I cili kur skanohet drejton përdoruesin ne faqën e infektuar , e cila përmban Jifake. Ky infektim është ripaketimi I ICQ , I cili përcon SMS te ndryshme me tarifa te larta. Edhe pse shumimi I infektimit bazuar ne kodin QR është paralajmeruar me heret [263], ky është sulmi I parë që ndodh [8].

Dy te fundit , Spitmo dhe Zitmo janë versione te sjella nga infektues kompjuterash si : SpyEye dhe Zeus. Ato punojne ne te njëjten menyre , kur një përdorues është duke bere

veprime bankare ne rrjet me një kompjuter te kompromentuar , përdoruesi do te drejtohet për te shkarkuar një aplikacion mobil te vecante , I cili pretendon se mbron me mire aktivitetin banker ne rrjet. Gjithsesi , aplikacioni I shkarkuar është një infektues, I cili mund te dergoje mTANs ose SMS tek një server ne distance. Keto dy tipe infektuesish bazohen ne një shfletues kompjuteri për te lancuar sulmin . Edhe pse duket e veshtire te infektoje përdorues reale , fakti që ata mund te vjedhin te dhëna bankare ngre një alarm serioz për përdoruesit [8].

5.3.1.4 Të tjerë

Deri me tani kemi paraqitur 3 menyra kryesore bazuar ne teknika inxhinierike sociale përdorur ne infektimin e Android. Me tej , ne provojme menyrat e tjera te mbetura te cilat nuk karakterizohen ne 3 te mesipërmet. Ne vecanti , te dhënat tona kane 1083 aplikacione te ripaketuara, te cilat lene 177 aplikacione me vete. Ne shohim keto aplikacione te mbetura me vete dhe I organizojme ne 4 grupe që vijojne me poshte [8].

Tabela 4 Ngjarjet në Android

Abbreviation	Events	Abbreviation	Events	Abbreviation	Events
BOOT (Boot Completed)	BOOT_COMPLETED	SMS (SMS/MMS)	SMS_RECEIVED WAP_PUSH_RECEIVED	NET (Network)	CONNECTIVITY_CHANGE PICK_WIFI_WORK
CALL (Phone Events)	PHONE_STATE NEW_OUTGOING_CALL	USB (USB Storage)	UMS_CONNECTED UMS_DISCONNECTED	MAIN (Main Activity)	ACTION_MAIN
PKG (Package)	PACKAGE_ADDED PACKAGE_REMOVED PACKAGE_CHANGED PACKAGE_REPLACED PACKAGE_RESTARTED PACKAGE_INSTALL	BATT (Power/Battery)	ACTION_POWER_CONNECTED ACTION_POWER_DISCONNECTED BATTERY_LOW BATTERY_OKAY BATTERY_CHANGED_ACTION	SYS (System Events)	USER_PRESENT INPUT_METHOD_CHANGED SIG_STR SIM_FULL

Grupi i parë janë te vetequajturat Spyware – ato kane për qëllim te instalohen ne telefonin e viktimes. Kjo ndoshta shpjegon pse sulmuesit nuk kane motivim ose nevojen te joshin përdoruesin për instalimin e tyre. GPSSMSpy është një shembull I cili degjon komandat e bazuara tek SMS për te rregjistruar dhe ngarkoje vendndodhjen e përdoruesit.

Grupi I dyte përfshin ato aplikacionet e rreme , te cilat maskohen si aplikacionet zyrtare por fshehurazi ekzekutojne veprime keqdashese, sit e vjedhin te dhënat e përdoruesit ose te dergojne SMS pa dijenine e tyre. FakeNetflix është një shembull I cili vjedh te dhënat

dhe fjalekalimin e përdoruesit te aplikacionit Netflix. Kjo nuk është një ripaketim I Netflix , por një aplikacion I maskuar si I tille , me te njëjten nderfaqë përdoruesi. FakePlayer është një tjeter shembull sesi maskohet si një player filmi ,por nuk mundeson as funksionin me te vogel te reklamuar. E gjitha cfare ai ben është te dergoje SMS me tarifa te larta drejt një numri pa dijenine e përdoruesit.

Grupimi I tretë përmban aplikacione te cilat me qëllim përfshijne funksione për te infektuar si psh dergimi I paautorizuar I SMS –ve ose pajtimi automatik tek disa sherbime me vlere te shtuar. Por diferenca nga grupi I dyte është se ato nuk janë te rreme. Ne te kundert , ato përformojne funksionet që thone. Por pa dijenine e përdorueseve , ato përfshijne dhe funksione keqdashese. Për shembull , RogueSPPush është një aplikacion astrologjik , por ai automatikisht pajtohet tek sherbime me një tarife te larte duke fshehur qëllimisht SMS konfirmues.

Grupi I fundit përfshin ato aplikacione që mbështëten tek privilegjet rrenjësore për te funksionuar Gjithsesi , pa pyetur përdoruesin për te aksesuar rrenjësisht ato aplikacione, ato shfrytezojne hapesira te ditura me parë për te arritur rrenjën. Edhe pse keto aplikacione jo qartesisht demostrojne qëllime keqdashese , fakti se arrijne deri ne burimin e aplikacionit , është alarmant. Shembujt ne kete grup janë Asroot dhe DroidDeluxe[8].

5.3.2 Aktivizimi

Këtu ekazaminoj ngjarjet gjatë aktivizimit të sistemin Android. Nga rregjistrimi i ngjarjeve të sistemit, një malware Android mbështetet në suportin e njoftimeve të automatizuara dhe rithirrjet në Android për të nxitur në mënyrë fleksible lancimin e payload-it të tij. Për thjesthësi, i shkurtoj disa ngjarje në Android të përdoruara në Tabelën III. Për secilën familje të malware-ve në bashkësinë e të dhënave tona, unë gjithashtu i paraqës ngjarjet e lidhura në Tabelën II [8].

Ndër të gjitha ngjarjet e sistemit të mundshme, BOOT_COMPLETED është më interesantja për gjetjen e malwareve Android. Kjo nuk është surpizuese kur sistemi

përfundon procesin e booting – koha përfekte për malware-in për të nisur shërbimet e background-it. Në bashkësinë e të dhënave tona, 29 (me 83.3% e kampjoneve) familje malware-sh degjojnë për këtë ngjarje. Për shembull, Geinimi (SHA1: 179e1c69ceaf2a98fdca1817a3f3f1fa28236b13) dëgjon për këtë ngjarje për të bootstrap shërbimet background – com.genimi.AdService.

The SMS_RECEIVED renditet i dyti me 21 familje malware-sh të lidhura me të. Kjo është gjithashtu e arsyeshme si shumë malware do të jenë të prirur në ndërprerjen ose përgjigjen e mesazheve SMS hyrëse. Si shembull, zSone dëgjon për këtë ngjarje SMS_RECEIVED dhe ndërpret ose heq të gjitha mesazhet SMS nga numri i origjinës përkatëse si "10086" dhe "10010." [8]

Gjatë analizës tonë, ne gjithashtu gjejmë se malware të caktuara rregjistrojnë për shumëllojshmëri ngjarjesh. Për shembull, AnserverBot rregjistron për callback nga 10 ngjarje të ndryshme ndërsa BaseBridge në 9 ngjarje të ndryshme. Ky numër i madh rregjistrimesh i ngjarjeve pritet ti lejojë malware-it në mënyrë të besueshme ose me shpjetësi të lancojë payloadin e mbajtur.

Vec kësaj, gjithashtu vëzhgova disa kampjone malware-sh direkt duke vjedhur entry-it e aktiviteteve të aplikacioneve host, të cilët do të nxiten kur përdoruesi klikon ikonën e aplikacionit në home screen ose një veprim i marrë qëllimisht me ACTION_MAIN. Vjedhja e entry-ve të aktivitetit i lejon malware-it që menjëherë të nxjerrë shërbimet e tij përpara fillimit të aktiviteteve primare të programit host. Për shembull, DroidDream (SHA1:fdf6509b4911485b3f4783a72fde5c27aa9548c7) zëvendoëson entry-n origjinal të aktiviteteve me atë të sajën com.android.root.main kështu që mund të marrë kontrill përpara lancimit të aktivitetit origjinal com.codingcaveman.SoloTrial.SplashActivity. Disa malware gjithashtu mund të vjedhin ngjarjet e ndërveprimeve specifike UI (psh. klikimi i butonit). Një shembull është malware-I zSone (SHA1:00d6e661f90663eeffc10f64441b17079ea6f819) që nxit dërgimin e kodit të tij SMS brenda funksionit onClick() në aplikacionin host [8].

Tabela 5 Lista e Malware-ve të cilët kërkojnë dhe përdorin root-in e SO Android

Vulnerable Program	Root Exploit	Release Date	Malware with the Exploit
Linux kernel	Asroot [23]	2009/08/16	Asroot
init (<= 2.2)	Exploid [24]	2010/07/15	DroidDream, zHash DroidKungFu[1235]
adbd (<= 2.2.1) zygote(<= 2.2.1)	RATC [25] Zimperlich [26]	2010/08/21 2011/02/24	DroidDream, BaseBridge DroidKungFu[1235] DroidDeluxe DroidCoupon
ashmem (<= 2.2.1)	KillingInThe NameOf [27]	2011/01/06	-
vold (<= 2.3.3)	GingerBreak [28]	2011/04/21	GingerMaster
libsysutils (<= 2.3.6)	zergRush [29]	2011/10/10	-

5.3.3 Kod me qëllim të keq

Malware egzistues mund te karakterizohen nga payloads te mbartura , gjithashtu survejuar ne te dhënat tona , i ndajme funksionet e tyre ne kater kategori te ndryshme , tejkalim privilegjesh (Privilege Escalation) , kontrollim ne distance (Remote Control), ngarkim financiar (Financial Charge) dhe vjedhje informacionesh private (Përsonal Information Stealing) [8].

1) **Tejkalim Privilegji (Privilege Escalation)**

Platforma Android është një sistem I përbere , I cili konsiston jo vetem ne Linux kernel , por edhe ne gjithe strukturen Android ne me shume se 90 librari open-source , përfshire WebKit , SQLite dhe OpenSSL. Natyrisht kjo përberje paraqët ane te cenueshme te sistemit , te cilat potencialisht shfrytezohen për tejkalim privilegji(Privilege Esclation)
Mbi te gjitha , janë një numer I vogel I cenueshmerive(exploit) ne nivel platforme , te cilat përdoren gjeresisht. Shfrytezuesit kryesore janë exploid, RATC(RACE AGAINST THE CAGE) dhe Zimpërlich. Ne vecojme se nëse RATC exploit është ne një aplikacion aktiv , ai është duke shfrytezuar efektivisht anomaline ne zygote daemon , jo adbd

daemon , që ka për qëllim, gjithashtu sjelle si Zimplerlich exploit. Marre ne konsiderate sjelljen e përbashket te te dyve , ne përdorim RATC për ti përshkruar te dy.

Nga analiza e bere , një rezultat alarmant shfaq se midis 1260 modelesh ne te dhënat tona , 463 prej tyre (36,7%) mbjellin te pakten një rrenjë shfrytezuese. Ne term ate popullaritetit te secilit shfrytezues janë 389,440,4 dhe 8 modele te cilët përmbajne exploid, RATC , GingerBreak dhe asroot , respektivisht [8].

Gjithashtu nuk është e pazakonte për një malware te kete dy ose me shume rrenjë shfrytezuese (root exploit) për te shumefishuar mundesite për shfrytezim sa me te suksesshem ne versione te shumta te platformes. Ne te dhënat tona , janë 378 modele me me shume se një rrenjë shfrytezuese.

Ne një investigim te metejshem se si keto shfrytezues përdoren , tregon se si shume me parë malware kopjojne tekstualisht rrenjët shfrytezuese pa ndonjë modifikim, edhe pa levizur vargun e nxjerre te rregulluar ose te ndryshojne emrat e dosjeve te përfshira ne rrenjët shfrytezuese. Përshembull DroidDream përmban te njëjtin emer për exploid , njësoj si ai I aksesueshem publikisht. Gjithsesi , kohet e fundit kane ndodhur ndryshime. Përshembull , DroidKungFu nuk I mbjell direkt keto rrenjë shfrytezuese. Ne te kundert , ai ne fillim enkripton keto rrenjë dhe me pas I ruan si një dosje aset. Kur përdoret , ai I shfaq keto rrenjë shfrytezuese te enkriptuara dhe I ekzekuton me rradhe , e cila e ben zbulimin e tyre shume te veshtire. Ne versionin e parë te DroidKungFu , u raportua se çdo antivirus mobile I asaj kohe nuk mund ta zbulonte , gje që demostron efektivitetin e tij. Për me shume , malware te tjere si DroidCoupon and gingerMaster tjetersojne emrin e dosjes psh duke pretenduar se është një fotografi me prapashtesen png. Ne besojme se kto ndryshime reflektojne natyren e evolimit te zhvillimit te malware dhe luftes se vazhdueshme te mbrojtjes kunder tyre [8].

2) Kontroll në Distance (Remote Control)

Gjatë analizes time për te ekzaminuar funksionin e kontrollit ne distance ndermjet malware payloads, jemi te surprizuar te njoftojme se 1.172 modele (93%) I kthejne telefonat e infektuar ne përdorimin e tyre te kontrolluar ne distance. Specifikisht, janë

1,171 lloje te cilat përdorin trafikun e rrjetit HTTP për te marre komanda nga serverat e tyre C&C. Gjithashtu disa lloje malware mundohen te jene te fshehte duke enkriptuar URL e serverit C&C gjitashtu si dhe komunikimin e tyre me serverin. Përshembull Pjappa përdor skemen e tij te kodimit për te enkriptuar adresen e serverit C&C. Një nga shembujt e tij (SH1: 663e8eb52c7b4a14e2873b1551748587018661b3) kodon serverin e tij C&C mobilemeego91.com në 2maodb3ialke8mdeme3gkos9g1icaofm. DroidKungFu3 vepron me standardin AES të enkriptimit dhe përdor celësin Fuck_sExy-aLl!Që për të fshehur C&C e tij. Genimi në mënyrë të ngjashme aplikon skemën e enkriptimit DES (me celësin 0x0102030405060708) për të enkriptuar komunikimin e tij në distancë me serverin C&C.

Gjatë studimit tonë, ne gjithashtu gjejmë që shumica e serverave C&C janë të rregjistruar në domain-et e tyre të kontrolluara nga vetë agresorët. Mirëpo, ne gjithashtu dallojmë raste ku serverat C&C mbahen në cloude public. Për shembull, spyëare-I Plankton në mënyrë dinamike ngarkon dhe vë në punë payload-in e tij nga një server i Amazon. Kohët e fundit, agresorët janë duke u kthyer edhe nga serverat e blogjeve se serverat e tyre C&C. AnserverBot është një shembull që përdor dy shërbime të blogjeve të famshme, si Sina dhe Baidu, si serverat e tyre C&C për të marrë payload-in dhe URL të re për C&C [8].

3) Tarifim Shtesë

Përvec tejkalimit të privilegjeve dhe kontrollit në distance, ne gjithashtu shikojmë në motivimet përtej infektimit të malware-it. Në vecanti, ne studjojmë nëse malware me qëllim do të shkaktojë tarifim financjar te përdoruesi i infektuar.

Një mënyrë përfituese për agresorët është të bëhen subscribe në mënyrë të fshehtë në shërbimet premium-rate, sic janë dërgimi i mesazheve SMS. Në Android, është një funksion përmission-guarded sendTextMessage që të lejon dërgimin e mesazheve SMS në background pa vetëdijen e përdoruesit. Ne jemi në gjendje ta konfirmojnë këtë tip sulmi që kishte si qëllim përdoruesit në Rusi, SHBA, dhe Kinë. Malware-I I parë Android FakePlayer dërgon mesazhe SMs “7987657” te shumë numra premium në Rusi.

GGTracker automatikisht rregjistron përdoruesin e infektuar në shërbimin premium në US pa dijeninë e përdoruesit. zSone dërgon mesazhe SMS në numrat premium në Kinë pa dijeninë e përdoruesit. Në total, janë 55 kampjone (4.4%) të ndara në 7 familje të ndryshme (të shënuara me ‡ në Table V) që dërgojnë mesazhe SMS në numrat premium që vendosura në aplikacionet e infektuara.

Për më tepër, disa malware zgjedhin të mos e bashkangjisin numrin premium në aplikacion. Në vend të kësaj, ata shfrytëzojnë fleksibilitetin e kontrollit në distancë për të marrë numrin në kohën e ekzekutimit. Në të dhënat tona, janë 13 familje të këtyre malwareve (të shënuara me† në Tabelën V). Në dukje, familjet e këtyre malware-ve janë më të qëndrueshëm se ato të parët sepse numri destinacion nuk mund të njihet vetëm nga analizimi i aplikacionit të infektuar.

Në analizën time, vëzhgoj në mënyrë automatike duke u bërë pjese e shërbimeve premium, këto familje malwaresh kanë nevojë tju përgjigjen disa mesazheve SMS specifikë. Kjo mbase për një politik e konfirmimit së dyti që kërkohet në disa shtete si në Kinë. Specifikisht, për tu rregjistruar në një shërbim premium, përdoruesi duhet ti përgjigjet një SMS konfirmuese dërguar nga ofruesi I shërbimit për të finalizuar ose aktivizuar shërbimin. Për të shmangur njoftimin e përdoruesit, ata kujdesen që t`iu rikthejnë përgjigje vetë këtyre mesazheve konfirmuese. Si shembull, RogueSPPush automatikishit përgjigjet me “Y” drejt këtyre mesazheve në background; GGTracker përgjigjet me “YES” te një numër premium, 99735, për të aktivizuar shërbimin. Në mënyrë të ngjashme, për të shmangur njoftimin e përdoruesve nga mesazhet e tarifimit, ata zgjedhin ti filtrojnë këto mesazhe. Ky veprim vërehet në disa malware, përshfirë edhe zSone, RougeSPPush, dhe GGTracker. [8]

Tabela 6 Klasifikimi i malware-ve

	Privilege Escalation					Remote Control		Financial Charges			Personal Information Stealing		
	Exploid	RATC/ Zimperlich	Ginger Break	Asroot	Encrypted	NET	SMS	Phone Call	SMS	Block SMS	SMS	Phone Number	User Account
ADRD						√							
AnserverBot						√			√ †				
Asroot				√									
BaseBridge		√				√		√	√ †	√			
BeanBot						√		√	√ †	√		√	
BgServ						√			√ †	√		√	
CoinPirate						√			√ †	√	√		
Crusewin						√			√	√	√		
DogWars									√				
DroidCoupon		√				√							
DroidDeluxe		√											
DroidDream	√	√				√							
DroidDreamLight						√							√
DroidKungFu1	√	√			√	√						√	
DroidKungFu2	√	√			√	√						√	
DroidKungFu3	√	√			√	√						√	
DroidKungFu4						√							
DroidKungFu5	√	√			√	√						√	
DroidKungFuUpdate													
Endofday						√			√			√	
FakeNetflix													√
FakePlayer									√ ‡				
GamblerSMS											√		
Geinimi						√		√	√ †	√	√	√	
GGTracker									√ ‡	√	√	√	
GingerMaster			√			√						√	
GoldDream						√		√	√ †		√	√	
Gone60											√		
GPSSMSSpy									√				
HippoSMS									√ ‡	√			
Jifake									√ ‡				
jSMSHider						√			√ †	√		√	
KMin						√			√ †	√			
Lovetrap									√ †	√			
NickyBot							√		√		√		
Nickyspy						√			√		√		
Pjapps						√			√ †	√		√	
Plankton						√							
RogueLemon						√			√ †	√	√		
RogueSPPush									√ ‡	√			
SMSReplicator									√		√		
SndApps													√
Spitmo						√			√ †	√	√	√	
TapSnake													
Walkinwat									√				
YZHC						√			√ ‡	√		√	
zHash	√												
Zitmo											√		
Zsone									√ ‡	√			
number of families	*6*	*8*	*1*	*1*	*4*	*27*	*1*	*4*	*28*	*17*	*13*	*15*	*3*
number of samples	*389*	*440*	*4*	*8*	*363*	*1171*	*1*	*246*	*571*	*315*	*138*	*563*	*43*

Pavarësisht këtyre numrave premium, disa malware gjithashtu shfrytëzojnë të njëjtin funksionalitet duke dërguar mesazhe SMS drejt numrave të tjerë. Megjithëse më pak serioz se ajo drejt numrave premium, ato akoma do të sjellin tarifime të caktuara financiare vecanërisht kur përdoruesi nuk ka një numër të pakufizuar dërgimi mesazhesh. Për shembull, DogËars dërgon mesazhe SMS te të gjithë kontaktet në telefon pa dijeninë e përdoruesit. Disa malware gjithshtu mund të bëjne thirrje telefonike në background. Me të njëjtën aftësi të kontrollit në distancë, numri i destinacionit mund të mundësohet nga një server C&C në distancë, sic tregohet edhe te Geinimi.

4) Information Collection

Përpos payload-it, gjithashtu shikoj se malware-t vazhdimisht vjedhin informacione në telefonat e infektuar, përfshirë mesazhet SMS, numrat telefonikë si edhe user account-et. Në vecanti, janë 13 familje malware-sh (138 kampjone) në të dhënat tona që mbledhin mesazhet SMS, 15 familje (563 kampjone) që mbledhin numrat telefonikë, dhe 3 familje (43 kampjone) që marrin dhe ngarkojnë informacione rreth user account-et. Për shembull, SndApps mbledh adresat e email-it të përdoruesit dhe i dërgon te një server në distancë. FakeNetflix mbledh llogaritë e Netflix të përdoruesve dhe password-et duke mundësuar një Netflix identik UI [8].

Konsiderohet mbledhja e mesazheve SMS të përdoruesve si një sjellje tëpër dyshuese. Kredencialet e përdoruesit mund të përfshihen në mesazhet SMS. Për shembull, të dyja si Zitmo (version Zeus në Android) dhe Spitmo (version SpyEpy në Android) tentojnë të ndërpresin mesazhet e verifikimit SMS dhe më pas ti ngarkojnë ato te një server në distancë. Në qoftë se me sukses, agresori mund ti përdorë ato të gjenerojë vetë transaksione mashtruese te përdoruesi i infektuar [8].

5.3.4 Përdorimi i lejeve

Për aplikacionet Android pa shfrytëzuar rooting, aftësitë e tyre janë rigorozisht të kufizuara nga autorizimi i të drejtave të përdoruesit për to. Gjithashtu, do të ishte me interes për tu krahasuar kërkuesit kryesorë për autorizimin e të drejtave nga këto aplikacione te dëmshme në bashkësinë e të dhënave nga më pak i rrezikshmi deri te më i rrezikshmi. Si përfundim, në mënyrë rastësore kemi zgjedhur 1260 aplikacionet më të mira pa pagesë të shkarkuara nga marketi zyrtar i Android.Rezultatet tregohen në figurën 5 dhe 6 [8].

Bazuar në krahasimet, INTERNET, READ_PHONE_STATE, ACCESS_NETWORK_STATE, dhe WRITE_EXTERNAL_STORAGE janë shumë të përhapura qoftë për aplikacionet malicious qoftë edhe ata të parrezikshëm. Dy të parët nevojiten për të lejuar libraritë embedded që të funksionojnë sic duhet. Por aplikacionet malicious tentojnë të kërkojnë vazhdimisht leje lidhur me SMS-të, si READ_SMS, ËRITE_SMS, RECEIVE_SMS, dhe SEND_SMS. Specifikisht, janë 790 kampjone (62.7%) në të dhënat tona që kërkojnë leje READ_SMS, ndërsa 33% e aplikacioneve të parrezikshëm (ose 2.6%) e kërkojnë këtë leje. Këto rezultate janë në përputhje me faktin që 28 familje malwaresh në të dhënat tona (ose 45.3% e kampjoneve) që janë të lidhura me fuksionalitetet malicious te SMS.

Gjithashtu, vëzhgova 688 kampjone malware-sh që kërkojnë lejen RECEIVE_BOOT_COMPLETED.

Ky numër është 5 herë më i madh se na aplikacionet e parrezikshëm (137 kampjone). Kjo mund të vijë nga fakti që malwari është më i prirur të ekzekutohet në background pa ndërhyrjet e përdoruesit. Po asht janë 398 kampjone malware-sh që kërkojnë lejen CHANGE_WIFI_STATE, i cili është i një rendi shumë të lartë kërkese se sa në aplikacionet e parrezikshme (34 kampjone). Kjo kryesisht sepse Exploid root kërkon evente hot plug si ndryshimin e gjendjes së WIFI, i cili lidhet me këto leje [8].

Së fundmi, vihet re se aplikacionet me qëllime të këqia tentojnë të kërkojnë më shumë leje se sa ata të padëmshëm. Në bashkësinë e të dhënave tona, numri mesatar i lejeve të kërkuara nga aplikacionet malicious është 11 ndërsa për aplikacionet e parrezikshëm

është 4. Ndër 20 lejet kryesore, mesatarisht 9 kërkohen nga aplikacione të dëmshme ndërsa 3 prej tyre kërkohen nga aplikacione të padëmshme [8].

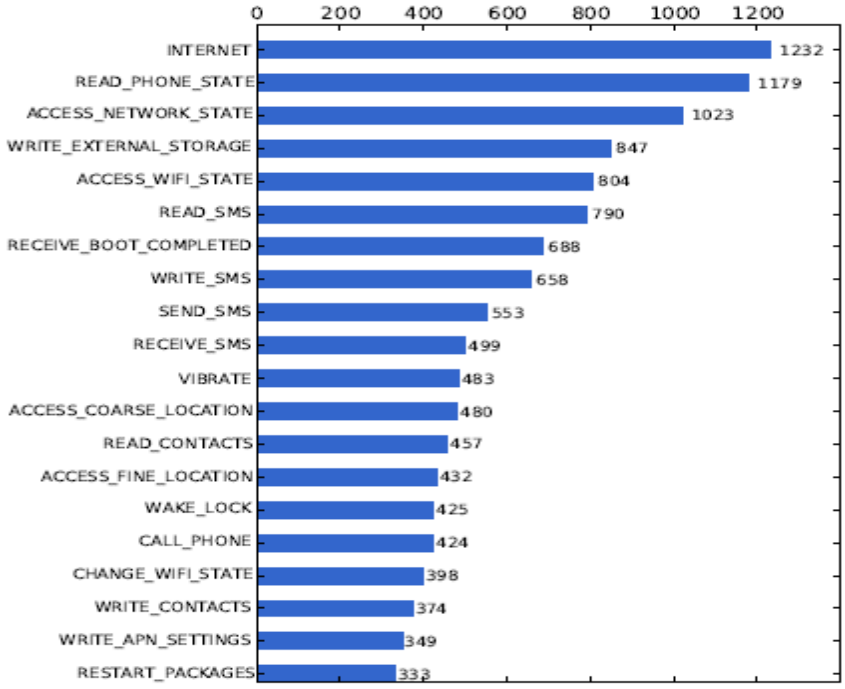


Fig. 5 Lejet që kërkohen nga 1260 Malware

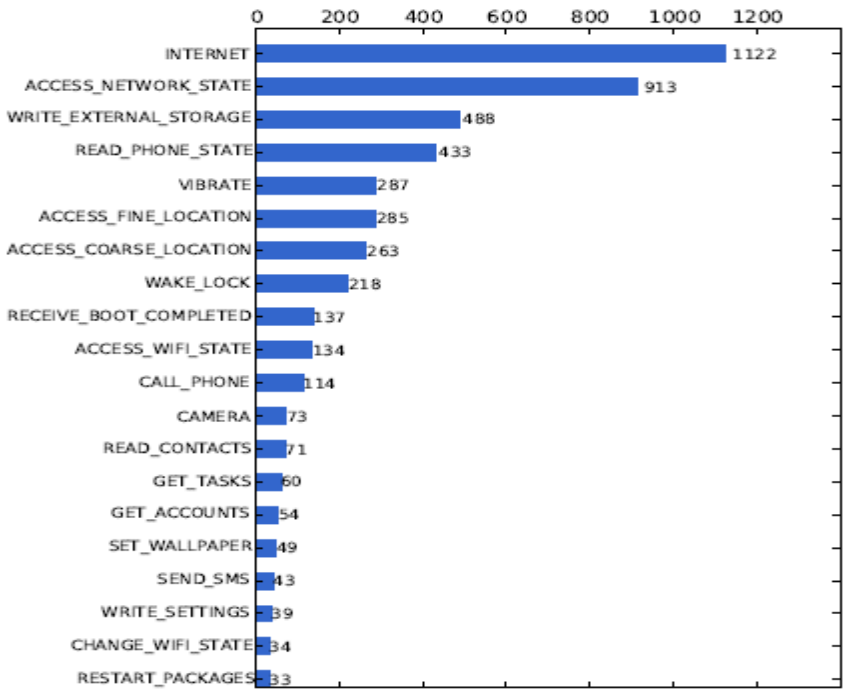


Fig. 6 Lejet që kërkohen nga 1260 aplikacione normale

KAPITULLI 6

Mbrojtja e SO Android

Në këtë seksion do të vëzhgojmë mekanizmat ekzistues të cilët janë zhvilluar për të shmangur lloje të ndryshme të kërcënimeve ndaj smartphone-ve dhe që janë përmbledhur të gjitha tek [7]. Fillimisht do të merremi me paraqitjen e sistemeve për detektim të ndërhyrjeve (Intrusion Detection System – IDS) dhe më pas me zgjidhjet e besueshme të bazuara mbi telefonin (Trusted mobile-based). Gjithë zgjidhjet do të jenë të organizuara në rend kronologjik.

6.1 Sistemet për detektim të ndërhyrjeve (IDS)

Në këtë paragraf do të paraqësim modelet dhe mjetet të cilat implementojnë sistemet me detektim të ndërhyrjeve (IDS) në smartphone.
IDS mund të bazohen në dy vëzhgime komplementare të njëra-tjetrës [7]:

1) Të bazuara mbi parashikimin – duke përdorur algoritmat e kriptografisë, nënshkrimin dixhital (digital signature), funksionet 'hash', vecori të rëndësishme të cilat kanë të bëjnë me konfidencialitetin, autenticitetin apo integritetin që mund të sigurohet; sipas këtij skenari, 'IDS' duhet të jenë 'online' dhe në kohe reale (real-time).
2) Të bazuara mbi detektim – IDS shërbejnë si linjë e parë e mbrojtjes duke identifikuar me efikasitet aktivitete keqdashëse.

Për më tepër kemi dy tipe kryesore të detektimit [7]:

1) Të bazuara mbi anomalite (Anomaly-based) (emër alternativ – të bazuara mbi sjelljen) – një krahasim i sjelljes "normale" me atë "reale"

2) Të bazuara mbi nënshkrimin (Signature based) (emër alternativ – detektim i nënshkrimit, detektim mbi njohuritë, detektim nëpërmjet paraqitjes) – të bazuara mbi anën e sulmeve tashmë të njohura.

Ekzistojnë gjithashtu edhe modele hibride të cilat kombinojnë mënyrat e mësipërme të detektimeve. Në modelin me detektim të nënshkrimit, avantazhi është se përqindja e alarmeve të rremë është shume i ulët në mënyre tipike. Disavantazhi është se mund të detektohen vetëm sulme të njohura më parë. Në krahun tjetër, sistemet IDS të bazuara mbi anomalinë detektojnë variacione të sulmeve të njohura si dhe sulme të reja gjithashtu, por përqindja e alarmeve false është shumë e lartë. Disa prej njësive matëse të tyre janë: përqindja e vërtetësisë, saktësia si dhe koha e përgjigjes. Referenca [283] na jep një prezantim më të përgjithshëm të 'IDS'-ve në rrjetet e telefonisë së lëvizshme[7].

Tabela 7 Aplikacione sigurie për Smartphone

Produkti	**Karakteristikat**	**Sistemi opërativ**	**Liçensa**	**Referenca**
WaveSecure	Lock and Wipe Backup and Restore Localization and SIM Tracking	Android BlackBerry Symbian Windows Mobile J2ME	Komercial	[277]
Norton Mobile Security Lite	Theft of Private Stuff Unauthorized Access Mobile Viruses Malware and Threats Harmful Doënloads	Android	Komercial	[278]
iCareMobile	Parëntal Control Automatic Pornographic Content Detection	Android Symbian	Pa pagese	[279]

BullGuard Mobile Security 10	Antivirus and Anti-Spyëare Anti-theft Parëntal Control Firewall Spam-filter Basic Backup	Android BlackBerry Symbian Windows Mobile	Komercial	[280]
Kaspersky Mobile Security 9	Privacy Protection Anti-theft Parëntal Control Encryption Anti-Spam Anti-Malware Fireëall	Android BlackBerry Symbian Windows Mobile	Komercial	[281]
Lookout Mobile Security	Lock and Wipe Backup Wipe Privacy of Data	Android iPhone	Pa pagese	[282]

Në vijim, ne do t'i kategorizojme 'IDS'-të ekzistuese duke përdorur këto tiparë[7]:

- Parimet e detektimit
  - Detektim i anomalisë
    - ❖ Aftësim i makinës
    - ❖ Konsum i energjisë
  - Të bazuara mbi nënshkrimin
    - ❖ Përcaktim automatik
    - ❖ Manual

- Arkitektura
  - E shpërndarë
  - Lokale
- Veprimi
  - Aktiv
  - Pasiv
- Mbledhja e të dhënave
  - Thirrje të sistemit
  - CPU, RAM
  - Tastierës
  - SMS, MMS
- Sistemi i operimit : Android

Në radhë të parë, unë do të përqëndrohem në IDS-të e telefonave bazuar mbi principet e detektimit të përdorura për të gjetur anomalitë: detektim i anomalisë (e cila përfshin aftësimin e makinës dhe konsumin e energjisë), bazimin mbi nënshkrimin (përcaktimi automatik apo manual) si dhe përforcim i politikave gjatë kohës së përdorimit. Më pas, ne do të shohim të dy tipet e arkitekturës, lokale e të shpërndarë. Më tej, do të dalloj mjetet, ato të cilat ndërmarrin veprime apo ato të cilat shërbejnë vetëm për detektim të anomalive. Pastaj do të klasifikoj IDS-të duke marrë parasysh cfarë lloj të dhënash përdoren si input dhe nga sistemi opërativ. Për çdo karakteristikë, të gjitha zgjidhjet do të shfaqën në rend kronologjik [7].

1) Principet e detektimit: Unë do të ndaj IDS-të duke përdorur principet e mëposhtme të detektimit:
   - ❖ Detektim i anomalisë
   - ❖ Bazim mbi nënshkrimin
   - ❖ Përforcim i politikave gjatë kohës së përdorimit

a) *Detektim i anomalisë:* Sistemi i detektimit të anomalisë krahason sjelljen "e parashikuar" të smartphone me atë "reale" (veprimet e ekzekutuara gjatë kohës së punës). Zgjidhjet e përfshira në këtë paragraf ose monitorojnë aktivitete të vecanta në telefon, p.sh. shërbimet e SMS apo MMS, apo koneksionet Bluetooth, apo analizën mbi konsumin energjitik.

Arkitektura e përgjithshme e një smartphone mund të ndahet në shtresat e mëposhtme (si thuhet edhe ne referencën [284]):

- User
- Application
- Virtual machine ose guest OS
- Hypërvisor
- Physical

Për çdo shtresë funksionale, autorët propozojnë karakteristika të vecanta të ndryshme të cilat duhen mbledhur për të matur sjelljen dhe ato që përdoren nga IDS për detektim të anomalisë.Tabela II jep disa nga vecoritë që mund të maten në çdo shtresë të vecantë.

Vëzhgimet e smartphone të bazuara mbi anomalinë bazohen ose mbi teknikat e aftësimit të makinës ose mbi monitorimin e konsumit të energjisë.

Aftesimi i makines: Artikulli [269], i autorëve të njëjtë me [271], propozojnë një 'Kontroll të sjelljes në grup proaktiv' (Proactive Group Behaviour Containment - PGBC), një strukturë e cila fokusohet në kufizimin e përhapjes së programeve keqdashës në rrjetet e mesazheve sikurse IM dhe SMS/MMS. Përbërësit primarë të PGBC janë grafikë shërbim-sjellje të gjeneruar prej modelit 'mesazh klient'(client messaging) dhe grupimeve të sjelljes të cilët ndajnë grafikun shërbim-sjellje në grupe me sjellje të ngjashme. Autorët paraqësin një algoritem i cili, në prani të sulmeve keqdashëse, në mënyrë automatike identifikon klientët me të cënueshëm bazuar mbi ndërveprimin ndërmjet tyre. Për më tëpër, ata përshkruajnë një strukturë të kontrollit proaktiv e cila i aplikohet dy mekanizmave që përdoren më zakonisht (më

specifikisht, kufizueshmëria e shkallës së vleresimit dhe karantina) në listën e klientëve më të cënueshëm në një rrjet mesazhesh kurdoherë kur një virus apo kërcënim bllokohet [7].

Ho dhe Hing kërkojnë të identifikojnë modele të sjelljeve të përgjithshme në sulmet e telefonave për te ndërtuar një model të përgjithshëm mbrojtjeje. Për të zvogëluar përhapjen e sulmeve mobile, autorët prezantojnë një model të zgjeruar (i cili ndërthur punën e [265] dhe [266]), i cili është një motorr i bazuar mbi Java që është i pavarur nga platforma. Në vijim të zgjidhjeve të mëparshme, ky model përfshin një tipar për të bllokuar kërkesat automatike dhe të heshtura(silent) për transmetim të file-ve të instalimit që përmbajnë virus prej telefonave të prekur nëpërmjet MMS, Bluetooth, Infrared, E-mail, dhe IM (Instant Messaging). [7]

Shtresa	**Karakteristikat**
Hypërvisor	Address spaces Data types, data constructors and data fields System parameters Virtual registers System calls Communication protocols
Application	Application manager framework Security frameëork Messaging framework Multimedia frameëork
User	SIM/phone password Key-stroke/T9 usage/speling analysis Top-n called/texted numbers/contacts based on hour, day, week, month

	Frequently executed smartphone applications Smartphone application usage analysis Bluetooth/wi-Fi usage analysis

Schmidt [266] shfrytëzon procesin e monitorimit të smartphone për të marrë karakteristika të cilat mund të përdoren në algoritmat e aftesimit të makinës për të detektuar anomalitë. Ky model përmban një klient monitorues, një Sistem të Largët të Detektimit të Anomalive (Remote Anomaly Detection System-RADS) dhe një komponent vizualizues. Klienti monitorues, i cili shfaq qëllimin kryesor të këtij dokumenti, është një klient i cili punon në një smartphone dhe ka tre përbërës kryesor[7]:

- Ndërfaqën me përdoruesin
- Moduli i komunikimit, i cili menaxhon koneksionin me RADS
- Nxjerrësi i Tiparëve (Feature Extractor), i cili implementon matjet dhe vëzhgimet e burimeve dhe komponentëve të tjerë

RADS është një shërbim i cili merr, prej klientit monitories, tiparët e vëzhguara dhe informacioni i nxjerrë, i ruajtur në bazën e të dhënave, për tu implementuar në algoritmin e makinës.

Disa zgjidhje të tjera përdorin një qasje probabilistike për të gjurmuar profilet e sjelljeve në smartphone. Për shembull, [267] shfaq një sistem detektimi të bazuar në sjellje (pBMDS) i cili adapton një qasje probabilistike përmes korrelacionit të të dhënave nga përdoruesi me thirrje të sistemit për detektim të aktiviteteve jonormal. pBMDS vëzhgon sjellje unike të aplikacionit dhe indikacioneve të përdoruesit dhe thekson modelin e fshehur të Markovit për të mësuar sjelljen e aplikacionit e përdoruesit nëpërmjet tranzicionit të gjëndjeve të procesit dhe modeleve të reagimit të përdoruesit. Ky vëzhgim përdoret për të kuptuar diferencën në sjellje ndërmjet aplikacioneve të inicializuara prej përdoruesit dhe atyre të inicializuara nga programet që paraqësin rrezik

(malware). Tipari më i spikatur i kësaj metode zgjidhjeje është se aftësia për detektim të kërcënimeve fokusohet në dallimin e sjelljeve të cilat nuk janë njërëzore në vend që të mbështetet në sulme të njohura tashmë. Për më tepër, në procesin trajnues, pBMDS nuk e kërkon numrin e kampioneve negative të jetë i njëjtë me ato pozitive. Sistemi përdor grafikun e sjelljeve, i cili reflekton gjëndjet e ndërmjetme të procesit përmes çdo thirrjeje të sistemit bazuar në kërkesat e përdoruesit, p.sh. shtypja e tasteve[7].

Në [285], të dhënat mbi sigurinë monitorohen në varësi të kohës në mënyrë të vazhdueshme e më pas vleresohen prej metodës Abstraksion Temporal i Bazuar në Njohuri (Knowledge-Based Temporal Abstraction – KBTA). Duke përdorur KBTA, të dhënat e monitoruara vazhdimisht (p.sh. numri i SMS të dërguara) dhe ngjarjet (p.sh. aplikacionet e instaluara) integrohen në fushën e sigurisë së smartphone, e cila është një ontologji për abstraksion të modeleve të pakuptimta prej të dhënave të sigurisë të papërpunuara dhe të orientuara në kohë. Këto modele përdoren për të krijuar modele më të larta, koncepte të orientuara në kohë dhe modele të quajtura *abstraksione temporale.* Këto abstraksione temporale të gjeneruara automatikisht monitorohen më pas për të detektuar veprime të dyshimta temporale dhe për të ndezur një alarm. Këto veprime janë kompatibel me një set të paracaktuar klasash sulmesh të përcaktuara prej ekspërtëve të sigurisë të specifikuara brenda disa kufijeve kohore dhe disa vlerash të caktuara.

Andromaly [286] është një strukturë e përgjithshme dhe modulare për të detektuar kërcënime në smartphone Android me përdorim të teknikës së bazuar mbi detektim të anomalive. Ky sistem bazohet mbi një host IDS që përmban disa njësi matëse të sistemit, si konsumi i CPU, numri i paketave të dërguara, numri i proceseve në punë. Kjo strukturë kompozohet prej katër elementeve, përkatesisht shfaqësi i tiparëve (feature extractor), procesorët, sherbimi kryesor dhe nderfaqja grafike me përdoruesin. Klasifikuesit e përdorur për të vlersuar strukturën janë mjetet-k (k-means), regresioni

logjistik, histogramat, pema e vendim-marrjeve, rrjetet Bajesiane (Bayesian Netëork) dhe Bajesi Naiv (Naive Bayes); për më tepër, një qasje filtruese përdoret për selektimin e të dhënave. Një total prej 88 të dhënash mblidhen për çdo aplikacion të monitoruar. Rezultatet ekspërimentuese tregojnë se Bajesi Naiv dhe regresioni logjistik ishin supërior ndaj klasifikuesve të tjerë ne shumicën e konfiguracioneve bazë dhe 'rezultati i peshkatarit' (fisher score) ishte selektuesi i të dhënave më i mirë[7].

[287] prezanton një strukturë për të analizuar sjelljen e aplikacioneve në menyrë dinamike për të detektuar kërcënimet në sistemet Android nëpërmjet mbledhjes së gjurmëve të sistemit prej përdoruesve të ndryshëm përmes 'crowdsourcing' (vëzhgimit masiv të përdoruesve) dhe një serveri qëndror. Një aplikacion klient, Crowdroid, monitoron thirrjet në sistem të Linux dhe i dergon ne serverin qëndror, i cili më pas i ndan të dhënat dhe krijon një vektor të thirrjeve të sistemit (system call vector). Në fund, çdo set i të dhënave grumbullohet përmes një algoritmi të grumbullimit, specifikisht 'mjeti-2' (2-means).

Konsumi i energjisë: një shembull i monitorimit të konsumit të energjisë prezantohet neë[288], ku autorët propozojnë një IDS të bazuar mbi baterinë. Kjo zgjidhje kryen ndjeshmerinë e sjelljes anormale të baterisë dhe ndryshimin e energjisë për të detektuar një llojshmëri sulmesh dhe përfshin dy module [7]:

- Motorr i Detektimit te Nderhyrjes në Host (Host Intrusion Detection Engine-HIDE), i cili mat energjinë e konsumuar në një hark kohor (në varësi të një algoritmi)
- Motorr i Gjurmimit të Nënshkrimit i analizuar në Host (Host Analyzed Signature Trace Engine-HASTE), i cili krahason diagramën e frekuencës së sulmeve bazuar në nënshkrimin e fuqisë dhe krahason këto nenshkrime me një list të shkurtër të nënshkrimeve të sulmeve të ditura.

Në mënyrë të ngjashme me zgjidhjet e mëparshme, Kim [275] propozon një strukturë malware ‘fuqi-kujdes’ të detektimit të sulmeve e cila monitoron, detekton dhe analizon malware‘energji-lakmi’. Kjo strukturë përbëhet nga një monitorues fuqie, për të mbledhur kampionet e fuqisë dhe për të ndërtuar historinë e konsumit të fuqisë prej kampioneve të marra, dhe prej një analizuesi të të dhënave, për të gjeneruar një nënshkrim fuqie nga historia e marrë [7].

Një shembull tjetër i monitorimit të konsumimit të energjisë propozohet në [289], i cili i bazon vëzhgimet e tij në faktin se çdo aktivitet keqbërës në smartphone konsumon fuqine e batërise. Kështu, zgjidhja e propozuar (VirusMeter) e kryen detektimin e malware duke përdorur gjëndjen e makinës dhe modelin e konsumit të energjisë duke krahasuar fuqinë aktuale të harxhuar me fuqinë që mund të harxhohej, sipas modelit të paracaktuar të konsumit të energjisë. Modeli i fuqisë i përqëndruar të përdoruesi karakterizon fuqinë e konsumuar si funksion të opëracioneve të zakonshme të përdoruesit dhe faktorëve të mjedisit rrethues (p.sh. telefonatat, SMS, MMS). Për të marrë të dhënat, për çdo përdorues ndërtohet një gjendje e makinës. Kjo gjendje përshkruan zhvillimin e ngjarjeve të brendshme të lidhura me veprimet e përdoruesit.

Edhe nëse konsumi i energjisë konsiderohet një limit, Yan ne [290] konsideron impaktin pozitiv në parandalimin dhe shuarjen e malware në telefon. Edhe nëse asnjë zgjidhje specifike nuk është implementuar, tre teknika potenciale, bazuar mbi limitet e vëna prej smartphone, propozohen [7]:

- *Monitorimi i konsumit të fuqisë*, i cili merret me aftesinë për të detektuar një sulm bazuar mbi konsumin e baterisë në smartphone
- *Përforcim i elementit harware*, në të cilën të gjitha modulet hardware që nuk janë përdorur shpesh dhe që mund të përdoren për përhapje të malware (p.sh. GPS ose Bluetooth) fiken

- *Rritje e llojshmërise së platformave*, në të cilën API të ndryshme përdoren për të krijuar dhe ekzekutuar një aplikacion.

b) *I bazuar mbi nënshkrimin:*Ky paragraf diskuton rreth mekanizmave të përdorura në smartphone për të detektuar anomali duke u bazuar mbi nënshkrimin.[7] Kjo qasje kontrollon nëse çdo nënshkrim i marrë prej aplikacioneve përputhet me nënshkrimin në bazën e të dhënave të malware. Baza e të dhënave për malware mund të përcaktohet në mënyrë automatike ose manuale.

*Përcaktimi automatik:*Venugopal në [291] aplikon teorinë e vendimarrjes të Bajes në librarinë e linkeve dinamike (dynamic-link library-DLL) përdorimin e programit për të detektuar viruse. Në fakt, qëkur shumica e viruseve mobile kanë funksionalitete të
zakonshme (p.sh. fshirja e fileve të sistemit, dërgim i MMS), këto programe kanë nevojën e përdorimit të DLL. Duke shfrytëzuar tiparët e zakonshme të përdorimit të DLL përmes viruseve, qasja e propozuar mund të detektojë viruse të vjetra dhe të reja : një klasifikues merr si të dhënë në hyrje një vektor binar që specifikon prezencën (ose jo) të çdo funksioni DLL në setin e modeleve.
Në [292, 293] autorët propozojnë adoptimin e një qasje të dyaneshmë për të bllokuar përhapjen e programeve keqbërëse në smartphone:

- Në nivelin e terminalit, ku një test grafik Turing dhe nënshkrime të bazuara në identitet bllokojnë mesazhe të paautorizuara të nisen nga telefona të prekur
- Në nivelin e rrjetit, ofruesi i Internetit, në mënyre automatike, shtyn copëza software në terminalet e prekura

Në [294], nënshkrimet që duhet të krahasohen, gjenerohen automatikisht duke analizuar aktivitetet e paisjes. Autorët diskutojnë dinamikat e malware mobile

që përhapen si pasojë e kontaktit të afërsisë dhe zhvillojnë tre strategji për të detektuar dhe shuar malware-t e afërsisë, respektivisht [7]:

- Detektimi lokal, në të cilin paisjet detektojnë kur ato bëhen të infektuara dhe ndalojnë përhapjen e mëtejshme
- Përhapjen e nënshkruar të afërsisë, në të cilën paisjet krijojnë nënshkrime të malware të bazuara mbi përmbajtjen dhe i përhapin ato përmes komunikimit të afërsisë gjithashtu
- Përhapja e nënshkruar gjerësisht, në të cilën një server qëndror mbledh të dhëna prej disa paisjeveindividuale, detekton malware që po përhapet dhe i shpërnadan nënshkrimet në të gjithë smartphone-t

Bauckhage ne [295] prezanton një skemë probabilistike të përhapjes për detektim të anomalive që tregojnë malware, që bazohet në mënyrë të përdorimit të paisjes. Ideja bazë është që të modelohen varësi të kampioneve dhe tiparëve më anë të një grafiku, i cili më tej shërben si domain i procesit të Markov. Algoritmi aplikohet në dy sete të ndara të të dhënave të përftuara nga smartphone gjatë një përdorimi ditor normal.

Të përcaktuara manualisht: Këto mekanizma për të detektuar anomalitë, përftojnë nënshkrimet e malware mobile duke analizuar në mënyrë manuale malware-in dhe sjelljen e tij.

Ellis në [296] prezanton një qasje të re për detektimin automatik të kërcenimeve në smartphone duke përdorur nenshkrimet e sjelljes, të cilat përcaktohen manualisht për të shënuar tiparë të zakonshme ndërmjet një familjeje kërcenimesh. Qasja e prezantuar fokusohet në detektim të modeleve në një nivel më të lartë abstraksioni, ku një model mund të jetë [7]:

- Dërgim i të dhënave të ngjashme ndërmjet paisjeve
- Përhapje në formë peme dhe zbulimi
- Ndryshimi i një paisje server në klient

Idealisht, një model është një sjellje specifike e një kërcënimi dhe duhet të jetë i ndryshëm nga trafiku normal i rrjetit. Frekuenca dhe ndërlidhjet nëpërmjet sjelljeve përmiresojnë shumë precizionin e detektimit. Për të shmangur një nënshkrim të bazuar në sjellje kërkohet një ndryshim thelbësor në sjellje dhe kjo detyrë është paksa e vështirë.

c) Përforcim i politikave gjatë kohes së përdorimit (Run-time Policy Enforcement): ideja bazë e këtyre modeleve është që konsumatorët e kodit mobil praktikisht pranojnë kodin si të mirqënë dhe adaptojnë mekanizma mbështetës për politikat përkatëse të kodit për të detektuar dhe ndaluar anomalitë.

Në ditët e sotme, shumë smartphone janë në gjëndje të suportojnë aplikacione Java, të cilët mund të krijojnë gjithashtu lidhje me Internetin, të dergojnë mesazhe SMS dhe të përformojnë opëracione të tjera të kushtueshme në smartphone. Kështu, një mbështetje e denjë për sigurinë kërkohet për të plotësuar kërkesat e këtij skenari. Në këtë pikë, [297] propozon një qasje për të shtuar sigurinë e Java Micro Edition (J"ME), bazuar në monitorimin e resurseve të smartphone të përformuar nga MIDlet-et. Një gjuhë procesimi e bazuar mbi algjebrën përcakton politikat e sigurisë, ndërsa një monitorues referencash përdoret për të kontrolluar përdorimin e burimeve.

Siguri me kontratë (Secyrity-by-contract) [298. 299] është një zgjidhje e përforcimit të politikave gjatë kohës së ekzekutimit e bazuar në një nënshkrim dixhital i cili:

- Certifikon origjinën e kodit
- E shoqëron kodin me një kontratë

Një *kontratë* përmban përshkrimin e vecorive përkatëse të një aplikacioni dhe ndërlidhjet përkatëse me host-in. Platforma mobile mund të specifikojë kërkesat e kontratës se platformës (*politikat-policy*), e cila duhet të përputhet

me kontratën e aplikacionit. Autorët propozojnë një sërë algoritmash për të verifikuar përputhjen kontratë-politika.

[300] përforcon arkitekturën e sigurisë me kontrata duke shtuar module të reja dhe konfigurime për menaxhimin e kontratave. Në kohën e startimit, sistemi i propozuar selekton konfiguracionin për menaxhimin e kontratave bazuar në kredencialitetet e ofruesit të kontratës; gjatë kohës së punës, sistemi mund të përforcojë politikat e sigurise si dhe të monitoroje kontratën e deklaruar. Në lidhje me sjelljen aktuale të programeve në punë, arkitektura mund të updetojë (update) nivelin e besimit në lidhje me ofruesin e kontratës. Avantazhi kryesor i arkitekturës së propozuar është menaxhimi automatik i nivelit të bësimit të programeve dhe kontraktorëve.

Së fundmi, [273, 274] propozojnë shërbimin e sigurisë Kirin për Android, i cili bën certifikime të pjesshme të aplikacioneve për të zvogëluar riskun në kohën e instalimit. Certifikatat Kirin përdorin rregullat e sigurisë që i përkojnë disa vetive të padëshirueshme në konfiguracion në lidhje me sigurinë e aplikacioneve [7].

2) *Arkitektura:* Lokale ose E shpërndarë: Në këtë seksion ne do të studiojmë mekanizmat të cilët detektojnë ndërhyrjet duke përdorur arkitekturën lokale dhe më pas do të merremi me ato me arkitekturë të shpërndarë[7].

   Ne *arkitekturen lokale*, si faza e mbledhjes dhe ajo e analizimit kryhen lokalisht në paisje dhe asnjë lidhje me serverin nuk kërkohet. Shembuj të arkitektures lokale janë paraqitur, për shembull, në [130, 289, 273, 274]. Në krahun tjetër, *arkitektura e shpërndarë* zakonisht kërkon një komponent të ndarë (p.sh. një server) për të analizuar aktivitetet e paisjeve. Në këtë arkitekturë, komponenti i sigurisë së jashtme mund të kryejë të gjithë analizën mbi aktivitetin e monitoruar te smartphone-it pa pasur problemet e:

   - Konsumit të energjisë
   - Madhësisë së vogël të ekranit
   - Burimeve të limituara

[301] është shembulli i një strukture për të detektuar sulme kundrejt smartphone-it duke përdorur një server të sigurt, të mëvetshëm dhe gjerësisht të sinkronizuar, i cili mbledh kopje të një ose më shumë smartphone-ve dhe aplikon mekanizmin e detektimit. Në këtë mënyrë, shumë teknika të shtrenjta të detektimit, të cilat nuk mund të implementohen lehtësisht në smartphone, aplikohen në server për të shmangur sulmet e smartphone-ve. Kjo arkitekturë përfshin një *gjurmues* në vet telefonin për të ndërhyrë si në thirrjet e sistemit dhe në set sinjalesh të procesit te mbrojtjes, ku një *ripërsëritës* në server më pas ripërsërit ekzekutimin e gjurmëve dhe kërkon për anomali.

3) *Reagimi:* Aktiv ose Pasiv: Në këtë paragraf, do të shohim nëse mekanizmat ekzistues për detektim të ndërhyrjeve reagojne në rast se një kërcënim i ri gjendet, p.sh. kërkojnë të ndalojnë dëmtimin e telefonit nga sulmet [7].

Një *reagim* mund të jetë një strategji, të përmbajë virusin ose malware-in, ose një mekanizëm, si për shembull të lajmërojë përdoruesin për infektimin. Një shembull i strategjisë së reagimit prezantohet në [269], ku lajmërimet për një sulm të mundshëm mblidhen para se të veprojë strategjia e reagimit. Struktura e propozuar mund të veprojë në qëndrën e shërbimit të mesazheve ku rregjistrat e komunikimit të klientëve ruhen. Këto rregjistra mund të analizohen për të gjeneruar një *grafik shërbim-sjellje* për rrjetin e mesazheve: ky grafik proçesohet më tej për të studiuar ecurinë e sjelljes duke krijuar *grafikët e ecurisë së sjelljeve*, p.sh. një grup klientësh sjelljet e të cilëve janë të ngjashme në aspektin e njësive matëse, respektivisht [7]

- Frekuenca e ndërveprimeve
- Madhësia e mesazheve të shpërndara
- Numri i mesazheve
- Numri i lidhjeve të dala drejt klientëve të tjerë
- Lista e kontakteve të gjurmuara

Kur numri i sulmeve në një grafik të ecurisë së sjelljes arrin një prag, mesazhet të cilat i përkasin atij grafiku limitohen fillimisht duke ngadalësuar një sulm të mundshëm. Kur

lajmërimet arrijnë një prag të dytë, algoritmi i mbrojtjes aplikon rezervimin (quarantine), p.sh. bllokon mesazhet e transmetuara prej klientëve të dyshuar prej këtyre grafikëve. Ky hap i lejon grafikëve të ecurisë së sjelljes te hyjnë në versionin e mbrojtjes prej përhapjes së malware.

Në [130], sistemi i propozuar i detekton viruset që janë në gjëndje pune në telefon duke përdorur një autorizim (proxy) i cili bën analizimin e sjelljes së telefonit dhe, kur një virus potencial kapet, i dërgon alarmet e shënuara drejt paisjeve te infektuara si dhe një nënbashkësie paisjet e painfektuara për të parandaluar përhapjen. Këto paisje zgjidhen bazuar në listën e kontakteve të përdoruesit dhe profileve të telefonisë, për të alokuar ato paisje të cilat mund të jenë në kontakt direkt me paisjen e prekur.

Një shembull i mëtejshëm i strategjisë së reagimit jepet ne [270] ku, për të limituar përhapjen e kërcënimeve mbi bazën e MMS dhe SMS, autorët propozojnë një metodologji të mbështetur në një qasje sipas një grafiku kategorizues. Problemi kapet duke përdorur një grafik të mardhënieve sociale ku paisjet kategorizohen në shumë ndarje bazuar në mardhëniet shoqërore ndërmjet tyre. Çdo ndarje përfshin smartphone që ndërveprojnë afërsisht me njëri-tjetrin. Konturet e komunikimeve përftohen prej përdorimit të një gjurme të rrjetit dhe, prej këtyre kontureve, ndërtohet grafiku i mardhënieve shoqërore të smartphone-ve në mënyrë që një rrjet telefonësh të afërt mund të lokalizohen në radhë të parë. Në këtë mënyrë, sistemi mund të lokalizojë telefonat të cilat kanë aftësinë për të infektuar numrin më të madh të telefonave të tjerë. Autorët propozojnë dy skema qasjeje [7]:

- Ndarje e balancuar, ku niveli i çdo ndarjeje zgjidhet të jetë sa më i ngjashëm të jetë i mundur në mënyrë që ndarja e dëmit prej kërcënimeve të balancohet në çdo ndarje
- Ndarje grumbulluese, ku kufijtë përbrenda çdo ndarjeje kanë peshë më të madhe sesa kufijtë ndërmjet dy ndarjeve, kështu që smartphone-t të cilët janë në aspektin shoqëror të afërt me njëri-tjetrin ndodhen në të njëjtën ndarje dhe pikë-nyjet që nuk ndodhen pranë nuk ne të njëjtën ndarje

Ruitenbeek në [302] studion përhapjen e viruseve në smartphone bazuar në MMS dhe propozon disa mekanizma reagimi për të vlerësuar efektivitetin e teknikave të zvogëlimit të virusit. Autorët analizojnë katër skenare të viruseve me MMS: në çdo skenar telefoni dërgon mesazhe MMS me një file të atashuar drejt një telefoni tjetër, i cili selektohet nga lista e kontakteve nga telefoni i infektuar ose ndodh të jetë një numër i rastësishëm. Pasi merr këtë MMS të ri, nëse përdoruesi e pranon këtë file të atashuar, virusi instalohet dhe viktima bëhet i infektuar gjithashtu dhe kalon nën kontrollin e sulmuesit. Mekanizmat e përgjigjes për secilin nga katër skenarët janë [7]:

- Skanimi për gjithë atashimet MMS në të gjitha portat MMS për të detektuar viruset
- Edukimi i përdoruesit
- Imuniteti duke përdorur pjesë të kodit të programit
- Monitorimi për sjellje anormale
- Një listë e zëzë për telefonat të cilët dyshohet se janë të prekur

Përgjigjet ekspërimentale zbuluan se çdo mekanizëm përgjigjeje duhet të jetë mjaftueshëm i shpejtë për të reaguar kundrejt përhapjes së virusit dhe në mënyrë të mjaftueshme diskriminues për të detektuar viruse të tjerë më te fshehtë dhe që përhapen më ngadalë.

4)*Të dhënat e mbledhura [7]:* Siç sygjerohet në [303], monitorimi i të dhënave në një mjedis mobil mund të jetë një sfidë si pasoje e limiteve administrative, teknike apo konceptuale. Vizibiliteti i të dhënave mund të limitohet nga marrëveshje të vecanta për këmbimin e të dhënave, për shembull, rregjisrat e telefonatave mund të jenë nën kontrollin e administratorit. Për më tepër, na duhet të konsiderojmë se për arsye të integrimit të disa ndërfaqëve të komunikimit, një smartphone mund të lidhet, në të njëjtën kohe, me pika të ndryshme aksesimi (si Bluetooth, Wi-Fi) dhe kështu sasia e të dhënave të marra mund të jetë shumë e madhe. Së fundmi, duhet të marrim parasysh se kemi tëbëjmë me disa tipe sulmesh, p.sh. Trojan, nuk përfshihen në aktivitetet e komunikimit, kështu që ata nuk janë të dukshëm në rrjet.

Bazuar në faktin se të gjitha zgjidhjet e bazuara në detektimin e ndërhyrjeve duhet të aksesojnë disa tiparë të smartphone-ve, na duhet të marrim në konsideratë problemin e privatësisë të të dhënave të aksesuara. Duke aksesuar një smartphone, disa copëza informacioni mund të ripërdoren, si për shembull vendndodhja e përdoruesit, komunikimet, kontaktet përsonale. Të gjitha këto tipe informatash janë, dukshëm, private dhe nuk duhet të ndahen me përsona të tretë: kështu, çdo mjet për detektim ndërhyrjesh duhet të punojë vetëm me autorizimin e përdoruesit në rast se do të aksesojë të dhëna private. Të dhënat private nuk duhet të nxirren prej smartphone-it nga asnjë mekanizëm, duke përfshirë sistemin që duhet të mbrojë telefonin. Kështu , një sistem i detektimit të ndërhyrjeve duhet ndërtuar në mënyrë që [7]:

- Të dhënat private të përdorura gjatë procesit të detektimit të ndërhyrjeve nuk transferohen jashtë paisjes
- Komunikimi ndërmjet funksionaliteteve të detektimit të ndërhyrjeve në telefon duhet të kufizohen në gjenerim alarmesh të ndërhyrjeve

Në [272], autorët fokusohen në kërcënimet dhe sulmet që dhunojnë privatësinë e përdoruesit duke “nuhatur” sensorët e smartphone-it. Autorët paraqësin një model kërcënimi bazuar në përdorimin e sensorëve dhe dizenjojnë një strukturë të përgjithshme për një sistem mbrojtjeje. Kjo strukturë përbëhet nga tre module: (i) politikat e motorrit, (ii) ndërhyrësi, (iii) ndërhyrja e përdoruesit. Politikat e motorrit, bazuar mbi të dhënat e futura prej veprimeve të përdoruesit dhe monitorimit/profilizimt të aplikacionit, përcaktojnë aksesin: këto vendime bazohen kryesisht në monitorimin dhe profilizomin e aplikacioneve pa kërkuar shumë ndërhyrjen e përdoruesit. Disa politika merren parasysh, si lista e bardhe, lista e zezë dhe gjurmimi i rrjedhjes së informacionit. Ndërhyrësi pozicionohet ndërmjet aplikacionit dhe sensorëve dhe/ose ndërmjet apikacionit dhe rrjetit dhe ai përforcon vendimarrjen e politikave të motorrit. Ndërhyrja e përdoruesit nuk është një përbërës i nevojshem, përderisa përdoruesi thjesht lajmërohet duke kërkuar leje për vendimet e saj. Për çdo modul, mekanizma të ndryshëm përfliten dhe diskutohen por pa ndonjë zgjidhje konkrete [7].

Në paragrafët në vijim do të diskutojmë zgjidhje të cilat analizojnë lloje të ndryshme të të dhënave në një smartphone, respektivisht:

- Ngjarjet e sistemit opërativ (si thirrjet e sistemit)
- Matjet (si CPU, RAM, aktivitetet I/O)
- Tastet e shtypura
- Ngjarjet e komunikimit (si SMS, MMS)

a*) Ngjarjet e sistemit operativ:* Këto ngjarje ndikojnë ato aktivitete që lidhen me funksionimin normal të sistemit opërativ (SO), të cilat mund të përdoren për të marrë informacionin përkates rreth sjelljes së smartphone-it dhe përfshijnë [7]:

- Thirrjet e sistemit
- Thirrjet e funksionit
- Veprimet e rrjetit

Të dhëna prej këtyre ngjarjeve mund të merren duke përpunuar disa mekanizma të bashkëngjitur me SO. Megjithatë, në shumë raste ngjarjet e SO nuk mund të monitorohen sepse nuk ka ndërfaqë aksesi direkte. Kështu, në këto raste disa shtesa të SO janë të nevojshme.

Zgjidhja e propozuar në [304] monitoron thirrjet e sistemit të ekzekutuara nga proceset që janë në punë dhe emërton kodin e ekzekutimit sipas asksesit të tij në ndërfaqën e rrjetit (p.sh. Wireless, GSM, Bluetooth). Emërtimet më pas transferohen ndërmjet proceseve dhe burimeve të sistemit si pasojë e aksesit apo ekzekutimit. Gjatë disa opëracioneve të kujdesshme, emërtimet e mbledhura në këtë rruge krahasohen me një sërë rregullash që i lejojnë përdoruesit të specifikojnë kontrollin e aksesit qartësisht të shërbimeve dhe të dhënave. Procesi i emërtimit, i cili mund të përfshijë procese dhe burime, ashtu si dhe politika të përforcimit, përformohet nga monitoruesi që i referencohet nivelit të kernelit. Struktura është e pavarur nga SO sepse përdoret gjuha e politikave që i lejojnë përdoruesit të shprehin cfarë veprimesh janë të lejueshme prej klasave të specifikuara të programeve në lidhje me klasa specifike të burimeve.

Një tjetër mekanizem, MobileSandbox [268], monitoron thirjet e sistemit të lëshuara prej programit që përdoruesi do të instalojë nëtelefonin e tij. MobileSandbox është një sistem që përformon në pasivitet (background) i cili kampionon dhe përkthen një opëracion në një sekuencë thirrjesh API të cilat program i lëshon gjatë ekzekutimit. Sistemi i propozuar mund të përdorë imituesin (emulator) e paisjes ose paisjen aktuale. Për të përformuar analizen dinamike të malware-it, zgjidhja e propozuar përfshin tre hapa, respektivisht [7]:

- Mbledhja e kampioneve të programit sa më e plotë të jetë e mundur
- Analizimi i kampioneve sa më i plotë të jetë e mundur
- Ndërmarrja e veprimeve të caktuara si përgjigje e analizës

Një zgjidhje për platformën Android prezantohet nga Schmidt në [307]: struktura e dhënë bën analizën e programeve të ekzekutueshme për të marrë thirrjet e funksioneve të tyre duke përdorur komandën 'readelf'. Kjo komandë kthen informacion të detajuar në ripozicionimin dhe tabelën e simboleve të çdo file ELF (Executable and Linking Format). Të dhënat e përftuara nga kjo analizë është lista statike e thirrjeve të funksioneve të referuara për çdo komandë të sistemit. Më tej, këto thirrje krahasohen me malware e ekzekutuar për tu klasifikuar.

b) *Matjet:* Një set matjesh përfshijnë indikatorë të ndryshëm të përformancës së smartphone-it, si aktiviteti i CPU, konsumi i memories, aktiviteti i file-ve I/O dhe aktiviteti nga rrjeti I/O [7]. Ideja thelbësore është se, duke supozuar se ndryshimet në përdorimin e smartphone-it janë graduale, përdorimi normal qëndron konstant përgjatë gjithë kohës. Kështu, ne mund të shohim këto profile të sjelljes dhe t'i përdorim ato për krahasim me sjelljet normale në mënyre që të gjejmë anormalitetin. Disa prej këtyre tiparëve (si hapsira e lirë e RAM-it, koha në të cilën përdoruesi nuk është aktiv, numri i proceseve, përdorimi i CPU, numri i SMS të dërguara), të cilat përdoren për detektim të anormalitetit, diskutohen në referencat [266, 286].

c) *Shtypja e tasteve:* Disa zgjidhje përdorin teknikat e *aksesimit përmes tastieres* (keystroke logging) për të detektuar anomalitë [7]. Këto teknika gjurmojnë tastet e

shtypura në tastiere për të monitoruar veprimet e përdoruesit. Normalisht, aksesimi (logging) kryhet në një mënyrë konvertuese në mënyrë që përdoruesi nuk i bëhet i ditur monitorimi. Teknika konvencionale e detektimit të anomalive të bazuara në sjellje fokusohet në ritmin e përdorimit të tasteve apo në probabilitetin e ndryshimit të grupit të komandave.

Në [308], aksesimi ndërmjet tastieres aplikohet në smartphone për të detektuar veprimet e përdoruesve të jashtëligjshëm duke përdorur një skemë e cila përdor një proces në hije (background) për të regjistruar tastet. Tastet e futura ndahen në *afatgjata* dhe *afatshkurtra*: përdorim i frekuencës për ato afatgjata, një algoritëm i detektimit të anomalive bëhet mbi bazën e profilit të përdoruesit; pastaj, tastet afatshkurtra krahasohen me profilin e përdoruesit për të detektuar përdoruesit e jashtëligjshëm.

[309] demonstron se dinamikat e shtypjës së tasteve të një përdoruesi mund të përkthehet në një set tiparësh për identifikim të saktë të përdoruesve. Në këtë pikë, të dhënat e tasteve të futura të njëzet e pesë smartphone-ve janë mbledhur dhe analizuar: bazuar në to, gjashtë tiparë të tasteve të shtypura të ndryshme prej njëra-tjetrës merren dhe përdoren për identifikim të përdoruesit. Rezultati tregon se sistemi i propozuar i identifikimit të përdoruesit ka një gabim mesatar prej 2% sipas mënyrës së detektimit dhe vleresimi i gabimit për refuzim të përdoruesve të ligjshëm bie në zero në modelin e verifikimit të PIN.

d) *Ngjarjet e komunikimit:* Ngjarjet e komunikimit janë tregues të një klase të vecantë ngjarjesh të cilat ndodhin në një paisje në nivel aplikacioni, sic janë veprimet e nivelit të lartë (p.sh. dërgimi/marrja e SMS, file-ve). Në mënyrë tipike këto veprime kompozohen nga disa veprime elementare të cila nuk mund të gjenerohen automatikisht prej SO. Ngjarjet e komunikimit përfshijnë opëracione si dërgimi dhe marrja e mesazheve apo shkarkimi/ngarkimi (download/upload) i file-ve [7].

[310] diskuton kërcënimet e smartphone-ve të pasqyruara nga kompjuterat përsonal (PC) dhe propozon një aplikacion detektimi i cili monitoron SMS e derguara pa autorizim të përdoruesit. Bose dhe Shin në [271] investigojnë përhapjen e kërcënimeve dhe viruseve në telefonat mobil të cilët përhapen fillimisht me anë të SMS/MMS dhe komunikimeve

radio me valë të shkurtër (p.sh. Bluetooth). Çdo smartphone modelohet si një agjent mobil i aftë për të dërguar mesazhe SMS dhe të zbulojë paisje të tjera që janë të pajisura me Bluetooth. Për të identifikuar një set vektorësh sjelljesh të zakonshëm, si dhe për të zhvilluar algoritmat e detektimit të viruseve mobil apo të kontrollit, viruset ekzistues të telefonave investigohen. Autorët studiojnë cënueshmërinë e mesazheve SMS/MMS apo te Bluetooth-it në thellësi dhe identifikojnë cënueshmëritë që mund të paraqiten prej viruseve të ardhshëm. Së fundmi, ata zhvillojnë një diagram gjëndjeje të një virusi të përgjithshëm i cili mund të përhapet me anë të SMS/MMS apo Bluetooth. Gjëndjet e zbulimit, infektimit dhe replikimit të virusit të përgjithshëm implementohen në një strukturë malware të bazuar mbi agjentin (agent-based framework) për të studiuar strategjitë e përhapjes dhe tëkontrollit [7].

[311] prezanton një qasje të re për testimin e sigurisë ndaj agjentëve përdorues të MMS duke marrë në konsideratë efektin e infrastrukturës në shpërndarjen e mesazheve MMS dhe më pas duke përdorur një infrastrukturë virtuale për të përshpejtuar fazën e testimit. Si në [276], shkrimi përdor crregullsinë për të shpërndarë mesazhet MMS të crregullta drejt agjentëve përdorues, duke hasur disa mbiderdhje të memorizuese (buffer). *SmartSiren* [130] është një sistem për detektim të viruseve dhe alarmi për smartphone i cili mbledh informacione për aktivitetin e komunikimit dhe bën analizime për të detektuar sjellje anormale. Për të ndalur thyerjet prej viruseve potenciale, autorët kërkojnë të minimizojnë numrin e smartphone-vë të prekur prej viruseve të rinj të lëshuar së fundmi. Një agjent i cili nuk e rëndon programin punon në secilin smartphone, ndërsa një centrali (proxy) të qëndërzuar asiston në detektimin e viruseve dhe në procesin e alarmimit. Çdo agjent gjurmon aktivitetet e komunikimit të paisjes dhe në mënyrë përiodike raporton një përmbledhje të këtyre aktiviteteve drejt centralit. Centrali përformon analiza mbi raportet dhe detekton çdo sjellje virale të ndonjë paisjeje apo sistemi në tërësi.

Së fundmi, [312] propozon një skemë që mund të detektojë sjellje anormale të SMS me saktësi të lartë. Autorët fillojnë të analizojnë një gjurmë SMS të mbledhur në një përiudhe 5-mujore dhe sipas rezultateve të analizës, katër skema detektimi janë

propozuar. Çdo skemë ndërton një profil të sjelljes normale për çdo përdorues të SMS dhe më pas i përdor ato për të detektuar anormalitet në SMS në rrjet (on-line) dhe duke transmetuar (streaming). Përderisa një skemë arrin të ruajë në memorie vetëm pak gjëndje të çdo përdoruesi, kjo krijon një kohëvonesë shumë të vogël gjatë detektimit 'on-line'. Së fundmi, autorët përcaktojnë këto katër skema dhe gjithashtu dy qasje hibride me gjurmë reale të SMS duke treguar se qasja e propozuar mund të detektojë më tepër se 92% të sulmeve me anë të SMS me një rankim të alarmeve të rreme prej 8.5% dhe 66% e sulmeve ku nuk gjenerohet asnjë alarm i rremë.

5) *Sistemi i operimit:* Në këtë pjesë , do të studiohet impakti i sistemit opërativ i përdorur për detektim të kërcënimeve ndaj tij, më specifikisht, sistemi Android [7].

Android është një sistem operativ për telefona i bazuar në një version të modifikuar të Linux. Aplikacionet shkruhen prej një bashkësie të gjerë programuesish për të zgjeruar funksionalitetin e paisjes.

[314] dhe [315] na prezantojnë një set rezultatesh të përftuara prej sigurisë së smartphone-ve Android. Fillimisht autorët analizojnë strukturën Android dhe Bërthamën Linux (kernel) për të kontrolluar funksionet e sigurisë, duke mbikqyrur gjithashtu disa mjete mbi sigurinë dhe disa mekanizma për të rritur sigurinë e smartphone-ve. Më pas autorët studiojnë mundësitë e shtimit të mekanizmave të detektimit të malware në nivel kernel p.sh. duke monitoruar ngjarjet çeles që ndodhin në kernel (hap file, aktivitetet e sistemit mbi file-at). Së fundmi ata aplikojnë analizën e thirrjeve statike të sistemit për të detektuar malware në ekzekutimet ELF, duke marrë parasysh një pemë vendimesh për të marrë vendimin nëse një aplikacion i ri është i dyshimtë në krahasim me një aplikacion të analizuar më parë (të mirë apo të këqinj qofshin ato). [316] prezanton një arkitekturë për të detektuar anormalitetet në platformën Android. Në [318], autorët na sigurojnë një përllogaritje mbi sigurinë e strukturës Android: fillimisht, ata diskutojnë mekanizmat aktuale të sigurisë të bashkëngjitur Android (respektivisht, mekanizmat Linux, vecoritë mjedisorë dhe mekanizmat e specifikuara prej Android); së dyti, ata propozojnë disa zgjidhje mbi sigurinë për të parandaluar kërcënimet e Android, duke përdorur pesë "pikat e kërcënimeve" [7]:

1) Kërcënime të cilat sulmojnë disponueshmërinë, konfidencialitetin apo integritetin duke përdorur në mënyrë keqdashese leje-kalimet e lejuara në një aplikacion të instaluar.
2) Kërcënime të cilat sulmojnë disponueshmërinë, konfidencialitetin apo integritetin të cilat ndodhin kur një aplikacion cënon kernel-in e Linux apo libraritë e sistemit
3) Kërcënime të cilat sulmojnë disponueshmërinë, konfidencialitetin apo integritetin e përmbajtjeve private apo konfidenciale: p.sh. aplikacione të cilat mund të lexojnë të dhënat e një karte SD apo sulme mbi komunikimet e largëta 'wireless'.
4) Sulme të cilat shterojnë burimet e smartphone: p.sh. aplikacionet Android nuk kanë hapsirë disku dhe as kuota memorie fikse apo CPU
5) Kërcenimë të cilat kompromentojnë rrjetet e brendshme apo ato të mbrojtura: p.sh. sulmuesit mund të përdorin paisjet Android për të kompromentuar paisje të tjera, kompjutera apo edhe rrjete nëpërmjet rrjeteve në punë apo portave të skanimit, SMS/MMS/e-mail të këqija dhe një llojshmëri sulmesh të tjera

[319] propozon një zgjidhje për smartphone-at Android përsa i përkët aspkektit të sigurisë, e cila përmban Security-Enhanced Linux.

Taintdroid[87] është një shtesë që i bëhet Android për të gjurmuar rrjedhen e të dhënave përmes aplikacioneve palë e tretë. TaintDroid supozon se aplikacionet e shkarkuara që funksionojnë si të tretë nuk janë të besueshëm dhe monitorojnë sesi këto aplikacione aksesojnë dhe manipulojnë të dhënat përsonale të përdoruesit. TaintDroid në mënyrë automatike etiketon të dhënat prej burimeve sensitive dhe aplikon etiketa ndërkohë që të dhëna sensitive kalojnë përmes variablave të programit, file-ve dhe mesazheve ndërmjet proceseve. Kur të dhënat transmetohen përmes rrjetit, TaintDroid kodon etiketat e të dhënave, aplikacioni është përgjegjës për transmetimin e të dhënave dhe për destinacionin e tyre. Kohë-vonesa e patestuar e përformancës është 14%.

Enck në [320] studion 1100 aplikacione Android të cilët janë pa pagesë duke përdorur një dekompilator për të rikthyer kodin e aplikacionit në Java prej imazhit Dalvik të tij të instalimit dhe duke analizuar në mënyrë statistike më tepër se 20 milion rreshta kodi. Studimi tregon se shumë aplikacione keqpërdorin informacionin privat, me anë të

identifikuesve të telefonave (IMES, IMSI dhe ICC-ID) dhe vendodhjen gjeografike, duke lejuar rrjedhje të informacionit rreth identifikuesve të telefonit përmes kërkesave të thjeshta (si HTTP GET/POST), duke i gjurmuar përdoruesit përmes telefonave të tyre.

B. Telefonat e Besuar

Trusted Computing Group (TCG) [7] ka publikuar një set specifikimesh për të matur, ruajtur dhe raportuar paprekshmërinë e hardëare dhe softëare përmes një rrenjë-besimi (root-of-trust) te hardëare, e cila është 'the Trusted Platform Module (TPM)' dhe 'Core-Root-of-Trust-Measurement (CRTM)'. Në platformën me TPM të aktivizuar, CRTM mat 'bootloader'-in (programi që bën startimin e sistemit) e sistemit para se ai të ekzekutohet dhe më pas e ruan vlerën e matur një prej 'Platform Configuration Register' (PCR) Brenda TPM. 'Bootloader'-i me pas ngarkon imazhin e SO, e njëhson atë dhe e ruan përmes zgjatimit (extension) PCR dhe më pas e ekzekuton atë [326]. Me radhë, SO njëhson aplikacionet e ngarkuara dhe ruan vlerat e paprekura të tyre tek PCR-t para se t'i ekzekutoje ato.

Gjithashtu ka specifikime për platformat e telefonave të leshuara prej 'TCG Mobile Phone Working Group', p.sh. Moduli i Besimit të Telefonava ('Mobile Trusted Module'-MTM). TCG mbron përdorimin e MTM për të rritur sigurinë e smartphone-ve duke siguruar aftesitë bazë të kriptografisë, si gjenerimi i numrave të rastesishëm, tabelat hash, mbrojtja e memorjes së të dhënave delicate (p.sh. fjalekalimet), enkriptimi asimetrik, si dhe gjenerimi i nënshkrimeve. Këto akte primitive të kriptografisë mund të përdoren për të implementuar shërbimet bazike mbi sigurinë në nivel hardëare, si për shembull vërtetimi i autenticitetit të paisjes, vlerësimi i integritetit, boot-im i sigurtë. MTM paraqët një rrënjë-besim në të njëjten mënyrë si TPM për një kompjuter përsonal. Në parim MTM është një adaptim i TPM për smartphone-t e si i tillë specifikimi i tij është i njëjtë si ai i TPM, i cili lehtëson ndëropërimin nëpërmjet strukturave ekzistuese kompjuterike të besimit në kompjuterat përsonal [7].

Ka dy lloje të ndryshme për njëhsimin e paprekshmërise për çdo aplikacion binary: *kohë-ngarkimi* (load-time) dhe *matja dinamike* (dynamic measurements). TCG specifikon vetëm matjen me kohë-ngarkim, kurnjë pjesë kodi ose e dhënë njëhsohet kur

hartëzohet/ngarkohet në memorie. Matja dinamike i referohet aktitit të njëjsimit të paprekshmërisë për aplikacione kritike gjatë kohës në të cilën ata janë në punë, p.sh. kur janë në ekzekutim. Në strukturën e Arkitekturës së Njëhsimit të Paprekshmërisë (Integrity Measurement Architecture-IMA), matjet kryhen për thirrjet e ndryshme të funksioneve të sistemit kur kodi apo module të kernel-it ngarkohen para se tëekzekutohen. Pasi kodi hartëzohet në memorie dhe gjatë kohës së punës, është shumë e vështirë për të matur paprekshmërinë e procesit duke marrë në konsideratë sjelljet tepër dinamike dhe jopërcaktuese të aplikacioneve tipike, si ngarkimi i një kodi aktiv, marrja e të dhënave të jashtme apo alokimi dinamik i memories.

Në vijim të boot-imit të saktë dhe vleresimit të paprekshmërisë me kohë-ngarkim, mbrojtja e paprekshmërisë për telefonat mobil ngre kërkesa shtesë [7]:

- *Boot-im i sigurtë:* një tërësi motorrësh ndodhen në një platforme të vetme telefoni dhe afrojnë shërbime kritike dhe të nevojshme të cilat duhet të punojnë në gjëndje të mirë, p.sh. paprekshmëria e tyre duhet të verifikohet për të siguruar besimin e plote. Kështu TCG Mobile Phone Reference Architecture shpreh se boot-imi i sigurt është detyrim për MTM.
- *Boot-im i ulët dhe kohë-vonesë gjatë kohës së punës:* shumica e smartphone-vë janë të limituar në fuqi kompjuterike. Kjo kërkon që zgjidhjet e bëra për sigurinë të jenë tëpër eficente dhe vleresim i paprekshmërisë të bëhet gjatë boot-imit dhe në gjëndjen pasi ka ndodhur boot-imi të mos degradojë në përformancën dhe ekspëriencën e përdorimit së tepërmi.
- *Siguri e paprekshmërise gjatë kohës së punës:* megjithëse vleresimi i paprekshmërisë gjatë kohës së punës nuk është praktik si për PC dhe për platformat mobile, duhet të kemi një mekanizëm për të ruajtur paprekshmërinë e aplikacioneve kritike dhe resurseve përgjatë kohës së punës, p.sh. shërbimet e lidhura me telefonin dhe agjentët e menaxhimit të platformës. Si TCG dhe IMA nuk propozojnë ndonjë mekanizem për këtë qëllim.

[323, 325] flasin për një strukturë për vleresimin e paprekshmëris edhe mekanizmat e vërtetimit duke propozuar një mekanizem të boot-imit të sigurt. Mekanizmi i propozuar siguron se një platform mobile mund të boot-oje në një gjëndje të sigurt duke përcaktuar një model të pavarur rrjedhshmërie për të arritur integritet të larte të sistemit. Zgjidhja huazon mekanizmin e SELinux MAC dhe shton disa pjesë të politikave të sigurisë të SELinux. Struktura kërkon një rrënjë-besim, si MTM. [324] provon të ruajë pavarsinë e aplikacioneve kritike prej funksionaliteteve jo të sigurta me shkallë të lartë rrezikshmerie dhe të krijojë një politike të vogël të SELinux për të vlersuar paprekshmërinë e platformës mobile duke përdorur qasjen PRIMA. Politikat SELinux të përftuara lejojnë sistemin e telefonit të lidhet me pjesë te largëta dhe të mbrojë aplikacionet prej pjesëve të kodit jo të besueshëm, duke lejuar përdoruesit të instalojnë dhe përdorin aplikacionet e besuara në një mënyrë të sigurt: politika është 90% më e zvogeluar sesa ajo e referuar.

Tek [63] autorët propozojnë një qasje praktike për dizenjimin dhe implementimin e platoformës mobile të besuar. Qasja, bazuar në konceptin e platformës së besuar si një tërësi motorrësh të besuar, përcakton një metodë për pronësinë e paisjes nga përdoruesi dhe migrimin e kredencialeve të tij ndërmjet paisjeve (p.sh. portabiliteti).

[330] identifikon tre specifikime të MTM dhe jep disa zgjidhje të mundshme. E para lidhet me nevojën për të balancuar disa qëllime të cilat i kundërvihen njëra-tjetrës në dizajnin në nivel sistemi, si përformaca dhe konsumi i energjisë. Një zgjidhje e propozuar integron disa vecori të TPM drejt e në bërthamën e procesorit (processor core). Problemi i dytë lidhet me faktin se cili algoritëm kriptografik duhet të suportohet nga MTM: disa algoritma, përkatësisht RSA dhe SHAI mund të kenë si përformacë të ulët ose dobësi në siguri. Zgjidhja e përcaktuar konsideron kriptografinë eliptike si një alternativë të shëndetshme. Së fundmi, problemi i tretëlidhet me implementimin e primitivëve të kriptografisë: autorët propozojnë një zgjidhje hardware/software e cila i kundërvihet një zgjidhjeje të bazuar thjesht në hardware, e cila vuan prej fleksibilitetit të mangët [7].

Në përfundim, me një shtim tepër të shpejtë të smartphone-ve të pajisur me një numër të madh elementësh, si lidhjet dhe sensorët e shumtë, numri i kërcënimeve prej malware

është gjithashtu në rritje. Ndryshe nga ambienti i PC, për të zgjidhur problemet e sigurisë që paraqiten në smartphone na duhet të marrim parasysh disa faktorë: limitin e burimeve në dispozicion, duke përfshirë energjinë dhe kapacitetin e procesimit, numrin e madh të elementëve që mund të bëhen objekt i sulmeve, si tipet e ndryshme të lidhjeve, shërbimeve, sensorëve dhe privatësisë së përdoruesit.

KAPITULLI 7

Zgjidhja e propozuar dhe testimi i saj

7.1 Thirrjet Sistem (System Calls)

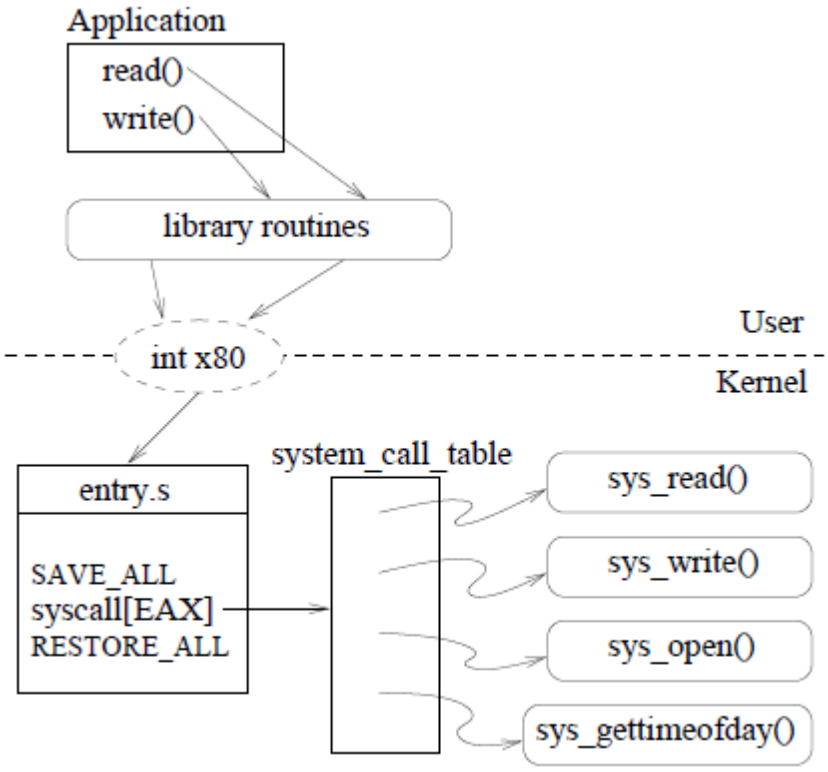


Fig. 7 Thirrjet Sistem (System Call)

Një thirrje sistem (system call) mundëson një mekanizëm për kalimin e ndarjes user/kernel në një mënyrë të kontrolluar [332]. Hapat për ta bërë këtë janë si më poshtë. Fillimisht parametrat që kërkohen nga thirrja sistem ruhen në stack ose në një regjistër të paracaktuar. Këto parametra përfshijnë parametrat e nivelit përdorues (user level), gjithashtu edhe id e system call-it që do të thirret. Pastaj proçesori kthehet në kernel mode duke përdorur një interrupt ose instruksion Trap. Menaxhuesi i interrupteve në kernel pastaj gjen menaxhuesin e duhur te thirrjeve sistem, një procedurë e cila është ruajtur në

hapësirën kernel, e bazuar në numrin e thirrjes sistem. Menaxhuesi i thirrjeve sistem fillimisht kontrollon parametrat e thirrjes për ti validuar dhe pastaj ekzekuton detyrën e tij. Cdo rrezultat kthehet mbrapsht te thirrësi në të njëjtën mënyrë si parametrat (dmth përmes stackut ose rregjistrave). Nqs ndodh një gabim, i kthehet thirrësit një numër i cili simbolizon gabimin.

Mapping midis emrave të thirrjeve sistem dhe numrave të tyre bëhet duke përcaktuar për secilin thirrje sistem *syscall_name* një vlerë unike dhe konstante e tipit *NR_syscall _name*. Në kernel të gjithë menaxhuesit janë proçedura të cilat janë të emërtuara si *sys_syscall_name*. Këto menaxhues mund të aksesohen nëpërmjet një vektori me pointera funksionesh *system_call_table* e cila është e indeksuar nga numri i thirrjes sistem. Numri aktual i thirrjeve sistem varion nga tippet e ndryshme te hardwareve por versioni që kam zgjedhur ka 255 thirrje sistem. [332]

7.2 Propozimi im

Duket patur parasysh karakteristikat e sistemeve mobile shtrohet problemi:

Si do te arrijmë të detektojme malwaret duke patur parasysh

Limitimet e paisjeve mobile? [333]

Karakteristikat e malwareve moderne?

Propozimi im është:

Realizimi një shtrese permanente në Kernelin e Sistemit Operativ Android i cili rregjistron thirrjet sistem të aplikacioneve.

Marrja dhe dërgimi i këtyre Sys Call (log files) në një server të jashtëm për përpunim.

Një skemë e thjeshtë e implementimit të propozimit tim jepet në figurën 7.2.

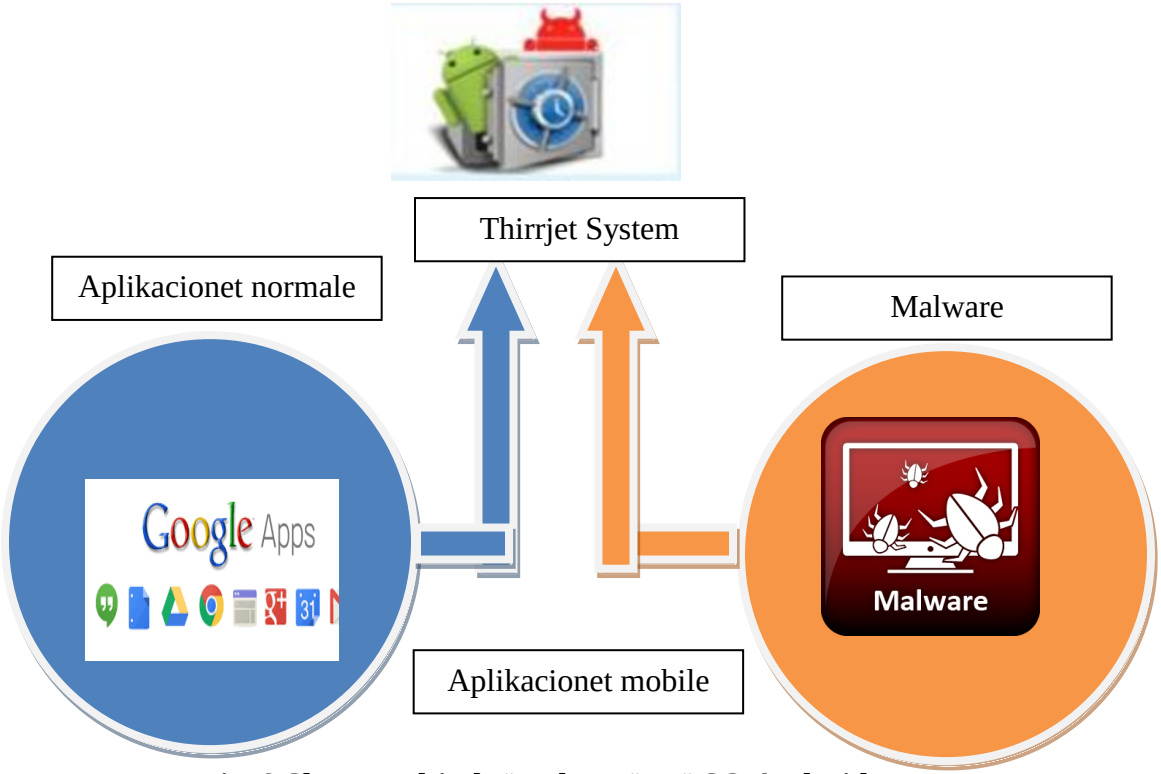


Fig. 8 Skema e thjeshtë e shtresës në SO Android

Sic tregohet edhe në fig. 8 unë kam zhvilluar një sistem i cili monitoron thirrjet sistem, i cili i mundëson përdoruesit që në mënyrë automatike të lexojë proceset që janë ekzekutuar tek SO Android. Kjo shtresë që u shtua transferon këto të dhëna në një server të jashtëm në një format si log file. Server është i progarmuar që te njohi komunikimin me pajisjen time dhe të mari për përpunim të dhënat që i vijnë. Këto përdoren pikërisht për të analizauar nësë një aplikim i caktuar ka tendenca të jetë virus apo jo.

Normalisht, siç u tregua edhe në paragrafin e mëparshëm nqs ndodh një thirrje sistem kemi :

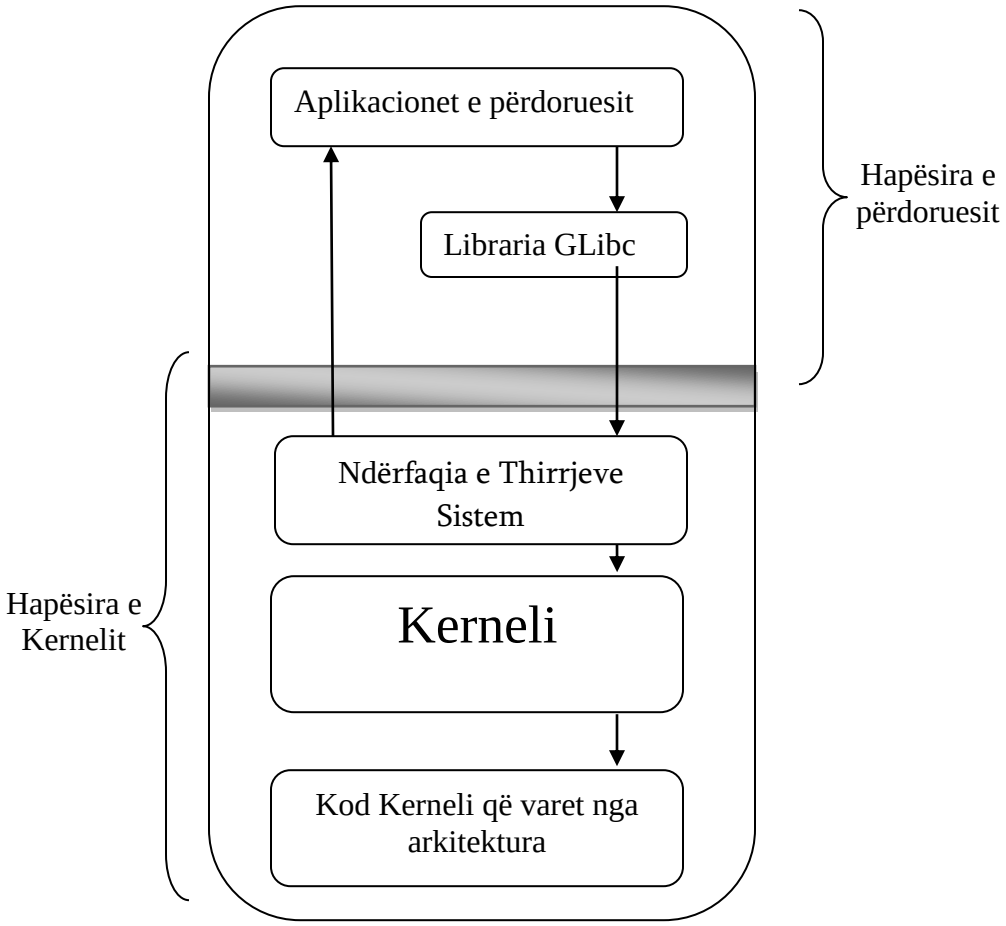


Fig. 9 Hapësira User dhe Kernel në Linux

Zgjidhja që propozoj është vendosja e një shtrese midis kernelit dhe ndërfaqes së thirrjeve sistem. Grafikisht sistemi i ri do të ketë pamjen e mëposhtme :

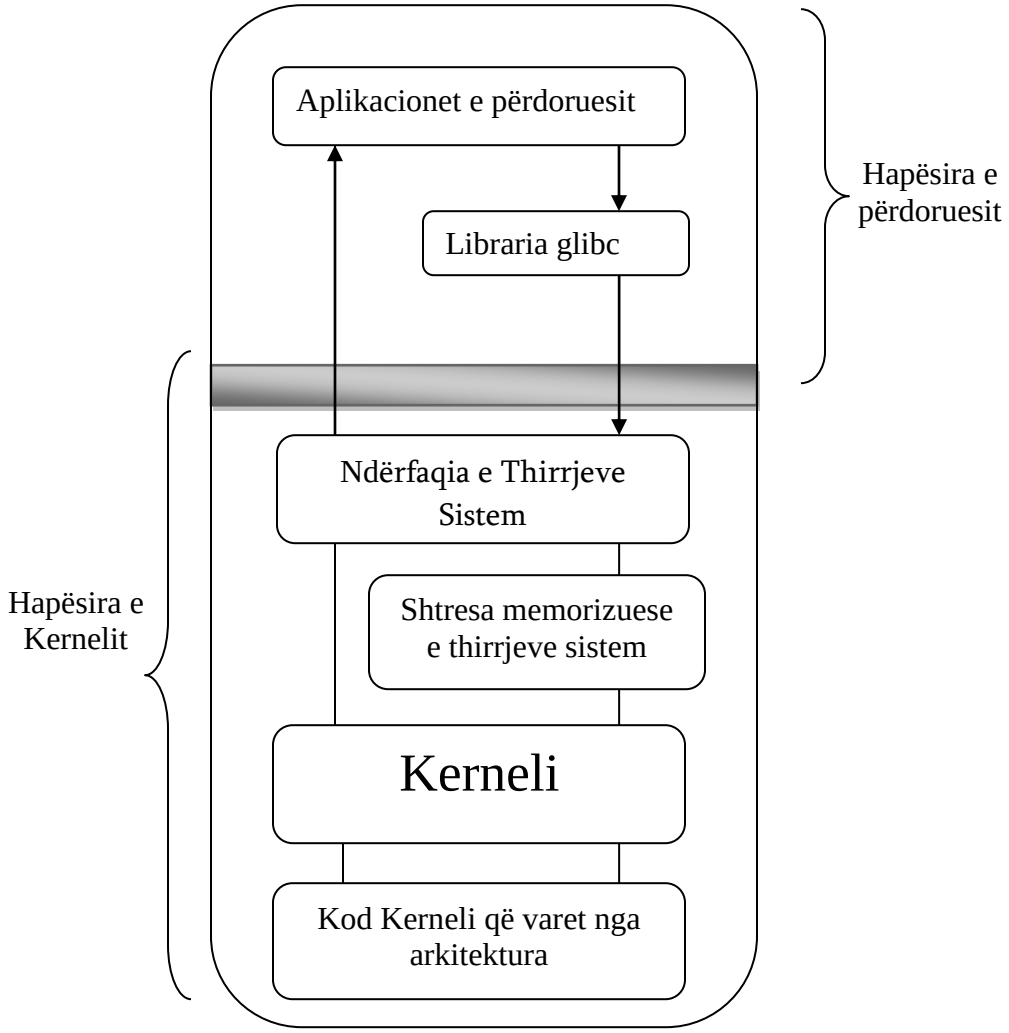


Fig. 10 Skema e zgjidhjes së propozuar

Pra sic shihet nga figura e mësipërme shtresa që unë propozoj vendoset midis ndërfaqes së thirrjeve sistem, dmth midis handlar-ave të ndryshëm, dhe kernelit. Kjo si shtresë do të ketë detyrën e marrjes, memorizimit dhe dërgimit të numrit të thirrjeve sistem në një server të jashtëm. Algoritmi për shtresën e propozuar jepet më poshtë.

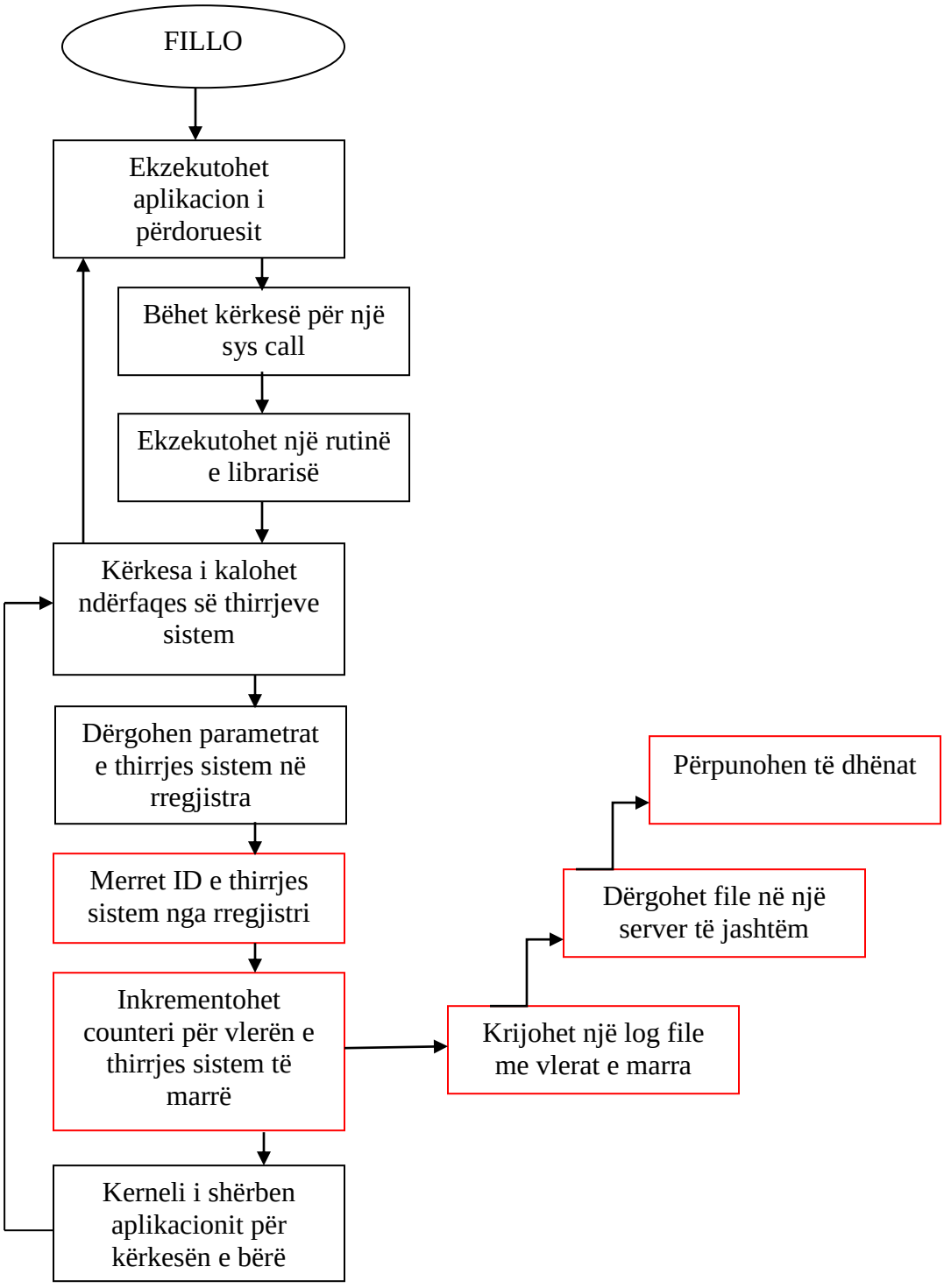


Fig. 11 Bllokskema e detajuar e funksionimit të zgjidhjes së propozuar

Siç shihet edhe nga bllokskema algoritmi im funksionon në këtë mënyrë. Mbasi fillon ekzekutimi i aplikacionit, ai për nevojat e veta të funksionimit kërkon nga kerneli shërbime të ndryshme. Gjatë proçedurave ekzekutohen disa procedura në library (glibc) e cila bën dekodimin e kërkesës së ardhur nga aplikacioni dhe pastaj bëhet një trap to kernel dmth i kalohet kompetenca kernelit dhe tani jemi në kernel mode. Ndërfaqia e thirrjeve sistem bën[338]:

- Shpëtimin e parametrave të thirrjes sistem në rregjistra në Kernel mode
- Thërret rutinat e nevojshme për ti përmbushur kërkesat e ardhura nga aplikacioni
- E nxjerr sistemin me anë të ret_from_sys_call

Zakonisht përdoren 6 rregjistra për shpëtimin e thirrjeve sistem. %eax, %ebx, %edx, %esi, %edi. Numri i thirrjes sistem ruhet tek rregjistri %eax. Pikërisht ne shfrytëzojmë këtë për të rregjistruar sa herë eshtë ekzekutuar një thirrje sistem. Programi jonë ka një counter për secilën thirrje sistem. Ai inkrementohet sa herë që një thirrje e caktuar sistem ekzekutohet. Veprimi i marrjes së thirrjes sistem nga rregjistri është thjesht një veprim copy. Kjo do të thotë që shtresa e re nuk cënon punën normale të sistemit operativ. Ky thjesht i shton ngarkesën për shak se duhet të ekzekutojë më shumë rrjeshta kodi dhe të bëjë disa veprime më shumë si kopjimi i vlerës së rregjistrit ku mbahet ID e thirrjes sistem, ekzekutimi i pjesës së kodit i cili bën shumë totalë (pra i counterit të sys call) dhe dërgimi i këtyre të dhënave nëpërmjet internetit në një server të jashtëm, në të cilin do të bëhet edhe përpunimi i të dhënave. Duke përdorur teknikën e prezantuar në [334] ne mund të ulim overhead-in e shkaktuar nga kjo shtresë, pavarsisht se ky është shumë i vogël dhe gati i papërfillshëm për aparatet moderne. Në bllokskemë pjesa e shtuar në sistemin operativ është bërë me të kuqe për ta dalluar nga blloqet normale që ka vetë sistemi operativ. Nëpërmjet përpunimit të të dhënave në serverin e jashtëm ne mund të arrijmë që të dallojmë, me një përqindje shumë të lartë nqs aplikacioni që ne po ekzekutojmë është një malware apo një aplikacion normal. Identifikimi i përdoruesve të veçantë që do të kërkojnë hyrje në serverin për përpunimin e të dhënave do të bëhet duke

përdorur teknikat e prezantuara në [335,337] Zgjedhja e thirrjeve sistem u bë pikërisht për arsyen e vetme që sado teknika të mira dhe të sofistikuara të përdori një malware për tu mos u kapur nga antiviruset apo edhe nga teknika të tjera, këto malware do tu duhet me patjetër të kalojnë nga kerneli për të ekzekutuar funkisonet e tyre të këqia. Dhe elementi të cilit ato nuk mund ti shmangen janë thirrjet sistem.

0 , 0 , 0 , 50, 58, 4 , 3 4 , 0 , 0 , 0 , 0 , 0 , 0 , 1 2 , 0 , 0 , 0 , 0 , 0 , 2 6 0 , 9 , 0 , 0 , 0 , 0 ,
1 6 4 9 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 0 , 0 , 0 , 0 , 5 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 2 2 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 3 4 6 6 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 2 , 0 , 0 , 0 , 0 ,
1 3 2 , 0 , 0 , 0 , 0 , 0 , 0 , 4 0 , 4 1 , 0 , 0 , 0 , 0 , 0 , 7 6 , 0 , 0 , 0 0 , 0 , 0 , 4 , 0 , 8 7 , 1 7 , 0 , .

Figure 7.7 Output-i i shtresës së re

File që del nga kjo shtresë është siç tregohet në figurën 7.7. Cdo numër, i ndarë me presje përfaqëson, sa kërkime/ekzekutime janë bërë nga një aplikacion specifik, gjatë proçesit të monitorimit Psh system call open() është përdorur 50 herë, close() 58 herë etj

Më poshtë do të hyjmë edhe më në detajë në lidhje me zgjidhjen e propozuar.

Thirrjet system janë të shumta dhe ato janë specifike për tipin e linuxit që përdorim. Linux njeh dy klasa ngjarjesh: përjashtimet dhe interruptet. Të dyja shkaktojnë një contex switch në një procedurë të re. Interruptet mund të ndodhin në një kohë të papërcaktuar gjatë ekzekutimit të një programi dhe përdoren për tju përgjigjur sinjaleve të hardware-it. Përjashtimet janë të shkaktuara nga ekzekutimet e instruksioneve. Dy burime interruptesh njihen nga i386 : Maskable interrupts and Nonmaskable interrupts. Dy tipe përjashtimesh njihen nga i386: përjashtimet e detektuara nga proçesori dhe përjashtimet e programuara. Cdo interrupt ose përjashtim ka një numër. NMI dhe përjashtimet e dedektuara nga procesori i janë caktuar vektorë në intervalin 0-31. Vektorët për maskable interrupts përcaktohen nga hardware. Kontrollera të jashtëm të interrupteve vendosin vektorin në bus. Çdo vektor në range 32-255 mund të përdoret për maskable interrupts dhe përjashtime të programuara. Këtu është një listë me interruptet dhe përjashtimet [338].

Tabela 8 Numrat e interrupteve dhe përjashtimeve

0	divide error
1	debug exception
2	NMI interrupt
3	Breakpoint
4	INTO-detected Overflow
5	BOUND range exceeded
6	Invalid opcode
7	coprocessor not available
8	double fault
9	coprocessor segment overrun
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	reserved
16	coprocessor error
17-31	reserved
32-255	maskable interrupts

Prioriteti i në rast se kemi interrupted dhe përjashtime në të njëjtën kohë është:

Në linux/kernel të android ekzekutimi i një thirrje system bëhet nga një maskable interrupt ose exception class transfer, e cila shkaktohet nga instruksioni int 0x80. Ky vektor përdoret për ti kaluar kontrollin kernelit. Ky interrupt inicializohet gjatë ndezjes së sistemit sëbashku me vektorë të tjerë si psh vektori i orës së sistemit. Sistemi që kemi zhvilluar identifikon klasen e sistemit dhe rrit performancën e sistemit në mënyrë automatike. Për të realizuar këtë ai përdor support vechtor machine për të identifikuar klasën e sistemit. [336]
Linux/kernel i android që kam përzgjedhur ka 255 thirrje system. Kur një nga aplikacionet kërkon një thirrje sistem ndodhin hapat e mëposhtme [338]

- Çdo thirrje futet në një vector në librarinë *libc.* Çdo thirrje në librarinë libc është përgjithësisht syscallX() macro, ku X është numri i parametrave i cili përdoret nga rutina aktuale. Disa thirrje sistem janë më shumë komplekse sesa disa të tjera sepse disa kanë gjatësi variablash më të madhe, por edhe në këtë gjëndje këto thirrje sistem duhet të kenë të njëjtën pikë hyrje, ato thjesht kanë vonesa më të mëdha për shkak të sasisë së parametrave. Shembuj të thirrjeve sistem komplekse janë open() dhe write().
- Cdo makro e thirrjeve sistem hapet në një rutinë assembler, e cila përgatit Stack frame dhe thërret *sys_call()* me anë të një interrupti me anë të instruksionit int $0x80

Psh thirrja sistem setuid kodohet

```
_syscall1(int,setuid,uid_t,uid);
```

E cila hapet si

```
_setuid:
  subl $4,%exp
  pushl %ebx
  movzwl 12(%esp),%eax
  movl %eax,4(%esp)
  movl $23,%eax
  movl 4(%esp),%ebx
  int $0x80
```

```
    movl %eax,%edx
    testl %edx,%edx
    jge L2
    negl %edx
    movl %edx,_errno
    movl $-1,%eax
    popl %ebx
    addl $4,%esp
    ret
L2:
    movl %edx,%eax
    popl %ebx
    addl $4,%esp
    ret
```

Në këtë pikë asnjë kod i sistemit nuk është ekzekutuar. Jo derisa ekzekutohet *int $0x80* e cila bën thirrjen për hyrjen në Kernel _system_call(). Kjo pikë hyrje është e njëjtë për të gjithë thirrjet sistem. Kjo realizon shpëtimin e regjistrave, kontrollon nëse një thirrje sistem e vlefshme është duke u thirrur dhe në fund kalimi i kontrollit te kodi i thirrjes sistem me anë të offsetit *sys_call_table.* Ndërfaqia ka përgjegjësi për thirrjen e *_ret_from_sys_call()* kur thirrja sistem ka mbaruar, por përpara se të shkojë në hapësirën e përdoruesit.

Mbasi thirrja sistem është ekzekutuar, thirret *_ret_from_sys_call().* Ai kontrollon nqs duhet te startojë schedulerin , apo jo.

Mbasi bëhet kthimi nga thirrja sistem, makro *syscallX()* kontrollon për një vlerë të kthyer negative, dhe nqs ka një të tillë, vendos vlerën positive në variablin global *_errno,* kështu që ai mund të aksesohet nga *perror().* Makro për një thirrje sistem është [338]:

```
#define _syscall1(type,name,type1,arg1) \
     type name(type1 arg1) \
     { \
     long __res; \
     __asm__ volatile ("int $0x80" \
          : "=a" (__res) \
          : "0" (__NR_##name),"b" ((long)(arg1))); \
```

```
if (__res >= 0) \
     return (type) __res; \
errno = -__res; \
return -1; \
}
```

Kur zgjerohet ky bëhet në formën e mëposhtme:

```
int setuid(uid_t uid)
{
     long __res;
     __asm__ volatile ("int $0x80" \
          : "=a" (__res) \
          : "0" (__NR_setuid), "b" ((long)(uid)));
     if (__res >= 0 )
          return (int) __res;
     errno = -__res;
     return -1;
}
```

Ktu shikohet sesi kodi i pastrimit konvertohet në assembler.

“=a” (__res) do të thotë që rezultati që kthehet është %eax

“0” (__NR_setuid) do të thotë vendos numrin e thirrjes sistem në %eax në hyrje

“b” ((long) (uid) do të thotë vendos argumetin e parë në %abx në hyrje

7.3 Testimi i zgjidhjes së propozuar

Në këtë seksion do të testojmë zgjidhjen e propozuar. Kjo do të thotë që do të testojmë nqs nëpërmjet kësaj teknike/shtrese arrihet të detektohen programet me qëllime të këqia (malwaret). Siç tregova edhe në kapitullin e 5, tipologjia e

Malwareve është shumë e gjërë dhe me qëllime të ndryshme. Unë synoj që nëpërmjet kësaj teknike të arrihet të identifikohen malwaret e pjesës më të madhe të familjeve që janë shfaqur deri tani, gjithashtu edhe të atyre që mund të shfaqen në të ardhmen. Për testimin e kësaj shtrese kam përzgjedhur disa malware nga më të rrezikshmit, të cilat janë kapur dhe detektuar. Këto informacione janë marre nga Threat reports (raportet e rreziqeve) të antiviruseve kryesore si Symantech, McAfee dhe gjithashtu edhe nga ekpertiza e Google në lidhje me malwaret që i janë shfaqur në marketin e tij. Në seksionet e mëposhtme do të proçedoj në këtë mënyrë. Në fillim do të tregoj karakteristikat familjes së malware-it, të cilin e kam marrë për të bërë eksperimetin. Pastaj do të tregoj rrezultatet e eksperimentit dmth numrin e thirrjeve sistem si për aplikacionin normal i cili është marrë nga marketi zyrtar i Google, ashtu edhe për aplikacionin e virusuar. Për secilin program do të bëhet përzgjedhja e një set-I thirrjesh sistem për shkak të numrit të lartë të tyre, dhe në fund do të bëhet krahasimi. Kjo do të nxjerri edhe efektshmërinë e propozimit tim.

7.3.1 AnserverBot Malware

Sic tregohet në detaje tek [328], NetQin Security Research Center identifikojë një Android Trojan i quajtur Anserverbot, i cili është konsideruar deri tani një nga viruset më të sofistikuara në sistemet Android. Ky virus trasmetohet nga programe legjitime dhe po shpërndahet nga disa markete të Android në Kinë. Ky Trojan është i famshëm për disa nga cilësitë e tij. Psh. Ai përdor disa teknika të sofistikuara për të shmangur kapjen dhe analizimin duke përfshirë Plankton code loading, invokimin e metodave të Java, obskurbim i kodit dhe enkriptim i të dhënave, vetë verifikim i firmave, gjithashtu edhe heqja e programeve të sigurisë së pajisjeve mobile. Për më tepër trojani ka disa funksione që realizojnë botnet. Është nga rastet e rralla që të gjitha këto teknika dashakeqëse integrohen bashkarisht në të njëjtin virus. Analiza e këtij virusi është bërë mbi një kopje reale. Vlera e SHA1 dhe payload tregohen më poshtë.

Sample : 002f537027830303e2205dd0a6106cb1b79fa704 Payload A : 34dac9fd5938389a94ea2a3450f1bcea6a7710b1 Payload B : ade014d299e691dcedf764822d4b52c9eb4605a2

Fig. 12 Vlera e SHA1 e trojanit dhe 2 payloade-et

Si një program Trojan, AnserverBot transportohet me anë të aplikacioneve legjitime. Në nivelet e larta, ai ripaketohet në një aplikacion transportues bashkë me dy aplikacione të fshehura (në direktorinë assets/) me emra anservera.db dhe asserverb.db. Për thjeshtësi i quaj anservera.db dhe anserverb.db si payload A dhe payload B. këto dy aplikacione kanë të njëjtën emër pakete *com.sec.android.touchScreen.server*

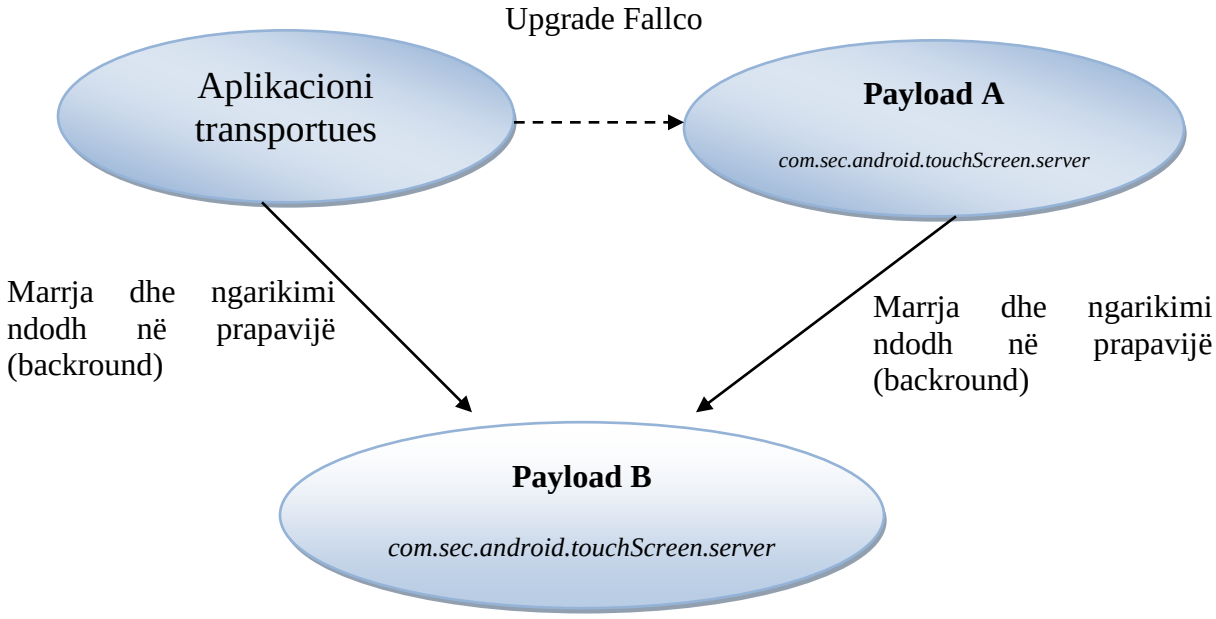


Fig. 13 Pamje e një niveli të lartë e tre aplikacioneve në AnserverBot [328]

Por me funksione të ndryshme. Specifikisht kur ekzekutohet aplikacioni transportues, ai do të kërkojë një upgrade fallso, në mënyrë që të shtyje përdoruesin që të instalojë payload A. payload A është kryesisht një program bot I cili ekzekutohet në mënyrë të qetë në prapavijë i cili nuk nxjerr asnjë ikonë në desktop kur instalohet. Gjatë ekzekutimit si aplikacioni host si payload A mund të ngarkojnë në mënyrë dinamikë dhe të ekzekutojnë payload B në Dalvik machine pa e instaluar atë. Përdorimi i mekanizmave Java reflection dhe fshehja e emrave të metodave e vështirësojnë shumë kapjen e virusit. Gjithashtu aplikacioni host dhe payload A mund të kërkojnë upgrade e payload B, kurse payload B mund të komunikojë me një server të jashtëm për të ekzekutuar komanda të tjera. Analiza të mëtejshme tregojnë se kodi shtesë që meret nga serveri dyfishon funksionet e payload A, gjë e cila prezumon rritjen e "fuqisë" së virusit, në mënyrë që ai të ekzekutohet edhe kur aplikacioni transportues të hiqet nga telefoni.

Kur AnserverBot ripaketohet në një aplikacion ekzistues ai do të kërkojë disa leje shtesë. Në shembujt e mëposhtëm kam marë një aplikacion të një loje. Figura 7.11 dhe 7.12 tregojnë lejet që kërkohen nga aplikacioni normal (i pastër) dhe aplikacioni i ripaketuar (i infektuar). Qartësisht versioni i ripaketuar kërkon shumë më shumë leje, duke përfshirë disa shumë të rrezikshme si SEND SMS, RECEIVE SMS, CALL PHONE, RESTART PACKAGE and READ LOGS

```
<uses-permission android:name="android.permission.INTERNET" />
```

Fig. 14 Leja e kërkuar nga aplikacioni origjinal

Përvec se kërkon më shumë leje, virusi shton edhe disa komponentë në aplikacionin host. Në figurën 14 tregohet ky marrës specifik dhe ngjarjet që ndodhin në aplikacionin e infektuar. Duke u bazuar tek aplikacioni origjinal, ai i infektuari shton dy module të reja : com.android.view.custom dhe com.sec.android.providers.drm. i pari përmban rutinat për të aksesuar Payload B. I dyti është një klinet bot i cili lidhet me serverin për të shkarkuar dhe instaluar Payload B. Gjithashtu ai i infektuari ka dy file asnservera.db dhe anserverb.db në direktorinë assets/. Nga prapashtesa e tyre .db ato meren si file database. E vërteta është që ato janë dy aplikacione Android: i pari do të instalohet sapo aplikacioni

host te ekzekutohet dhe i dyti do të ngarkohet në mënyrë dinamike për tu ekzekutuar pa instalim

```
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RESTART_PACKAGES" />
<uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
<uses-permission android:name="android.permission.READ_LOGS" />
```

Fig. 15 Lejet që kërkohen nga aplikacioni i ripaketuar

```
<receiver android:name="com.android.view.custom.BaseABroadcastReceiver">
    <intent-filter android:priority="2147483647">
        <action android:name="android.net.wifi.PICK_WIFI_WORK" />
        <action android:name="android.net.conn.MEDIA_NOFS" />
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
        <action android:name="android.intent.action.USER_PRESENT" />
        <action android:name="android.intent.action.UMS_CONNECTED" />
        <action android:name="android.intent.action.UMS_DISCONNECTED" />
    </intent-filter>
</receiver>
```

Fig. 16 Lejet që kërkohen nga aplikacioni origjinal

Kur inicializohet AnserverBot do të kontrollojë nqs payload A është aktualisht I instaluar. Nqs jo, ai do të shfaqë një tabelë e cila kërkon një upgrade fallco, e cila e mashtron përdoruesin për të bërë një upgrade, I cili nuk ekziston. Nqs përdoruesi I jep leje që të bëhet upgrade, payload A do të instalohet në telefon. Mbas instalimit payload A nuk tregon asnje ikonë në desktop. Por ai punon në prapavijë në mënyrë të fshehtë. Funksioni kryesor është shtimi I disa rrjeshtave kod aplikacionit host. Si rrezultat duke instaluar payload A, virusi sigurohet që edhe nëse aplikacioni transportues hiqet nga telefoni, aim und të vazhdojë të ekzekutohet në telefon. Kodi i cili është përgjegjës për kontrollin e

prezencës dhe instalimin e payload A gjendet në payload B. Aplikacioni host mund të thërrasë metoda në payload B dhe këto metoda përdoren në mënyrë indirekte për të instaluar payload A. Duhet patur parasysh që mënyra normale për të kontrolluar prezencën e një aplikacioni të vecantë është të marrësh listën e aplikacioneve të instaluara dhe ta kontrollosh atë për të gjetur aplikacionin. Gjithsesi AnserverBot përdor një mënyrë tjetër. Ai fillimisht merr PackageContex duke përdorur emrin e paketës të payload A si argument dhe merr *sharedPreference* nga PackageContex. Nqs payload A nuk është instaluar, aksesi te sharedPreference do të bejë një përjashtim. Në këtë mënyrë AnserverBot do të kuptojë që payload A nuk është instaluar dhe pastaj tenton të instalojë këtë payload.

Fig. 17 Screenshoti dhe kërkesa për lejet nga upgrade i ri (Payload A)

Procesi i instalimit të payload A nuk është i komplikuar. Specifikisht AnserverBot në fillim tregon upgrade-in fallco përdoruesit. Nqs përdoruesi shtyp butonin upgrade, ai do

të shpëtojë pathin e payload A dhe do ta dërgojë te packageinstaller për instalim. Kodi për instalimin e payload A ndodhet në fuksionin handleMessage të klasës com.sec.android.providers.drm.Charset në payload B. Fig 18 tregon një makinë gjëndjesh për instalimin e payload A.

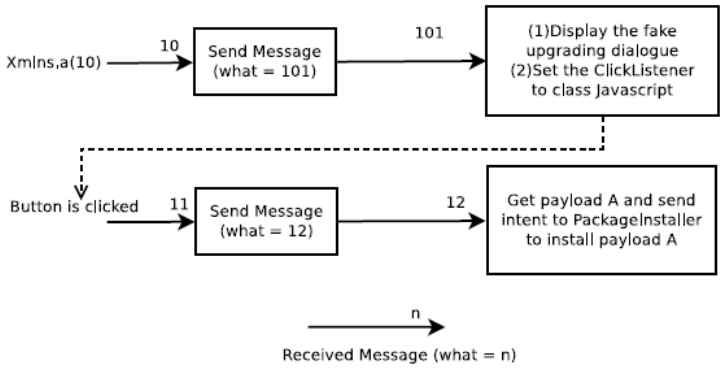


Fig. 18 Makina e gjëndjeve për instalimin e payload A

Ambjenti i eksperimentit

Duke marrë aplikacionet normale dhe gjithashtu duke marrë edhe aplikacioni e virusuar me këtë malware kam bërë eksperimentet e mëposhtme. Ambjenti i ekseprimentit është një SO android, RAM 2GB, HDD 16G Samsusng smartphone. Duke i vendosur shtresën e re këtij smarphone ky do të na gjenerojë log file me system calls dhe do ti dërgojë në një server të jashtëm. Lidhja me serverin është e tipit ftp. Server është i përgatitur të njohë këtë komunikim. Ai është një kompjuter me intel core i3, RAM 2GB hdd 500GB. Eksperimentet do të zhvillohen për sasi të ndryshme ciklesh ekzekutimi.

Eksperimentet

Duke marë dhe instaluar aplikacionin normal Voice SMS  , i cili është marrë nga Android Market ne kemi arritur rrezultatet e mëposhtme.

Tabela 9 Numri ekzekutimeve të thirrjeve sistem për aplikacionin zyrtar

Sys. call / Koha	Read ()	Write()	Fork ()	Link ()	Kill ()	Msgget()
10 sek	10 herë	9 herë	20 herë	12 herë	8 herë	2 herë
70 sek	50 herë	40 herë	35 herë	61 herë	22 herë	15 herë
100 sek	106 herë	92 herë	51 herë	95 herë	54 herë	56 herë

Ekzekutimi i këtij programi është bërë me mbi 100 sekonda. Thirrjet sistem të zgjedhura janë Read () Write () Link() Kill() dhe msgget()

Grafikisht do të kishim.

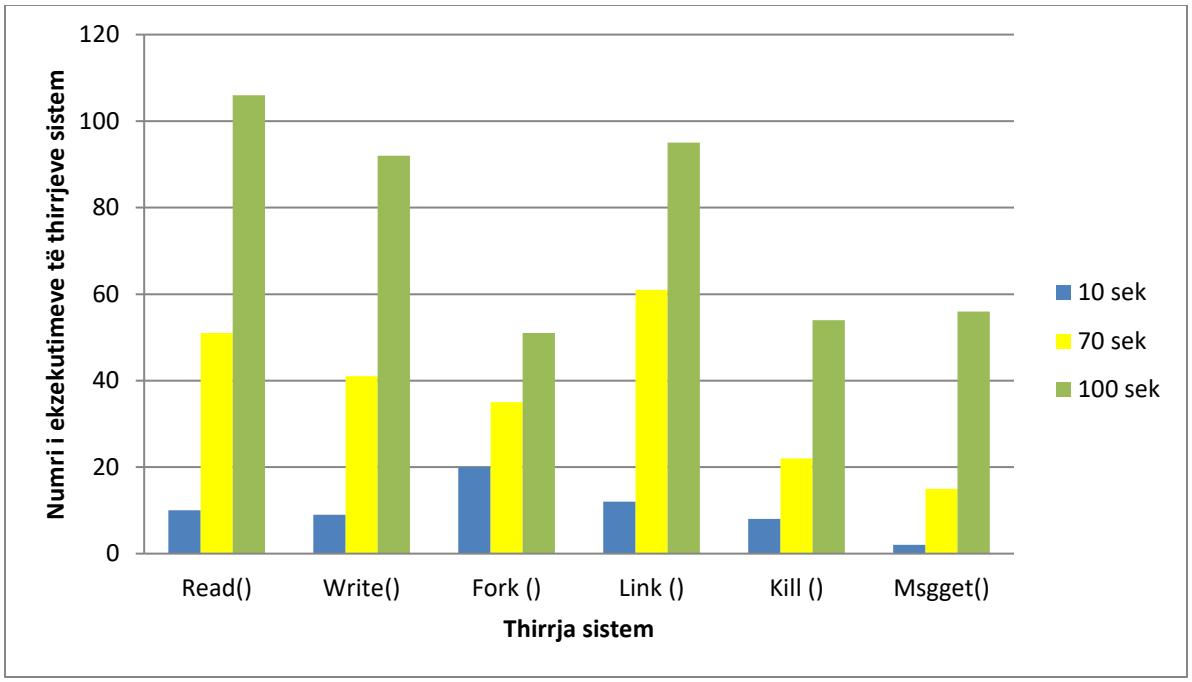


Fig. 19 Grafiku i sys call-ve të applikacionit zyrtar

Tani kemi marrë apliakcionin e virusuar me anserverBot dhe e instalojmë dhe e ekzekutojnë. Ai na kërkon të bëjmë një update dhe mbasi ja bëjmë këtë updatë marrim rrezultatet e mëposhte

Tabela 10 Numri i thirrjeve sistem për aplikacionin e virusuar

Sys. call / Koha	Read ()	Write()	Fork ()	Link ()	Kill ()	Msgget()
10 sek	104 herë	91 herë	207 herë	173 herë	92 herë	98 herë
70 sek	274 herë	251 herë	302 herë	215 herë	184 herë	175 herë
100 sek	106 herë	392 herë	51 herë	95 herë	226 herë	219 herë

Grafikisht kemi

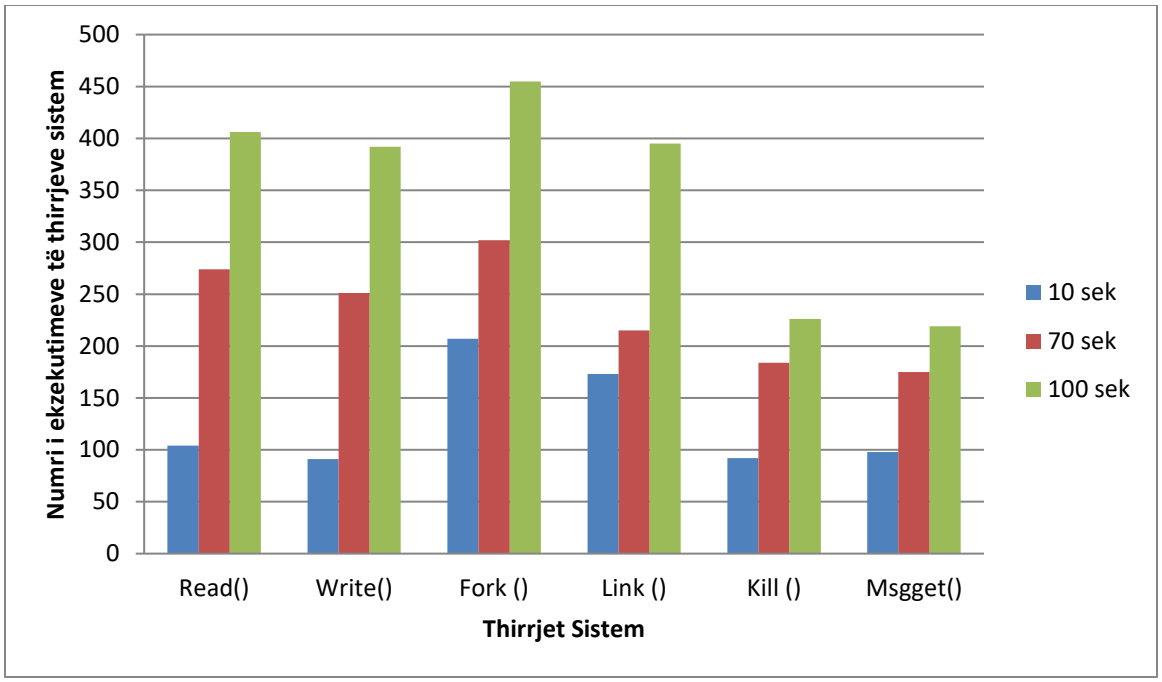


Fig. 20 Grafiku i thirrjeve sistem të applikacionit të virusuar

Normalisht ne bëmë mbledhjen e këtyre të dhënave që të kemi mundësinë e bërjes së krahasimeve dhe mbas këtyre të arrijmë në konluzionet përkatëse. Nqs bëjmë një krahasim midis appikacionit normal dhe atij të virusuar kemi grafikun e mëposhtëm.

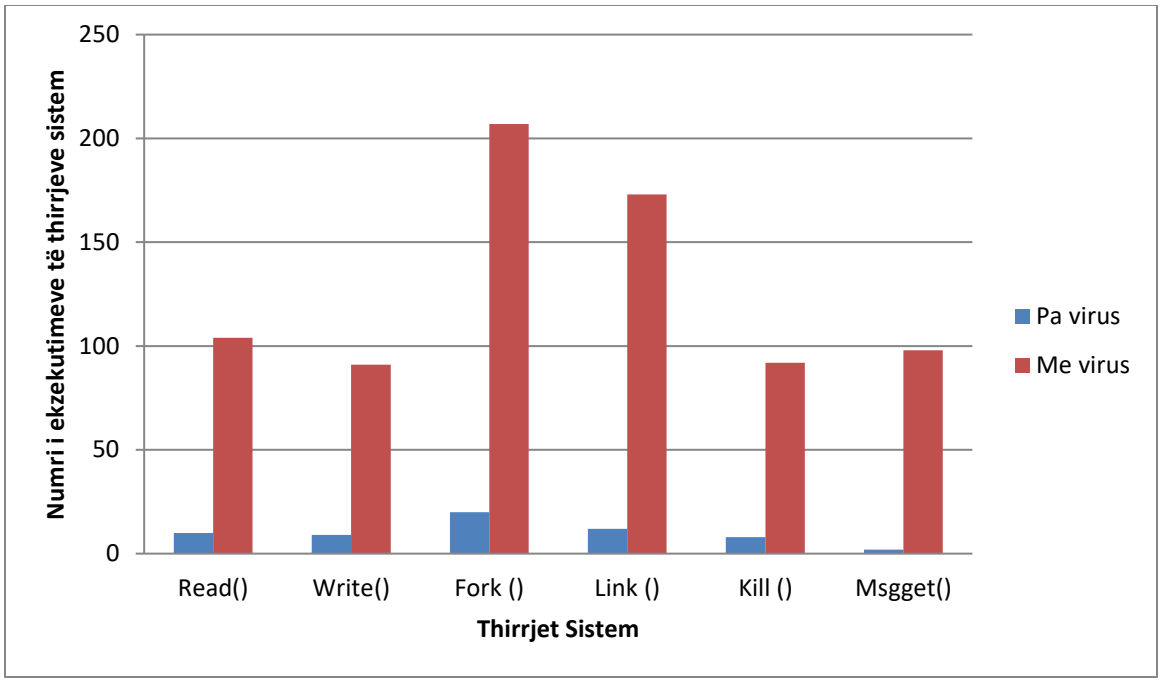


Fig. 21 Grafiku i krahasimit të numrit të thirrjeve sistem për 10 sekonda ekzekutimi

Kurse për 100 sekinda ekzekutim kemi grafikun e mëposhtëm.

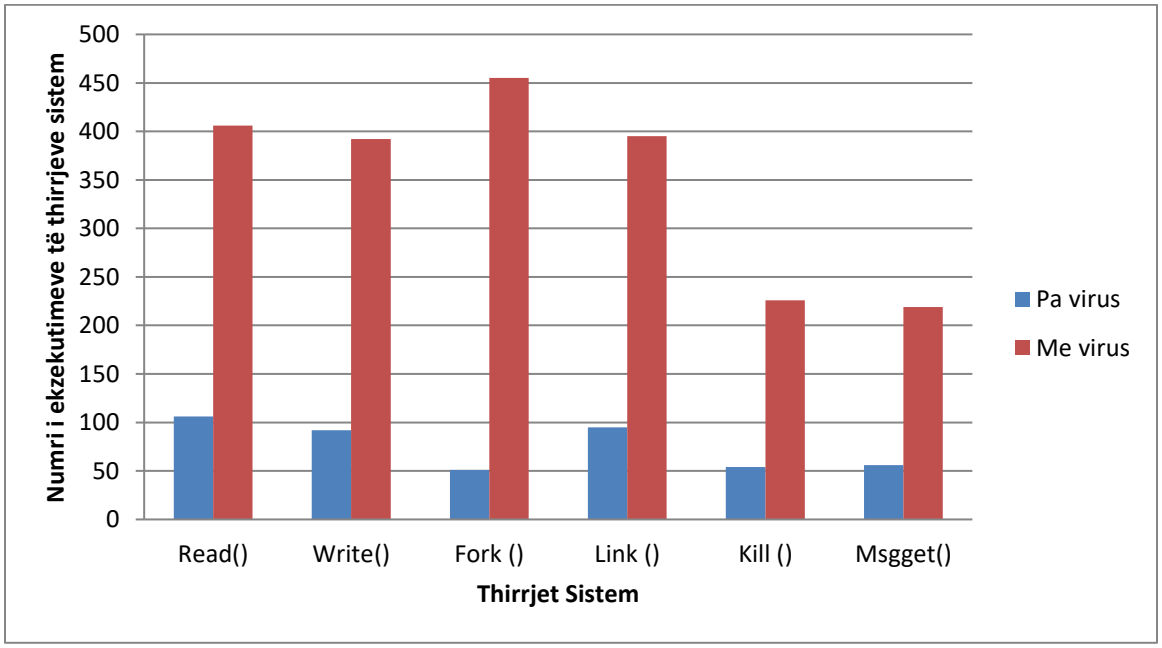


Fig. 22 Grafiku i krahasimit të numrit të thirrjeve sistem për 100 sec ekzekutimi

Pra siç shihet qartë edhe nga grafikët e mësipërm diferenca e përdorimit të thirrjeve sistem nga aplikacioni i virusuar është shumë e madhe. Ajo vizualisjt shikohet me sy të lirë, pa patur nevojën e modeleve të komplikuara matematike. Në këto eksperimente shikohet se teknika jonë për marrjen e thirrjeve sistem dhe analizimi i tyre na çon në dëtektimin e këtij virusi. Në seksionet e mëposhtme do të realizoj prova me malware të familjeve të tjera për të arritur një konkluzion më të përgjithshëm.

7.3.2 Basebridge Malware

Siç tregohet tek [8] një ndër viruset që godet platformën Andorid është edhe BridgeBase. Malware-i gjendet në kopje të infektuara të aplikacioneve të famshme Andorid si QQ Doudizhu, Drag Racing, Trader, Donkey Jump, Jungle Monkey, dhe Gold Minor dhe shumë të tjerë. Ai lehtësisht mund t'i bashkangjitet aplikacioneve legjitime. Kur një aplikacion i infektuar instalohet, malware-i do t'i kërkojë përdoruesit që ta përmirsojë (upgrade) atë. Në qoftë se përdoruesi zgjedh ta bëjë këtë veprim, ai do të instalohet vetë në një zone tjetër të telefonit me emrin "com.android.battery". Pas instalimit, një tjetër dritare do t'i kërkojë përdoruesit ta ristartojë aplikacionin në mënyrë që ta vendosë në punë. Sapo aplikacioni ristartohet, malware-i aktivizohet. Në aktivizim, malware-i do të aktivizojë tre shërbime të këqia: AdSmsService, BridgeProvider dhe PhoneService , për të komunikuar me një server kontrolli, nga ku do të shkarkojë një listë konfigurimesh për të lexuar informacionet që kërkohen, telefonatat e thirrura ose SMS e dërguara, duke shkaktuar trafim të përdoruesit. Ndërkohë, malware-i gjithashtu bllokon mesazhet në mënyrë që përdoruesi mos jetë në gjendje ë të marrë SMS në lidhje me harxhimet që ka kryer, kështu që të gjitha aktivitetet e këqia kryehen pa dijeninë e përdoruesit. Malware-i gjithashtu mund të fuse mesazhe në inbox-in e telefonit. Pas gjithë DroidDream-ve që Google-it iu desht të hiqte nga marketi zyrtar, duket se iu desh ta bënte përsëri këtë punë. Si zakonisht, nuk ka një mënyrë 100% të sigurtë për të parandaluar viruset – vetëm mënyra të zgjuara të përdorimit të telefonit veçanërisht në qoftë se duam të shkarkojmë aplikacione. Duhet të mundohuni që gjithmonë ti ruajmë të dhënat tona diku tjetër në mënyrë që, nëse do tna duhet ti bëjmë telefonit restore, të mos i humbasim këto të dhëna.

Në instalim, Trojan:Andorid/BaseBridge.A shfaq një mesazh të rremë që përdoruesi të lejojë që të instalohet një ‘update’[327].

Trojani kërkon privilegjet e mëposhtme:

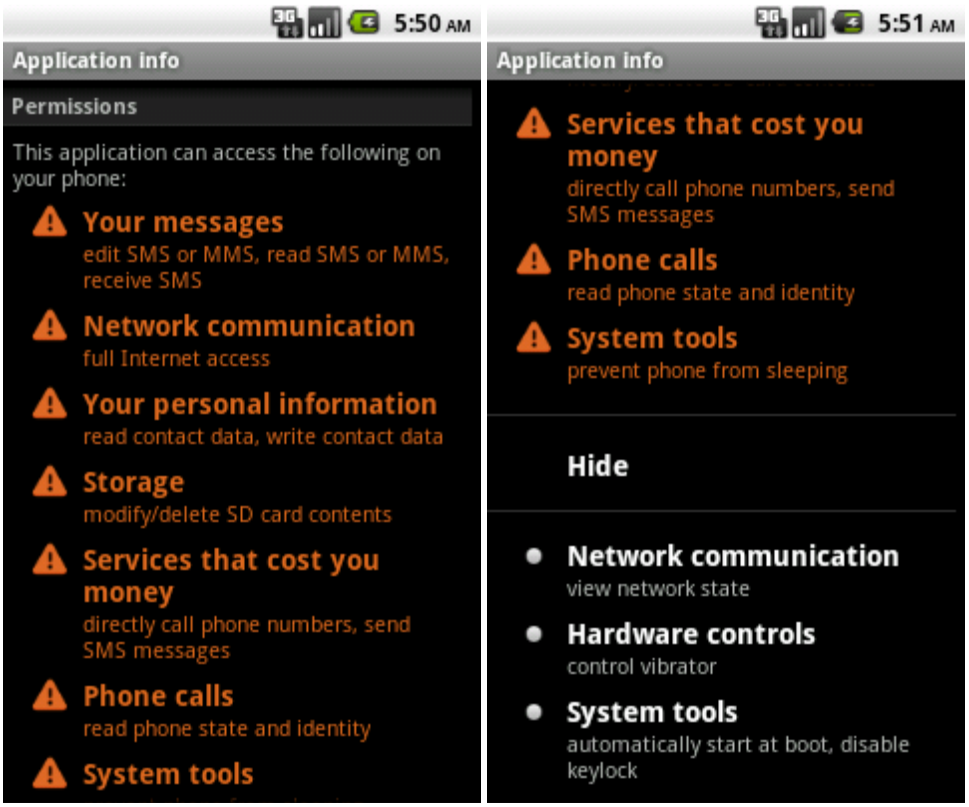


Fig. 23 Privilegjet që kërkon Basebridge [327]

Përdoruesi në mënyrë aktive lejon proçesin e përditësimit të fillojë. Më pas kërkohet një ristartim i pajisjes. Sapo ristartohet, ky Trojan instalohet me sukses me emrin “com.android.battery”. [327]

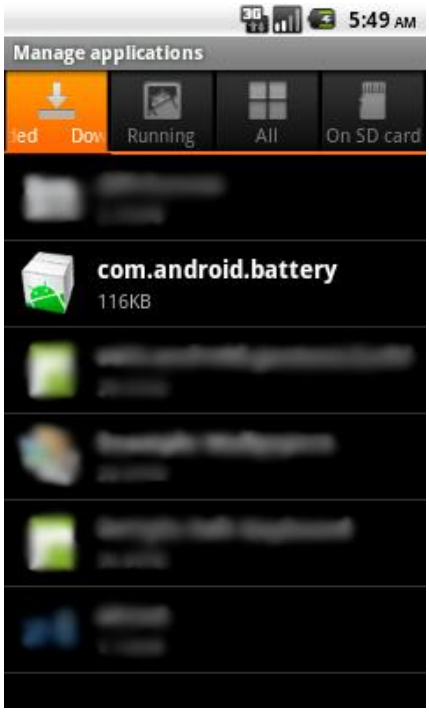


Fig. 24 Instalimi i trojanit në smartphone[327]

Sapo instalohet në pajisje, trojani do të vendosë në punë një ose disa nga shërbimet dashakeqe në background [327]:

- AdSmsService
- BridgeProvider
- PhoneService
- ZlPhoneService
- BaseBroadcastReceiver

Këto shërbime dhe leje të garantuara të trojanit i lejojnë atij të marrë informacione si më Poshtë[327]:

- Informacione të pajisjes (Modeli, prodhuesi, etj.)
- IMSI (International Mobile Subscriber Identity)
- Përmbajtjen e SMS-ve
- Thirrjet telefonike

Këto informacione mëpas dërgohen në një server në distance. Çdo pagesë që mund të kërkohet për të kryer këto aktivitete ju faturohen përdoruesit.

Rrezultatet e eksperimentit

Aplikacioni i përzgjedhur për këtë eksperiment është Drag Raicing. Mbas instalimit dhe ekzekutimit të aplikacionit zyrtar kemi rrezultatet e mëposhtme. Seti – i zgjedhur i thirrjeve sistem është read () open () access chmod() chown()

Tabela 11 Numri i thirrjeve sistem për aplikacionin zyrtar

Sys call / **Koha**	**Read ()**	**Open ()**	**Access ()**	**Chmod ()**	**Chown()**
10 sec	111 herë	80 herë	175 herë	147 herë	139 herë
50 sec	569 herë	445 herë	402 herë	605 herë	598 herë
70 sec	889 herë	695 herë	754 herë	805 herë	850 herë
100 sec	1325 herë	1229 herë	1058 herë	1205 herë	1124 herë

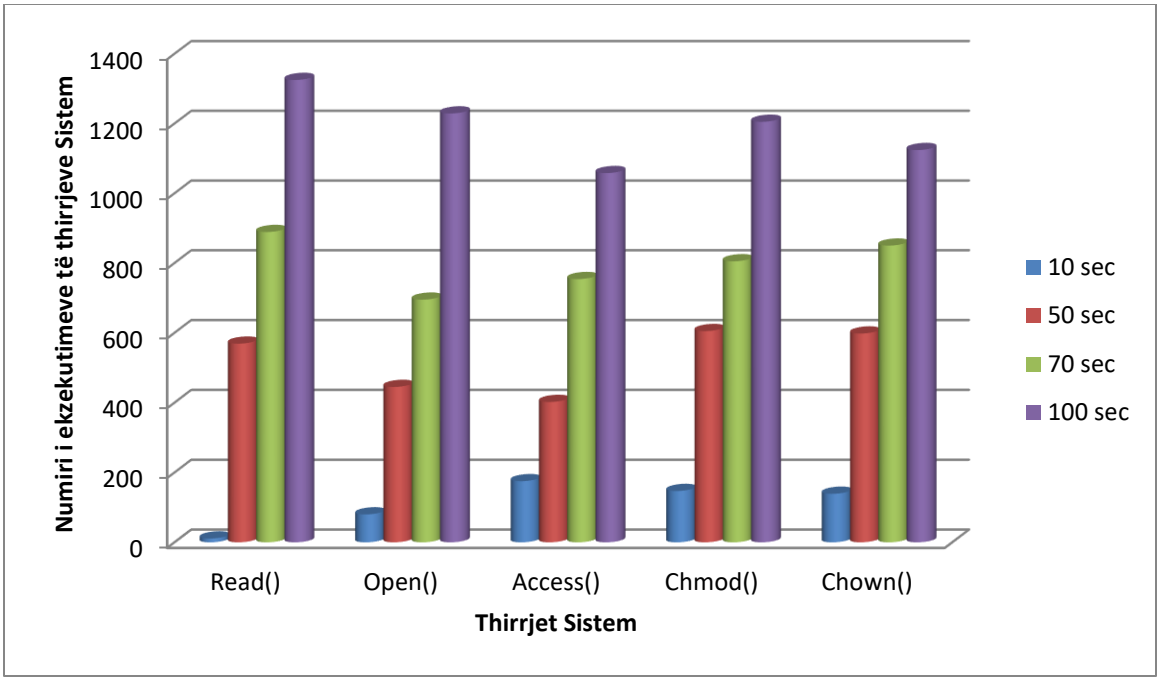


Fig. 25 Grafiku i thirrjeve sistem të applikacionit të zyrtar

Rrezultatet e mara mbas instalimit të aplikacionit të virusuar janë paraqitur në tabelën e mëposhtmë.

Tabela 12 Numri i thirrjeve sistem për aplikacionin e virusuar

Sys call / Koha	Read ()	Open ()	Access ()	Chmod ()	Chown()
10 sec	425 herë	254 herë	408 herë	560 herë	268 herë
50 sec	1500 herë	1445 herë	1205 herë	1325 herë	1169 herë
70 sec	2258 herë	2697 herë	1987 herë	2146 herë	2014 herë
100 sec	2998 herë	3158 herë	2687 herë	2458 herë	2985 herë

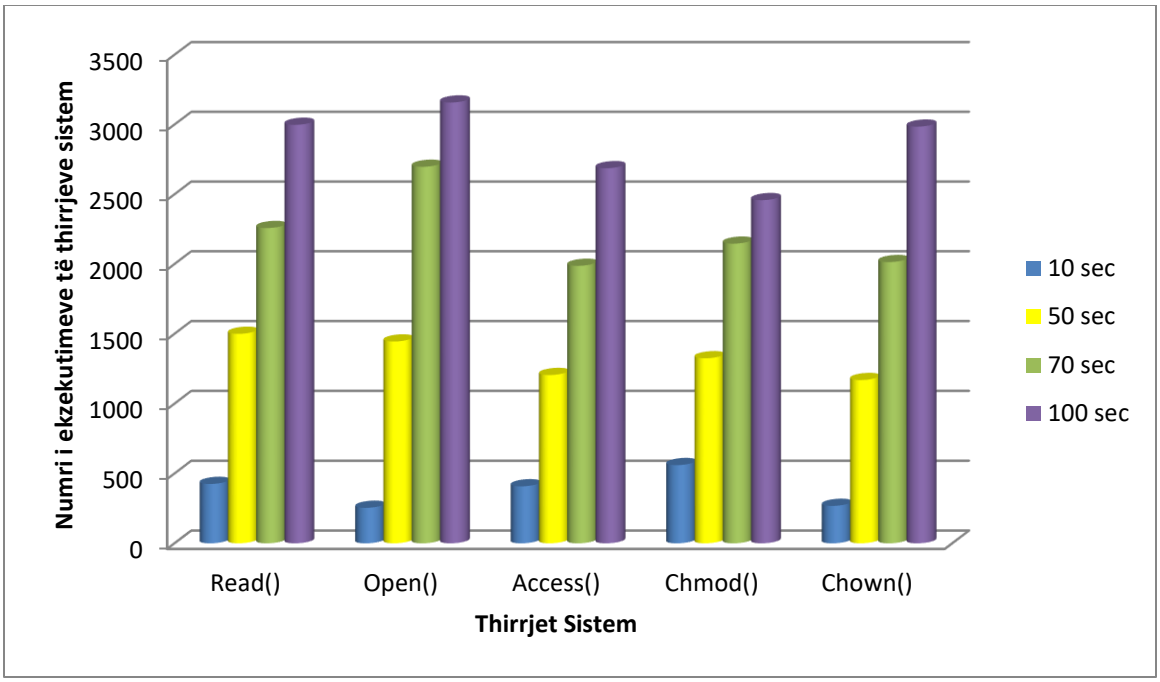


Fig. 26 Grafiku i thirrjeve sistem të applikacionit të virusuar

Grafiku i cili tregon diferencë midis aplikacionit zyrtar dhe atij të virusuar për 50 sec ekzekutim jepet më poshtë.

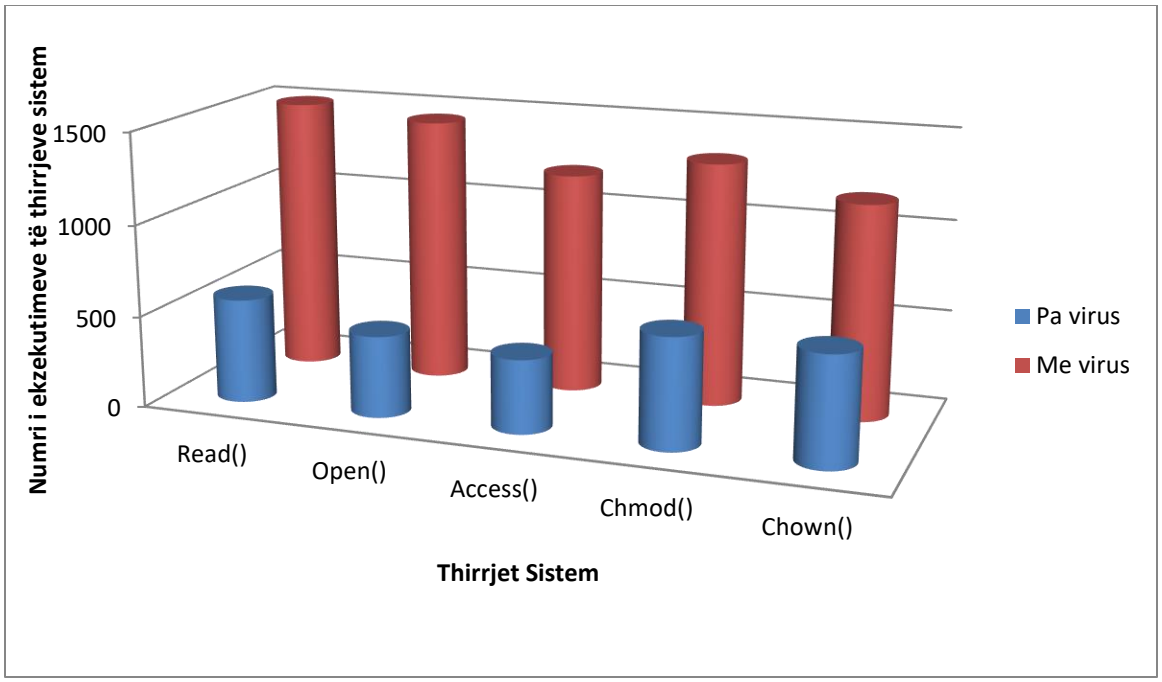


Fig. 27 Grafiku i krahasimit të thirrjeve sistem për 50 sec ekzekutim

Siç shikohet nga grafiku por edhe nga të dhënat në tabelë kemi që thirrja sistem Read() në rastin e aplikacionit të virusuar është thirrur 2.63 herë më shumë së në rastin normal. Kjo do të thotë një rritje me 263% të ekzekutimeve të thirrjeve sistem. Thirrja Open() është përdorur 3.24 herë më shumë në aplikacionin e virusuar (dmth 324 % më shumë përdorim). Thirrja Chmod është përdorur 2.2 herë më shumë kurse thirrja Chown është përdorur 1.95 herë më shumë ose 195% më shumë.

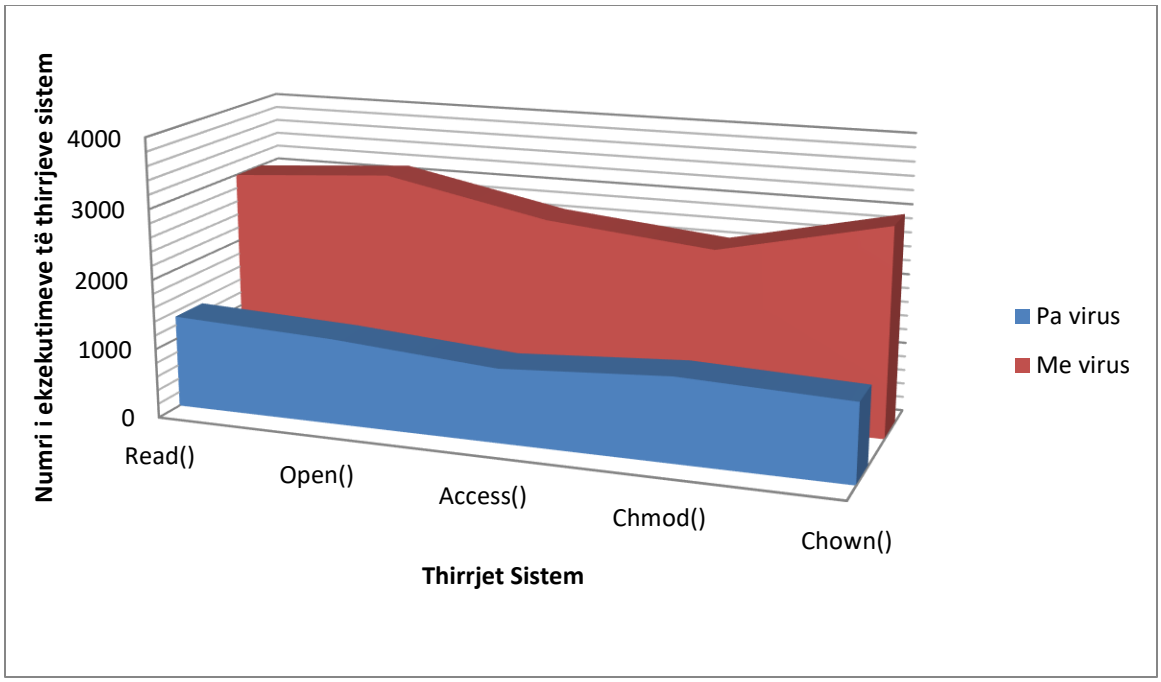


Fig. 28 Grafiku i krahasimit të thirrjeve sistem për 100 sek ekzekutim

Edhe këtu diferenca e përdorimit të thirrjeve sistem nga aplikacioni i virusuar është shumë e madhe. Në këtë rat kemi që thirrja sistem Read() është përdorur 2.62 herë më shumë nga aplikacioni i virusuar. Kjo do të thotë një rritje me 262% të përdorimit të kësaj thirrje. Thirrja sistem Access() është përdorur 2.54 herë më shumë në aplikacionin e virusuar.

Pra edhe në këtë rast përdorimi i ndryshimit tonë në SO Android rrezultoi i suksesshëm në detektimin e kësaj familje malware me anë të analizimit të thirrjeve sistem.

7.3.3 HongTouTou Malware

Prova e rradhë është bërë me një aplikacion që quhet Monkey jump 2  .

[329] Ky aplikacion ka rrezultuar i virusuar me HongTouTou. Aplikacionet e virusuara gjenden në markete Kineze. Ky virus ka si karakteristikë dërgimin e IMEI dhe IMSI në një server të jashtëm. Pastaj ai merr instruksione për të klikuar në faqë

web të ndryshme në varësi të fjalëve kyçe që i vjen nga server. Ai gjithashtu ka aftësinë për të download-uar një aplikacion i cili monitoron SMS-të dhe fut Spam-e në to.

Duke ekzekutuar aplikacionin zyrtar të marrë nga Google market kemi rrezultate e mëposhtme.

Tabela 13 Numri i thirrjeve sistem për aplikacionin zyrtar

Sys call / Koha	Read ()	Open ()	Access ()	Chmod ()	Chown()
10 sec	334	306	208	288	168
50 sec	3558	2158	1682	1875	987
70 sec	8069	9147	8745	6874	4897
100 sec	12560	11147	10256	9874	8455

Mbas instalimit të aplikacionit të virusuar me HongTouTou kemi rrezultatet e mëposhtme.

Tabela 14 Numri i thirrjeve sistem për aplikacionin e virusuar

Sys call / Koha	Read ()	Open ()	Access ()	Chmod ()	Chown()
10 sec	658	397	408	415	173
50 sec	4258	3054	2985	2014	1021
70 sec	8158	10256	9541	6994	4958
100 sec	12365	12148	11658	9824	8510

Këtu shikojmë diçka shumë interesante. Numri i thirrjeve sistem në aplikacioni e virusuar është më i vogël se numri i thirrjeve sistem për aplikacinin normal. Kjo ndodh për thirrjen sistem Read() dhe për kohën e ekzekutimit prej 100sekondash. Shpjegimet mund të jenë të shumta. Nga tabela ne shikojmë që për kohët nga 10 sek deri në 70 sek kemi rritje të përdorimit të thirrjes sitem Read() në aplikacionin e virusuar. Kurse mbas 100 sek ekzekutim numri nuk rritet më, biles ngelet më i vogël sesa numri i thirrjes Read() në palikacionin normal. Një arsye mund të jetë që aplikacioni i virusuar nuk e ka më nevojën e përdorimit të kësaj thirrhe sistem mbas një kohe të caktuar. Pra ai i kryen funksinet e veta dashakeqëse brënda 100 sekondëshit (siç tregohet edhe nga përdorimi i tepërt i thirrjes sistem Read()) dhe pastaj "fokusohet" në thirrje të tjera sistem për qëllimet e veta. Kjo tregohet nga përdorimi mbi normal i thirrjeve sistem Open() dhe Access() në mënyrë më të shpeshtë sesa në aplikacionin normal. Psh thirrja open() mbas 70 sek ekzekutim përdoret 1.12 herë më shumë në aplikacionin e virusuar sesa në atë normal.

Një gjë tjetër shumë interesante vihet re me thirrjen sistem Chown(). Shikohet nga tabela që vlerat si në aplikacionin zyrtar, si në atë të virusuar janë pak a shumë të ngjshashme. Dmth nuk ka diferencë. Kjo do të thotë që virusit nuk "I hyn në punë" kjo thirrje sistem për të arritur qëllimet e tij të këqia. Kjo mund të vijë nga fakti që ky malware merret kryesisht me marrje të dhenash dhe dërgimin e tyre në server të jashtëm. Pra nuk ndikon shumë në ndryshimin e funksionalitetit të sistemit operativ android.

Pra kur do tna duhet të bëjmë analizen e të dhënave që na vijnë nga android duhet të kemi kujdes sin ë zgjedhjen e kohëve të monitorimit si në përzgjedhjen e thirrjeve sistem në varësi të karakteristikave të malware-it që studiohet.

Nga ana grafike kemi

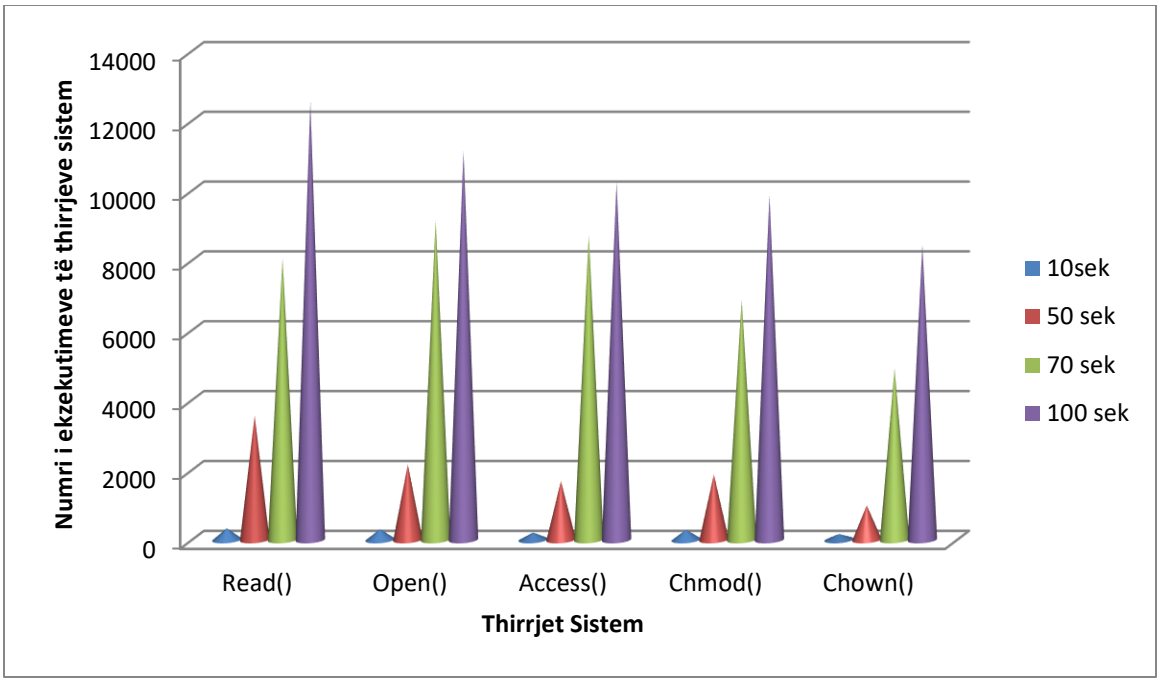


Fig. 29 Grafiku i thirrjeve sistem të aplikacionit zyrtar

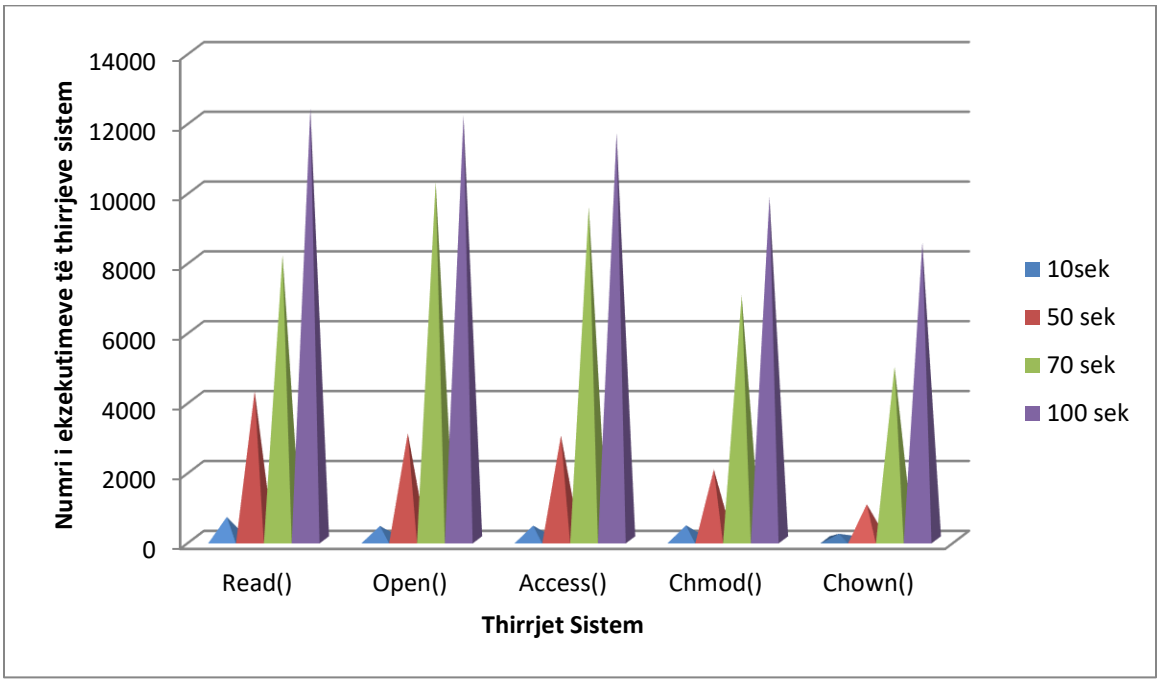


Fig. 30 Grafiku i thirrjeve sistem të aplikacionit të virusuar

Nqs do të krahasojmë rrezultatet e marra për 100 sek kemi.

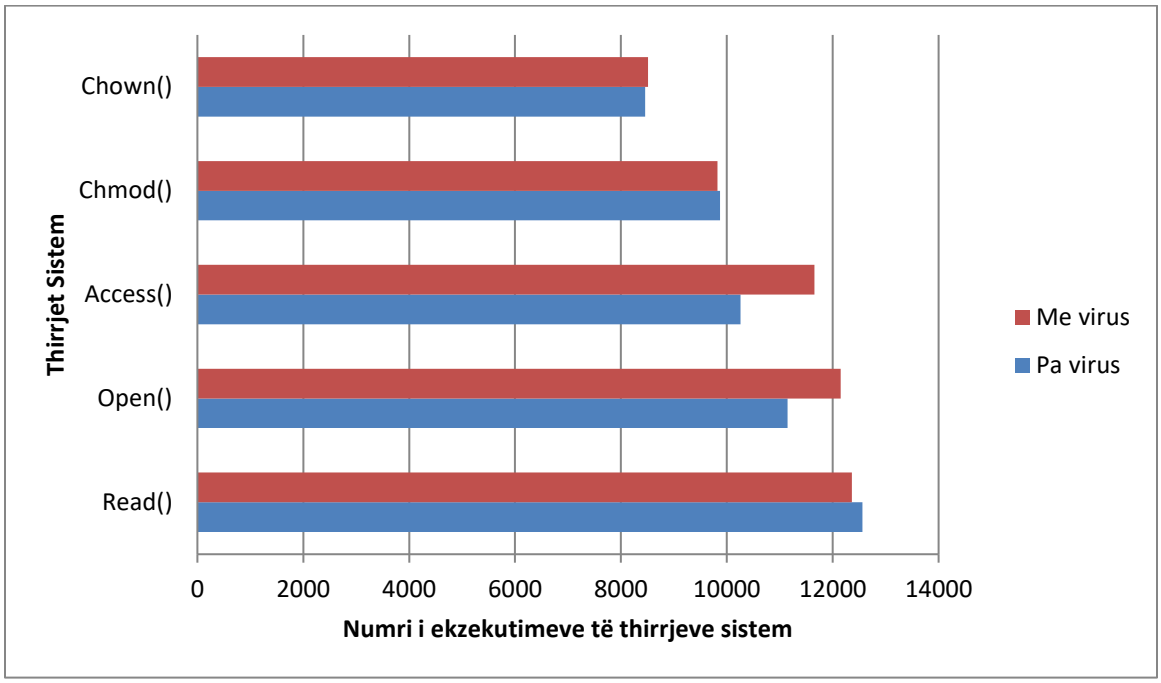


Fig. 31 Grafiku i krahasimit të thirrjeve sistem për 100 sek

Pra siç e thamë ëdhë më siëpr kur kemi rastin e 100 sek ekzekutim të aplikacioneve për thirrjen sistem Read() kemi që lehtësisht është përdorur më shumë në aplikacionin zyrtar dhe për thirrjen sistem Choën() është gati i barabartë numri i thirrjeve për këtë kohë ekzekutimi.

7.3.4 Droidkungfu Malware

Sic tregohet tek [326] [331] një nga malware-t më të dëmshëm është familja e malware-ve KungFu. Në këtë familje gjejmë viruse me emra të ndryshëm si KungFuA (KungFu1), KungFuB (KungFu2), KungFuC (KungFu3), KungFuD (KungFu4), KungFuE (KungFuSapp) ose KungFu Lena (Legacy Native) me veçori që analizohen siç vijon[112]: Të gjithë malwaret KungFu janë të paketuara dhe shkarkohen nga markete të tjera. Ai shton në aplikacion një shërbim të ri dhe një marrës të ri. Me privilegjin e root-it të marrë, ai automatikisht lançon shërbimin që të mos të ketë interaktivitet me

përdoruesin. KungFu mund të mbledhë informacione në telefonin e infektuar, duke përfshirë numrin IMEI, modelin e telefonit, versionin e sistemit operativ android. Varjanti i parë përdor koded Dalvik të bazuara në Java dhe një server C&C dhe payload-i enkriptohet me AES. Ndryshe, KungFuB përdor kodin e saj dhe tre server C&C. KungFuC trashëgon nga KungFuB, përdor dobësitë që ti lejojë pëdoruesit të marrë privilegjet duke dërguar një mesazh NETLINK (CVE-2009-1185)[113]. KungFuD trashëgon nga KungFuA dhe i enkripton binaries-t e sajë. KungFuE trashëgon nga KungFuD dhe enkripton disa stringje për të fshehur kodin e saj dhe përdor çertifikatën e përdoruesit në marketin zyrtar [114, 115]. Qëllimi i saj është që të shmangë detektimin nga programet antivirus të pajisjeve të lëvizshme. Në këtë mënyrë është shumë e vështirë që virusi të dedektohet. Android/DroidKungFu.A është një Trojan që dërgon informacione sensitive te
agresori duke përfshirë edhe funksionalitete të brënshme të telefonit. Ai gjthashtu shfrytëzon dobësitë për të marr aksesin root. Ndikimet nga infektimi janë [331] :

- Dërgon të dhëna sensitive
- Pëdor dobësi të njohura për të marrë aksesin root
- Instalon një aplikacion Android në direktorinë e sistemit
- Ka funksione të fshehta

Ky malware kërkon që përdoruesi qëllimisht ta instalojë në pajisjen e tij. Si gjithmonë, përdoruesit asnjëherë nuk duhet të instalojnë software të panjohura ose jo të besueshëm. Kjo sidomos është e vërtetë për software ilegale, siç janë aplikacionet e crack-a, këto janë vektorët e parapëlqyer për infektim nga malware-t.
Android/DroidKungFu.A është një version i crack-ar i një aplikacioni legjitim. Ai përfshinë funksionalitete që të ekzekutojë komanda të fshehura dhe të shfrytëzojë dobësitë në mënyrë që të marrë aksesin root. Instalimi i Trojan-it tregohet në figurën 32.

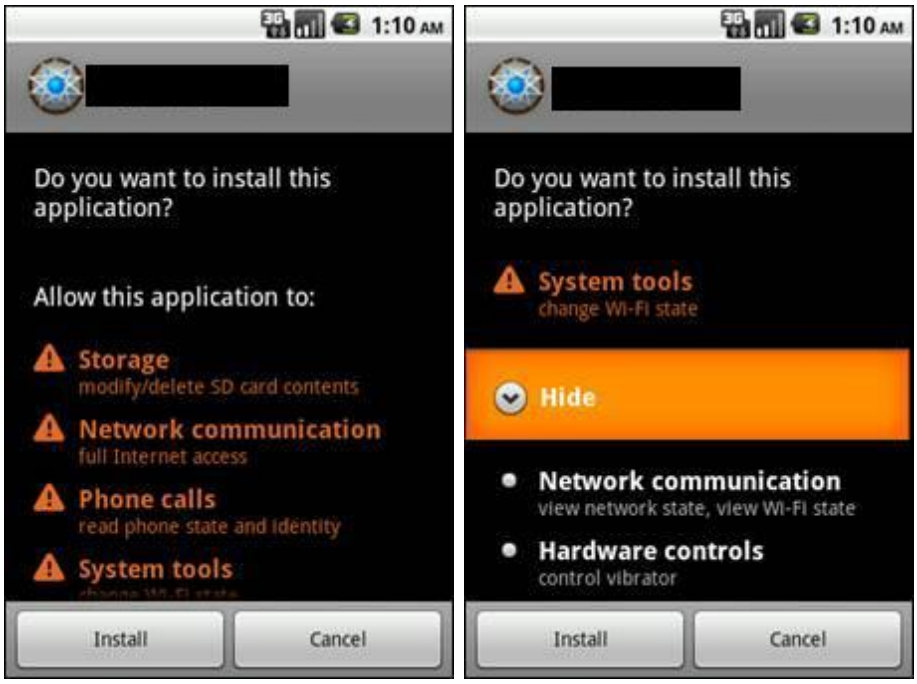


Fig. 32 Leja që kërkohet nga Android/DroidKungFu.A[331]

Kur pajisja e infektuar ndizet, shërbimi keqdashës "SearchService" do të aktivizohet

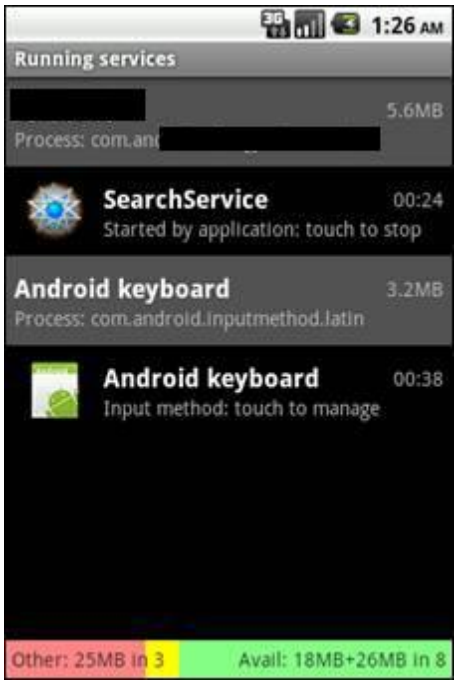


Fig. 33 Shërbimi SearchService [331]

Android/DroidKungFu.A në mënyrë të vazhduar lançon dy komanda të tijat Android të ekzekutueshme “assets/ratc” dhe “assets/gjsvro”. Këto më pas ruhen në file-in APK dhe dedektohen si Exploit/DiutesEx.B dhe Exploit/LVedu.B respektivisht. Në qoftë se një ndërhyrje është e sukseshme, Andorid/DroidKungFu.A ringreh /system.in në mënyrë që të kopjojë një APK të dëmshme në direktorinë “/system/app”. Në të kundër, ai shfaq një mesazh që thotë se ndërhyrja dështoi.

Fig 34 Dritarja që tregon se ndërhyrja dështoi

[331]Andorid/DroidKungFu.A kryen funksionalitetet e fshehta në përgjgje të komandava që jepen nga një server i jashtëm:

- Delete file
- Install APK
- Uninstall APK
- Launch Web browser with URL
- Launch application

Ai gjithashtu dërgon IMEI-n dhe nesë trojani e mori apo jo akesin root në serverin e jashtëm. APK e infektuar, e cila është e instaluar, do të futet në punë një shërbim kur pajisja ndizet, edhe nëse Andorid/DroidKungFu.A është hequr.

Duke instalaur një program të marrë në marketin zyrtar te google dhe duke e ekzekutuar do të marrim tabelën e mëposhtme. në këtë rast duke qenë se malware është I tipit më të rrezikshëm, dmth i tipit që me mënyra të ndryshme arrin të marrë root-in e sistemit, vendosa që si thirrje sistem në këtë rast të marr :
Read() Write() Open() Close() Time() Kill() Access() Rename() Chmod() Chown()

Tabela 15 Numri i thirrjeve sistem për aplikacionin zyrtar

Sys call **Koha**	**Read ()**	**Write ()**	**Open ()**	**Time()**	**Chown()**
10 sec	78	52	50	21	72
50 sec	156	93	87	35	159
70 sec	368	283	281	42	349
100 sec	526	395	391	57	489

Sys call **Koha**	**Kill ()**	**Rename ()**	**Access ()**	**Chmod ()**
10 sec	25	59	123	35
50 sec	87	105	307	68
70 sec	215	306	638	81
100 sec	288	415	905	112

Kurse mbas instalimi të një aplikacioni të virusuar me Droidkungfu dhe mbas ekzekutimit të tij kemi rrezultatet e mëposhtme.

Tabela 16 Numri i thirrjeve sistem për aplikacionin e virusuar

Sys call Koha	Read ()	Write ()	Open ()	Time()	Chown()
10 sec	157	105	104	22	207
50 sec	367	484	478	37	594
70 sec	698	869	862	43	984
100 sec	1028	1125	1115	57	1369

Sys call Koha	Kill ()	Rename ()	Access ()	Chmod ()
10 sec	168	67	150	139
50 sec	409	123	739	556
70 sec	699	325	1668	873
100 sec	973	432	1872	1305

Në këtë rast kemi marrë të dhëna shumë interesante. Siç shikohet nga tabelat diferenca e thirrjeve sistem nga aplikacioni i virusuar është shumë i madh. Sidomos kjo vihet re tek thirrjet Chown (), Access() dhe Chmod(). Respektivisht në secilën nga këto thirrje në aplikacionin e virusuar për 70 sekonda kemi: Për Chown() kemi një përdorim 2.82 herë më shumë (282%), për Access () kemi 2.67 herë më shumë (267%) dhe për Chmod () kemi 10.1 herë më shumë përdorim (1010%). Siç shikohet edhe nga këto rrezultate kemi një përdorim shumë të madh të këtyre thirrjeve sistem dhe sidomos të thirrjes Chmod().

Kjo vjen nga fakti që ky malware është i tipit root, dmth që tenton të marri rootin e sistemit. Dhe mbasi e merr shikohet që përdor shumë thirrjen Chmod për ndryshimin e lejeve(permisions) të file-ve të ndryshëm për të arritur qëllimet e tij të këqia. Një gjë tjetër që vihet re është mospërdorimi nga malware i thirrjes time(). Vlerat e saj janë afërsisht të barabarta me aplikacionin zyrtar. E njëjta gjë ndodh edhe me thirrjen Rename(). Për gjithë thirrjet e tjera sistem vihet re diferenca në e ekzekutimit të thirrjeve sistem nga aplikacioni i virusuar.

Nqs do ti paraqisnim grafikisht do të kemi.

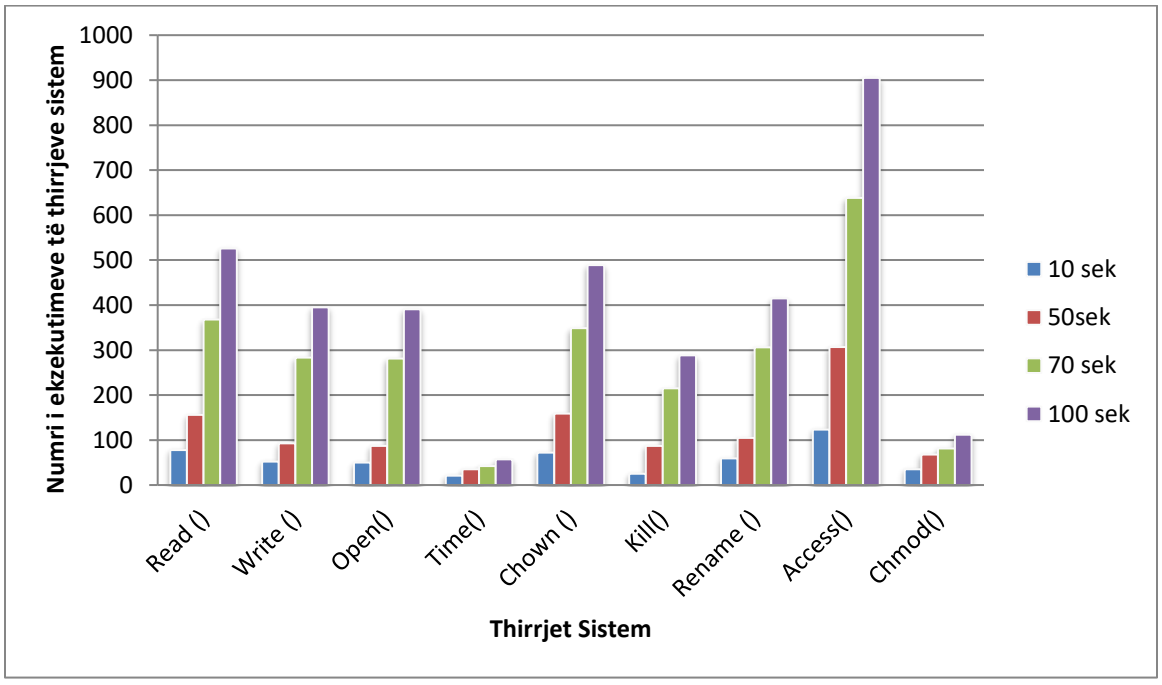


Fig. 35 Grafiku i thirrjeve sistem për aplikacionin zyrtar

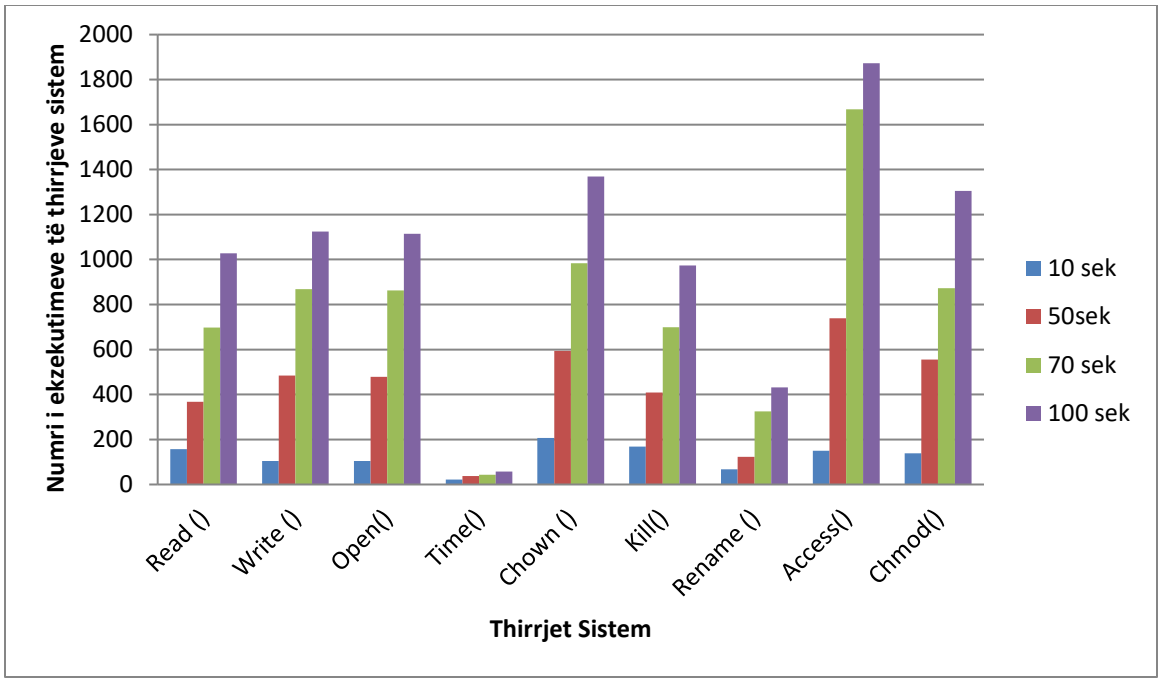


Fig. 36 Grafiku i thirrjeve sistem për aplikacionin e virusuar

Tani do të paraqes grafikun e ndryshimit të numrit të ekzekutimeve të thirrjeve sistem për 50 sek dhe 100 sek.

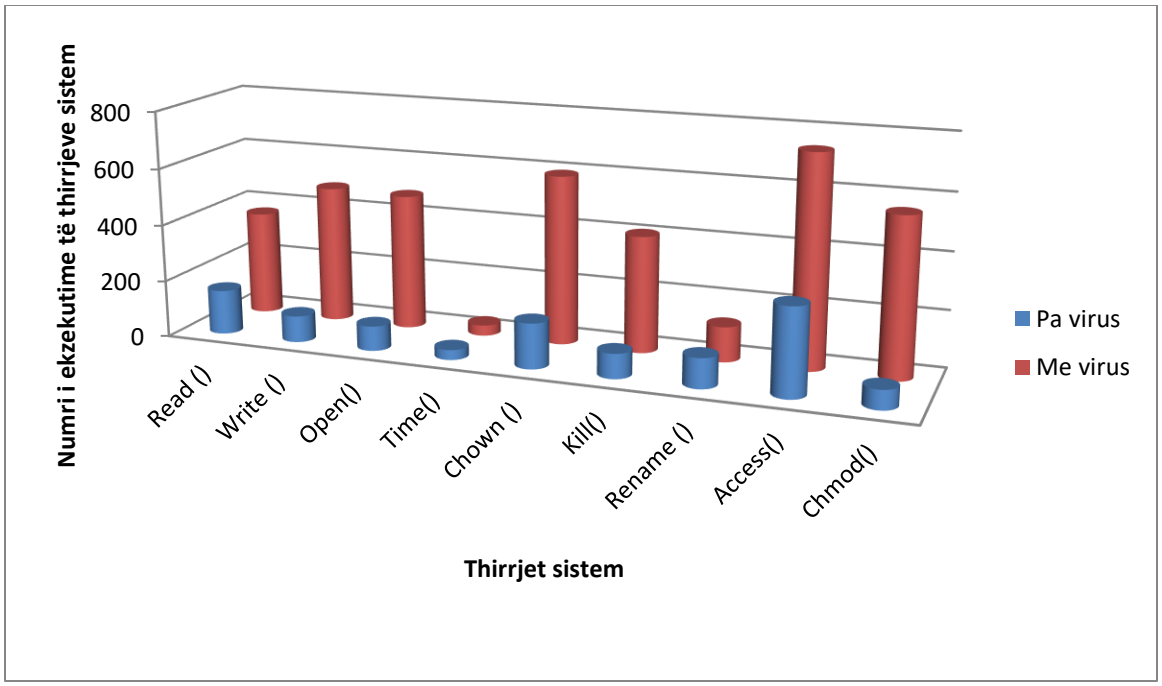


Fig 37 Grafiku i krahasimit të thirrjeve sistem për 50 sek

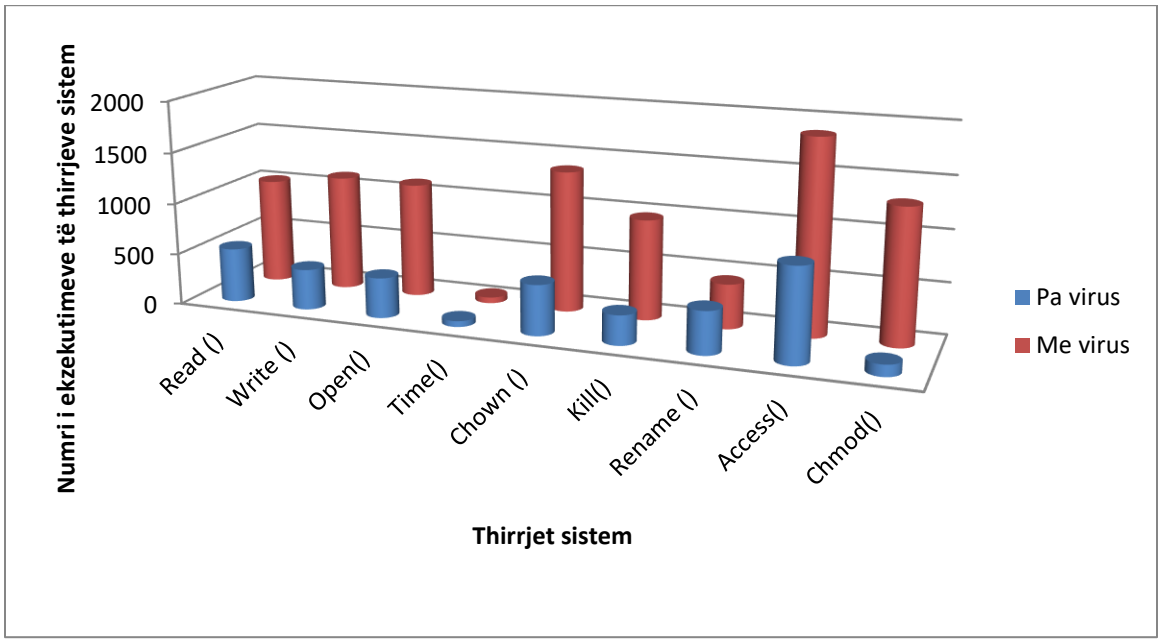


Fig 38 Grafiku i krahasimit të thirrjeve sistem për 100 sek

Edhe grafikisht tregohet diferenca e madhe në përdorimin e thirrjve sistem nga aplikacioni i virusuar. Pra siç u tregua edhe në të gjithë shembujt dhe eksperimente e mësipërme sistemi jonë realizon detektimin e malwareve duke bërë analizen e thirrjeve sistem. U tregua konkretisht që ky sistem funksionon.

KAPITULLI 8

Përfundime

8.1 Përfundimet e disertacionit

Në ditët e sotme problemi i sigurisë kompjuterike dhe sidomos problemi i sigurisë së sistemeve mobile është temë kryesore e konferencave dhe botimeve shkencore dhe sidomos është një problem i përditshëm për secilin përdorues.

Në këtë disertacion u trajtua tema e sigurisë së sistemeve operative mobile duke u fokusuar te sistemin operativ android. Këtu u propozua një algoritëm i ri i cili ka në bazë të tij vendosjen e një shtrese shtesë në sistemin operativ Android e cila do të shërbejë për kapjen, memorizimin dhe dërgimin e thirrjeve sistem nga aparati mobile në një server të jashtëm. Kjo shtresë u vendos në kernelin e SO Android. Qëllimi i kësaj pune është rritja e sigurisë së këtyre sistemeve nëpërmjet analizmit të thirrjeve sistem.

Zgjedhja e thirrjeve sistem si element kyç në përcaktimin dhe në detektimin e malwareve u bë mbas studimit intensiv dhe ezaurues të literaturës në lidhje me malwaret dhe me mënyrat aktuale të mbrojtjes së këtyre sistemeve. U analizuan 1260 malware të cilat u grupuan në 49 familje të ndryshme. Kuptohet që numri i malwareve është shumë më i madh por këto tregojnë tendencën dhe përmbledhin pjesën më të madhe të teknikave dashakeqëse.

Nga kjo analizë e cila mbeshtetet ne [8] u arrit në përfundimin se:

- Nga 1260 malware 1083 ose 86.0% janë të ripaketuara (repackaged).
- Gati 1/3 (ose 36.7%) hyjnë në root të sistemit Android.
- Më shumë se 90% e kthejnë telefonin ne një pajisje që mund të komandohet nag rrjeti ose nga SMS.

- 27 familje malware or 51.1% Vjedhin informacionet e përdoruesit, duke përfshirë account-et dhe mesazhet.

Mbas kësaj analize u kalua në eksperimentimin e zgjidhjes së propozuar me anë të instalimit dhe ekzekutimit të malwareve reale. Eksperimentet ishin të shumta. U përzgjodhën 4 familje malware-esh, AnserverBot, Basebridge,HongToutou dhe DroidkungFu. Llogjika e përzgjdhjes së tyre ishte shfaqja e karaktekteristikave të ndryshme gjatë kohës së infektimit. Kjo solli nxjerrjen e përfundimeve më të sakta mbi efektshmërine e zgjidhjes së propozuar duke u testuar me familje të ndryshme virusesh të cilët sjellin dëme të ndryshme.

Gjatë eksperimenteve me Anserverbot, i cili është konsideruar si një nga malwaret më të sofistikuar morëm të dhënat për një aplikacion normal të download-uar në marketin zyrtar të google dhe i krahasuam me të dhënat e marra gjatë ekzekutimit të aplikacionit të virusuar me Anserverbot. Rrezultatet ishin evidente edhe me sy të lirë. Kishim që :

- Thirrja sistem Read() është përdorur 3.83 herë më shumë në aplikacioni e virusuar sesa në atë normal, pra 383% më shumë.
- Thirrja sistem Write () është përdorur 4.26 herë më shumë në aplikacionin e virusuar. Pra 426% më shumë.
- Thirrja sistem Kill() u përdor 4.18 herë më shumë. Pra 418% më shumë

Pra siç e theksova nëpërmjet analizimit dhe krahasimit të vlerave të thirrjve sistem ne arritëm në detektimin e këtij malware.

Te HongTouTou gjatë fazes eksperimentale kishim diçka interesante. Thirrja sistem read (), për 100 sek, ishte përdorur më pak në aplikacionin e virusuar sesa në aplikacionin normal. Në intervalin nga 10 sek deri në 70 sek kishim një përdorim më të madh në kësaj thirrje nga aplikacioni i virusuar, por pastaj në 100 sek stopohet. Kjo vjen nga karakteristika e virusit i cili për arritjen e qëllimeve të tij të këqia e shfrytëzon këtë thirrje deri në 70 sekondat e para dhe pastaj, fokusohet në thirrje të

tjera sistem. Pra ktu na del një elemet i ri në analizen e thirrjeve sistem. Për ta patur sa më të saktë duhet të zgjedhim intervalet e duhura kohore në mënyrë që sistemi të na detektojë malware-in.

Tek Droidkungfu mora në shqyrtim më shumë thirrje sistem në mënyrë që të nxjerr sa më mirë detektimin apo jo të malware-it. Gjatë eksperimenteve u vu re që:

- Thirrja sistem read() u përdor 1.95 herë më shumë në aplikacioni e virusuar. Pra 195% më shumë
- Thirrja access () u përdor 2 herë më shumë. Pra 200%
- Thirrja Chmod() u përdor 11.6 herë më shumë. Pra 1160% herë më shume.

Nga këto të dhëna ne nxjerrim që karakteristika e këtij virusi, i cili është i tipit root shfrytëzon më shumë thirrjet që merren me ndryshimin e lejeve të file-ve të ndryshëm. Gjithashtu ai përdor edhe thirrjet read dhe access për të aritur qëllimet e tij të këqia, por diferencë u pa në përdorimin e Chmod.

Si përfundim mund të themi që sistemi që u propozua, dmth shtimi i një shtrese në kernelin e sistemit operativ Android, i cili merr dhe dërgon e thirrjet sistem në një server të jashtëm, rrezultoi i suksesshëm në detektimin e malwareve më të rrezikshëm që janë kapur deri në ditët e sotme.

8.2 Puna në të ardhmen

Sistemi që u propozua është realizuar në kushte eksperimentale. Që ky sistem të bëhet aktiv dhe i përdorshëm edhe në jetën reale duhet që:

- Të realizojmë enkriptimin e të dhënave që trasmetohen nga paisja mobile në serverin për përpunimin e të dhënave. Kjo do të realizohet duke përdorur teknikat e prezantuara tek [339].
- Të realizojmë teknika të data mining për analizimin e të gjithë thirrjeve sistem që vijnë nga paisja mobile.

- Gjetja e një mënyre të sigurtë dhe pa kosto për lajmërimin e përdoruesit në rast të detektimit të një malware.

REFERENCAT

[1] Cheng, J., S. H. Wong, H. Yang, and S. Lu (2007) "SmartSiren: Virus Detection and Alert for Smartphones," in Proceedings of the International conference on Mobile Systems, Applications, and Services (MobiSys).

[2] CCNA Security 640-554 Official Cert Guide, CISCO Press, July 2012

[3] Enck M. (2011) Dissertation "Analysis Techniques for Mobile Operating System Security" Computer Science and Engineering, Pennsylvania State University.

[4] Ramu S., "Mobile Malware Evolution, Detection and Difense (2012) EECE 571B, Term Survey Paper, Institute for Computing, Information and Cognitive Systems (ICICS), University of British Columbia, Vancuver, Canada.

[5] Android Developers Documentation https://developer.android.com/guide/topics/permissions/overview

[6] Castillo, Carlos A., (2011) "Android Malware Past Present and Future" Mobile Security Working Group, McAfee.

[7] LA Polla, M., Martinelli, F., Sgandurra, D. (1971) "A Survey in Security for Mobile Devices" in Proceedings of IEEE Communication Surveys and Tutorials

[8] Zhou, Y., Jiang, X. (2012) "Dissecting Android Malware: Charachterization and Evolution" IEEE 978-0-7695-4681-0, San Francisco, USA.

[9] Fraim, L. J. (1983) "Scomp: A Solution to the Multilevel Security Problem," IEEE Computer, 16(7), pp. 26–24.

[10] Shockley, W. R., T. F. Tao, and M. F. Thompson (1988) "An Overview of the GEMSOS Class A1 Technology and Application Expërience," in Proceedings of the 11th National Computer Security Conference, pp. 238–245.

[11] Rushby, J. M. (1981) "Design and Verification of Secure Systems," ACM SIGOPS Opërating Systems Review, 15(5).

[12] (1982) "Proof of Separability: A Verification Technique for a Class of Security Kernels," in Proceedings of the 5th International Symposium on Programming.

[13] Accetta, M., R. Baron, W. Bolosky, D. Golub, R. R. an dAVadis Tevanian, and M. Young (1986) "Mach: A New Kernel Foundation For UNIX Development," in Proceedings of the Summer USENIX Conference.

[14] Liedtke, J. (1993) "Improving IPC by Kernel Design," in Proceedings of the 14th ACM Symposium on Opërating Sysetms Principles (SOSP).

[15] (1995) “On μ-Kernel Construction,” in Proceedings of the 15th ACM Symposium on Opërating Sysetms Principles (SOSP).

[16] Goldberg , R. (1972) Arcitechural Principles for Virtual Computer Systems, Ph.D. thesis, Harvard University.

[17] (1974) “Survey of Virtual Machine Research,” IEEE Computer Magazine,7, pp. 34–45.

[18] Creasy , R. J. (1981) “The Origin of the VM/370 Time-Sharing System,” IBM J. Research and Development, 25(5), pp. 483–490.

[19] Kelem, N. L . and R. J. Feiertag (1991) “A Sepration Model for Virtual Machine Monitors,” in Proceedings of the IEEE Symposium on Research in Security and Privacy.

[20] Sugerman, J., G. Venkitachalam, and B.-H. Lim (2001) “Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor,” in Proceedings of the USENIX Annual Technical Conference.

[21] Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield (2003) “Xen and the Art of Virtualization,” in Proceedings of the 19th ACM Symposium on Opërating Systems Principles (SOSP).

[22] Garfinkel, T., B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh (2003) “Terra: A Virtual Machine-Based Platform for Trusted Computing,” in Proceedings of the 19th ACM Symposium on Opërating Systems Principles (SOSP), pp.193–206.

[23] Sailer, R., T. Jaeger, E. Valdez, R. Caceres, R. Përez, S. Berger, J. L. Griffin, and L. van Doom (2005) “Building a MAC-Based Security Architecture
for the Xen Open-Source Hypërvisor,” in Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC), pp. 276–285.

[24] Denning, D. E. (1976) “A Lattice Model of Secure Information Flow,” Communications of the ACM, 19(5), pp. 236–243.

[25] Bell, D. E. and L. J. LaPadula (1973) Secure Computer Systems: Mathematical
Foundations, Tech. Rep. MTR-2547, Vol. 1, MITRE Corp., Bedford, MA.

[26] Biba, K. J. (1977) Integrity Considerations for Secure Computer Systems, Tech.
Rep. MTR-3153, MITRE.

[27] Clark, D. D. and D. Wilson (1987) “A Comparison of Military and Commercial
Security Policies,” in Proceedings IEEE Symposium on Security and Privacy.

[28] Shankar , U ., T. Jaeger, and R. Sailer (2006) “Toward Autmoated Information-Flow Integrity Verification for Security-Critical Applications,” in Proceedings of the ISOC Network and Distributed Systems Security Symposium (NDSS).

[29] Li, N., Z. Mao, and H. Chen (2007) “Usable Mandatory Integrity Protection for Opërating Systems,” in Proceedings of the IEEE Symposium on Security and Privacy, pp. 164–178.

[30] Sun, W., R. Sekar, G. Poothia, and T. Karandikar (2008) “Practical Proactive Integrity Protection: A Basis for Malware Defense,” in Proceedings of the IEEE Symposium on Security and Privacy.

[31] Vandebogart, S., P. Efstathopoulos, E. Kohler, M. Krohn, C. Frey, D. Ziegler, F. Kaashoek, R. Morris, and D. Mazieres` (2007) “Labels and Event Processes in the Asbestos Opërating System,” ACM Transactions on Computer Systems (TOCS), 25(4).

[32] Zeldovich, N., S. Boyd-Wickizer, E. Kohler, and D. Mazi`eres (2006) “Making Information Flow Explicit in HiStar,” in Proceedings of the 7th symposium on Opërating Systems Design and Implementation (OSDI), pp. 263–278.

[33] Krohn, M., A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris (2007) “Information Flow Control for Standard OS Abstractions,”
in Proceedings of ACM Symposium on Opërating Systems Principles (SOSP), pp.321–334.

[34] Myers, A. C. and B. Liskov (2000) “Protecting Privacy Using the Decentralized
Label Model,” ACM Transactions on Software Engineering and Methodology, 9(4),pp. 410–442.

[35] Badger, L., D. Sterne, D. Sherman, K. Walker, and S. Haghighat (1995) “A Domain and Type Enforcement UNIX Prototype,” in Proceedings of the Fifth
USENIX UNIX Security Symposium.

[36] Boebert, W. E. and R. Y. Kain (1985) “A Practical Alternative to Hierarchical Integrity Policies,” in Proceedings of the 8th National Computer Security Conference.

[37] National Security Agency, “Security-Enhanced Linux (SELinux),” http:// www.nsa.gov/selinux.

[38] Spencer, R., S. Smalley, P. Loscocco, M. Hibler, D. Andersen, and J. Lepreau (1999) “The Flask Security Architecture: System Support for Diverse

Security Policies," in Proceedings of the 8th USENIX Security Symposium, pp. 123–139.

[39] Ferraiolo, D., J. Cugini, and D. R. Kuhn (1995) "Role-Based Access Control (RBAC): Features and Motivations," in Proceedings of 11th Annual Computer Security Application Conference (ACSAC).

[40] Novell (Accessed January 2011), "AppArmor Application Security for Linux," http://www.novell.com/linux/security/apparmor/.

[41] C owa n , C ., S. Beattie, G. Kroah-Hartman, C. Pu, P. Wagle, and V. Gligor (2000) "SubDomain: Parsimonious Server Security," in Proceedings of the 14th USENIX conference on System administration.

[42] Saltzer, J. and M. Schroeder (1975) "The Protection of Information in Computer Systems," Proceedings of the IEEE, 63(9).

[43] Hardy, N. "The Confused Deputy: (or why capabilities might have been invented)," SIGOPS Opërating Systems Review, 22(4), pp. 36–38.

[44] Wulf, W., E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, and F. Pollack (1974) "HYDRA: The Kernel of a Multiprocessor Opërating Systems," Communications of the ACM, 17(6).

[45] Boebert, W. E. (1984) "On the Inability of an Unmodified Capability Machine to Enforce the *-propërty," in Proceedings of the DoD/NBS Computer Security Conference, pp. 291–293.

[46] Karger, P. A. and A. J. Herbert (1984) "An Augmented Capability Architecture to Support Lattice Security and Traceability of Access," in Proceedings of the IEEE Symposium on Security and Privacy.

[47] Karger, P. A. (1988) Improving Security and Përformance for Capability Systems, Ph.D. thesis, University of Cambridge.

[48] Shapiro, J. S. (1999) EROS: A Capability System, Ph.D. thesis, University of Pennsylvania.

[49] Goldberg, I., D. Wagner, R. Thomas, and E. Brewer (1996) "A Secure Environment for Untrusted Helpër Applicationos: Confining the Wily Hacker," inProceedings of the USENIX Security Symposium.

[50] Jaeger, T., A. Rubin, and A. Prakash (1996) "Building Systems That Flexibly Control Downloaded Executable Content," in Proceedings of the 6th USENIX UNIX Security Symposium, pp. 131–148.

[51] Acharya, A. and M. Raje (2000) "MAPbox: Using Parameterized Behavior Classes to Confine Untrusted Applications," in Proceedings of the 9th USENIX Security Symposium.

[52] Lai, N. and T. Gray (1988) “Strengthening Discresionary Access Controls to Inhibit Trojan Horses and Computer Viruses,” in Proceedings of the 1988 USENIX Summer Symposium, pp. 275–286.

[53] Berman, A., V. Bourassa, and E. Selberg (1995) “TRON: Process-Specific File Protection for the UNIX Operating System,” in Proceedings of the USENIX Technical Conference, pp. 165–175.

[54] Seaborn , M., “Plash,” http://plash.beasts.org.

[55] Stiegler , M ., A. Karp, K.-P. Yee, and M. Miller (2004) Polaris: Virus Safe Computing for Windows XP, Tech. Rep. HPL-2004-221, HP Laboratories Palo Alto.

[56] Wichers, D., D. Cook, R. Olsson, J. Crossley, P. Kerchen, K. Levitt, and R. Lo (1990) “PACL’s: An Access Control List Approach to Anti-viral Security,” in Proceedings of the 13th National Computer Security Conference, pp. 340–349.

[57] Enck, W., P. McDaniel, and T. Jaeger (2008) “PinUP: Pinning User Files to Known Applications,” in Proceedings of the 24th Annual Computer Security Applications Conference (ACSAC).

[58] Enck, W., S. Rueda, Y. Sreenivasan, J. Schiffman, L. S. Clair, T. Jaeger, and P. McDaniel (2007) “Protecting Users from ”Themselves”,” in Proceedings of the 1st ACM Computer Security Architectures Workshop.

[59] Ioannidis, S., S. Bellovin, and J. Smith (2002) “Sub-Opërating Systems: A New Approach to Application Security,” in Proceedings of ACM SIGOPS European workshop, pp. 108–115.

[60] Snowberger, P. and D. Thain (2005) Sub-Identities: Towards Opërating System Support for Distributed System Security, Tech. Rep. 2005-18, University of Notre Dame, Department of Computer Science and Engineering.

[61] Schmid, M., F. Hill, and A. Gosh (2002) “Protecting Data from Malicious Software,” in Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC).

[62] Sailer, R., X. Zhang, T. Jaeger, and L. van Doom (2004) “Design and Implementation of a TCG-based Integrity Measurement Architecture,” in Proceedingsof the 13th USENIX Security Symposium.

[63] Trusted Computing Group (2007), “TCG mobile reference architecture specification version 1.0,” https://www.trustedcomputinggroup.org/specs/ mobilephone/tcg- mobile- reference- architecture- 1.0.pdf.

[64] Zhang, X., O. Aciic¸mez, and J.-P. Seifert (2007) “A Trusted Mobile Phone Reference Architecture via Secure Kernel,” in Proceedings of the ACM workshop on Scalable Trusted Computing, pp. 7–14.

[65] (2009) “Building E fficient Integrity Measurement and Attestation for Mobile Phone Platforms,” in Proceedings of the First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec).

[66] Nauman, M., S. Khan, X. Zhang, and J.-P. Seifert (2010) “Beyond Kernellevel Integrity Measurement: Enabling Remote Attestation for the Android Platform,” in Proceedings of the 3rd International Conference on Trust and Trustworthy Computing.

[67] Muthukumaran, D., A. Sawani, J. Schiffman, B. M. Jung, and T. Jaeger (2008) “Measuring Integrity on Mobile Phone Systems,” in Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 155–164.

[68] Shabtai, A., Y. Fledel, and Y. Elovici (2010) “Securing Android-Powered Mobile Devices Using SELinux,” IEEE Security and Privacy Magazine

[69] VMware, Inc., “VMware Mobile Virtualization Platform,” http://www.vmware.com/products/mobile/, accessed January 2011.

[70] Open Kernel Labs, “OK:Android,” http://www.ok- labs.com/products/ ok- android, accessed January 2011.

[71] Lee, S.-M., S. bum Suh, B. Jeong, and S. Mo (2008) “A Multi-Layer Mandatory Access Control Mechanism for Mobile Devices Based on Virtualization,” in Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC)

[72] Winter, J. (2008) “Trusted Computing Building Blocks for Embedded Linuxbased ARM TrustZone Platforms,” in Proceedings of the 3rd ACM workshop on Scalable Trusted Computing (STC).

[73] Mulliner, C., G. Vigna, D. Dagon, and W. Lee (2006) “Using Labeling to Prevent Cross-Service Attacks Against Smart Phones,” in Proceedings of Detectionof Intrusions and Malware & Vulnerability Assessment (DIMVA).

[74] Ion, I., B. Dragovic, and B. Crispo (2007) “Extending the Java Virtual Machine to Enforce Fine-Grained Security Policies in Mobile Devices,” in Proceedings of the Annual Computer Security Applications Conference (ACSAC).

[75] Desmet, L., W. Joosen, F. Massacci, P. Philippaerts, F. Piessens, I. Siahaan, and D. Vanoverberghe (2008) “Security-by-contract on the .NET platform,” Information Security Technical Report, 13(1), pp. 25–32.

[76] Nauman, M., S. Khan, and X. Zhang (2010) “Apex: Extending Android Përmission Model and Enforcement with User-defined Runtime Constraints,” in Proceedings of ASIACCS.

[77] Conti, M., V. T. N. Nguyen, and B. Crispo (2010) “CRePE: Context-Related Policy Enforcement for Android,” in Proceedings of the 13th Information Security Conference (ISC).

[78] Ongtang, M., S. McLaughlin, W. Enck, and P. McDaniel (2009) “Semantically Rich Application-Centric Security in Android,” in Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC), pp. 340–349.

[79] Ongtang , M., K. Butler, and P. McDaniel (2010) “Porscha: Policy Oriented Secure Content Handling in Android,” in Proceedings of the 26th Annual ComputerSecurity Applications Conference (ACSAC).

[80] Karlson, A. K., A. B. Brush, and S. Schechter (2009) “Can I Borrow Your Phone? Understanding Concerns When Sharing Mobile Phones,” in Proceedings of the Conference on Human Factors in Computing Systems (CHI).

[81] Liu, Y., A. Rahmati, Y. Huang, H. Jang, L. Zhong, Y. Zhang, and S. Zhang (2009) “xShare: Supporting Impromptu Sharing of Mobile Phones,” in Proceedings of the International conference on Mobile Systems, Applications, and Services (MobiSys).

[82] Ni, X., Z. Yang, X. Bai, A. C. Champion, and D. Xuan (2009) “Di ffUser: Di fferentiated User Access Control on Smartphones,” in Proceedings of the 5th IEEE Workshop on Wireless and Sensor Networks Security (WSNS).

[83] Shin, W., S. Kiyomoto, K. Fukushima, and T. Tanaka (2010) “A Formal Model to Analyze the Përmission Authorization and Enforcement in the Android Framework,” in Proceedings of the International Conference on Social Computing.

[84] Shin, W., S. Kwak, S. Kiyomoto, K. Fukushima, and T. Tanaka (2010) “A Small but Non-negligible Flaw in the Android Përmission Scheme,” in Proceedings of the International Symposium on Policies for Distributed Systems and Networks.

[85] Chaudhuri, A. (2009) “Language-Based Security on Android,” in Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS).

[86] Fuchs, A. P., A. Chaudhuri, and J. S. Foster, “ScanDroid: Automated Security Certification of Android Applications,” http://www.cs.umd.edu/~avik/ projects/scandroidascaa/papër.pdf, accessed January 11, 2011.

[87] Venugopal, D. and G. Hu (2008) “E fficient Signature Based Malware Detectionon Mobile Devices,” Mobile Information Systems, 4(1).

[88] Bose, A., X. Hu, K. G. Shin, and T. Park (2008) “Behavioral Detection of Malware on Mobile Handsets,” in Proceedings of the 6th International Conferenceon Mobile Systems, Applications, and Services (MobiSys).

[89] Shabtai, A., U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss (2011) ""Andromaly": A Behavioral Malware Detection Framework for Android Devices,"Journal of Intelligent Information Systems, published online January 2011.

[90] Nash, D. C., T. L. Martin, D. S. Ha, and M. S. Hsiao (2005) "Towards an Intrustion Detection System for Battery Exhaustion Attacks on Mobile Computing Devices," in Proceedings of the 3rd International Conference on Përvasive Computing and Communications Workshops (PërCom Workshops).

[91] Kim, H., J. Smith, and K. G. Shin (2008) "Detecting Energy-Greedy Anomalies and Mobile Malware Variants," in Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 239–252.

[92] Miettinen, M., P. Halonen, and K. Hatonen (2006) "Host-Based Intrusion Detection for Advanced Mobile Devices," in Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA).

[93] Schmidt, A.-D., F. Peters, F. Lamour, C. Scheel, S. A. Camtepe, and S. Albayrak (2009) "Monitoring Smartphones for Anomaly Detection," Mobile Networks and Applications, 14(1), pp. 92–106.

[94] Chun, B.-G. and P. Maniatis (2009) "Augmented Smartphone Applications Through Clone Cloud Execution," in Proceedings of the 12th Conference on Hot Topics in Opërating Systems (HotOS).

[95] Portokalidis, G., P. Homburg, K. Anagnostakis, and H. Bos (2010) "Paranoid Android: Versatile Protection For Smartphones," in Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC).

[96] Oberheide, J., K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian (2008) "Virtualized In-Cloud Security Services for Mobile Devices," in Proceedings of the 1st Workshop on Virtualization in Mobile Computing.

[97] Oberheide, J., E. Cooke, and F. Jahanian (2008) "CloudAV: N-Version Antivirus in the Network Cloud," in Proceedings of the 17th USENIX Security Symposium.

[98] Migliavacca, M., I. Papagiannis, D. M. Eyers, B. Shand, J. Bacon, and P. Pietzuch (2010) "DEFCon: High-Përformance Event Processing with Information Security," in Proceedings of the USENIX Annual Technical Conference.

[99] Wang, X., Z. Li, N. Li, and J. Y. Choi (2008) "PRECIP: Towards Practical and Retrofittable Confidential Information Protection," in Proceedings of 15th Network and Distributed System Security Symposium (NDSS08).

[100] Sabelfeld, A. and A. C. Myers (2003) “Language-based information-flow security,” IEEE Journal on Selected Areas in Communication, 21(1), pp. 5–19.

[101] Myers, A. C. (1999) “JFlow: Practical Mostly-Static Information Flow Control,”in Proceedings of the ACM Symposium on Principles of Programming Langauges(POPL).

[102] Heintze, N. and J. G. Riecke (1998) “The SLam Calculus: Programming with Secrecy and Integrity,” in Proceedings of the Symposium on Principles of Programming Languages (POPL), pp. 365–377.

[103] Roy, I., D. E. Porter, M. D. Bond, K. S. McKinley, and E. Witchel (2009) “Laminar: Practical Fine-Grained Decentralized Information Flow Control,” in Proceedings of the Conference on Programming Language Design and Implementation (PLDI), pp. 63–74.

[104] Hicks, B., K. Ahmadizadeh, and P. McDaniel (2006) “Understanding Practical Application Development in Security-Typed Languages,” in 22st Annual Computer Security Applications Conference (ACSAC), Miami, Fl, pp. 153–164.

[105] Denning, D. E. and P. J. Denning (1977) “Certification of Programs for Secure Information Flow,” Communications of the ACM, 20(7).

[106] Newsome, J. and D. Song (2005) “Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software,” in Proceedings of the 12th Network and Distributed System Security Symposium (NDSS).

[107] Qin, F., C. Wang, Z. Li, H. seop Kim, Y. Zhou, and Y. Wu (2006) “LIFT: A Low-Overhead Practical Information Flow Tracking System for Detecting Security Attacks,” in Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society Washington, DC, USA, pp. 135–148.

[108] Clause, J., W. Li, and A. Orso (2007) “Dytan: A Generic Dynamic Taint Analysis Framework,” in Proceedings of the 2007 international symposium on Software testing and analysis, ACM New York, NY, USA, pp. 196–206.

[109] Yin, H., D. Song, M. Egele, C. Kruegel, and E. Kirda (2007) “Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis,” in Proceedings of the 14th ACM conference on Computer and Communications Security, pp. 116–127.

[110] Egele, M., C. Kruegel, E. Kirda, H. Yin, and D. Song (2007) “Dyanmic Spyware Analysis,” in Proceedings of the USENIX Annual Technical Conference, pp. 233–246.

[111] Zhu, D., J. Jung, D. Sung, T. Kohno, and D. Wetherall (2009) Privacy Scope: A Precise Information Flow Tracking System For Finding Application Leaks, Tech. Rep. EECS-2009-145, Department of Computer Science, UC Berkeley.

[112] Costa, M., J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham (2005) "Vigilante: End-to-End Containment of Internet Worms," in Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), pp. 133–147.

[113] Vachharajani, N., M. J. Bridges, J. Chang, R. Rangan, G. Ottoni, J. A. Blome, G. A. Reis, M. Vachharajani, and D. I. August (2004) "RIFLE: An Architectural Framework for User-Centric Information-Flow Security," in Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society Washington, DC, USA, pp. 243–254.

[114] Crandall, J. R. and F. T. Chong (2004) "Minos: Control Data Attack Prevention Orthogonal to Memory Model," in Proceedings of the International Symposium on Microarchitecture, pp. 221–232.

[115] Suh , G . E ., J. W. Lee, D. Zhang, and S. Devadas (2004) "Secure Program Execution via Dynamic Information Flow Tracking," in Proceedings of the conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 85–96.

[116] Chow, J., B. Pfaff, T. Garfinkel, K. Christopher, and M. Rosenblum (2004) "Understanding Data Lifetime via Whole System Simulation," in Proceedings of the 13th USENIX Security Symposium.

[117] Cheng, W., Q. Zhao, B. Yu, and S. Hiroshige (2006) "TaintTrace: E fficient Flow Tracing with Dyanmic Binary Rewriting," in Proceedings of the IEEE Symposium on Computers and Communications (ISCC), pp. 749–754.

[118] Ho, A., M. Fetterman, C. Clark, A. Warfield, and S. Hand (2006) "Practical Taint-Based Protection using Demand Emulation," in Proceedings of the European Conference on Computer Systems (EuroSys), pp. 29–41.

[119] Xu, W., S. Bhatkar, and R. Sekar (2006) "Taint-Enhanced Policy Enforcement: A Practical Approach to Defeat a Wide Range of Attacks," in Proceedings of the USENIX Security Symposium, pp. 121–136.

[120] Lam, L. C. and T. cker Chiueh (2006) "A General Dynamic Information Flow Tracking Framework for Security Applications," in Proceedings of the Annual Computer Security Applications Conference (ACSAC), pp. 463–472.

[121] Saxena, P., R. Sekar, and V. Puranik (2008) “E fficient Fine-Grained Binary Instrumentation with Applications to Taint-Tracking,” in Proceedings of the IEEE/ACM symposium on Code Generation and Optimization (CGO), pp. 74–83.

[122] Haldar, V., D. Chandra, and M. Franz (2005) “Dynamic Taint Propagation for Java,” in Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC), pp. 303–311.

[123] Halfond, W. G., A. Orso, and P. Manolios (2008) “WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation,” IEEE Transactions on Software Engineering, 34(1), pp. 65–81.

[124] Chandra, D. and M. Franz (2007) “Fine-Grained Information Flow Analysis and Enforcement in a Java Virtual Machine,” in Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC).

[125] Nair, S. K., P. N. Simpson, B. Crispo, and A. S. Tanenbaum (2007) Design and Implementation of a Virtual Machine Based Information Flow Control System, Tech. Rep. IR-CS-040, Department of Computer Science, Vrije Universiteit.

[126] Vogt, P., F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna (2007) “Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis,” in Proceedings of the 14th Network & Distributed System Security Symposium.

[127] Yip, A., X. Wang, N. Zeldovich, and M. F. Kaashoek (2009) “Improving Application Security with Data Flow Assertions,” in Proceedings of the ACM Symposium on Opërating Systems Principles.

[128] Ashcraft, K. and D. Engler (2002) “Using Programmer-Written Compiler Extensions to Catch Security Holes,” in Proceedings of the IEEE Symposium on Security and Privacy.

[129] Chen, H., D. Dean, and D. Wagner (2004) “Model Checking One Million Lines of C Code,” in Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS).

[130] Schwarz, B., H. Chen, D. Wagner, G. Morrison, J. West, J. Lin, and W. Tu (2005) “Model Checking an Entire Linux Distribution for Security Violations,” in Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC).

[131] Ball, T., E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S. K. Rajamani, and A. Ustuner (2006) “Thorough Static Analysis of Device Drivers,” in Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys).

[132] Ball, T. and S. K. Rajamani (2001) “Automatically Validating Temporal Safety Properties of Interfaces,” in Proceedings of the 8th International SPIN Workshop on Model Checking of Software.

[133] Ware, M. S. and C. J. Fox (2008) “Securing Java Code: Heuristics and an Evaluation of Static Analysis Tools,” in Proceedings of the Workshop on Static Analysis (SAW).

[134] Hovemeyer, D. and W. Pugh (2004) “Finding Bugs is Easy,” in Proceedings of the 19th annual ACM SIGPLAN conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA).

[135] Livshits, V. B. and M. S. Lam (2005) “Finding Security Vulnerabilities in Java Applications with Static Analysis,” in Proceedings of the 14th USENIX Security Symposium.

[136] Felmetsger, V., L. Cavedon, C. Kruegel, and G. Vigna (2010) “Toward Automated Detection of Logic Vulnerabilities in Web Applications,” in Proceedings of the 19th USENIX Security Symposium.

[137] Jovanovic, N., C. Kruegel, and E. Kirda (2010) “Static Analysis for Detection Taint-style Vulnerabilities in Web Applications,” Journal of Computer Security, 18(5).

[138] Balzarotti, D., M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna (2008) “Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications,” in Proceedings of the IEEE Symposium on Security and Privacy.

[139] Kirda , E., C. Kruegel, G. Banks, G. Vigna, and R. A. Kemmerer (2006) “Behavior-based Spyware Detection,” in Proceedings of the 15th USENIX Security Symposium, pp. 273–288.

[140] Jung, J., A. Sheth, B. Greenstein, D. Wetherall, G. Maganis, and T. Kohno (2008) “Privacy Oracle: A System for Finding Application Leaks with Black Box Di fferential Testing,” in Proceedings of the 15th ACM conference on Computer and Communications Security, ACM New York, NY, USA, pp. 279– 288.

[141] Yumerefendi, A. R., B. Mickle, and L. P. Cox (2007) “TightLip: Keeping Applications from Spilling the Beans,” in Proceedings of the 4th USENIX Symposium on Network Systems Design & Implementation (NSDI), pp. 159–172.

[142] BBC. BBC. “Mobiles hope to be ‘smart wallet’.” [Online] November 21, 2006. http://news.bbc.co.uk/2/hi/technology/6168222.stm.2010. http://www.gartner.com/it/page.jsp?id=1452614.

[143] Canalys. “Google’s Android becomes the world’s leading smart phone platform.” Canalys Corp. [Online] January 31, 2011. http://www.canalys.com/pr/2011/r2011013.html.

[144] Rubin, Andy. @Arubin. Twitter. [Online] 10 19, 2010. https://twitter.com/arubin/status/27808662429.

[145] Hypponen, Mikko. Android. F-Secure Blog. [Online] 11 6, 2007. http://www.f-secure.com/weblog/archives/00001311.html.

[146] Development, “OPT. T-Mobile G1 Hits the UK.” [Online] 10 30, 2008. http://www.opt-development.co.uk/press-office/admin/media-releasefiles/242/G1_London_launch.pdf.

[147] Blitz. Kafan. [Viruses] “Blitz Force released the world’s first mixed-attack system Android.” [Online] 9 10, 2008. http://bbs.kafan.cn/thread-325655-1-1.html.

[148] McAfee. Android_CallAccepter.A. McAfee. [Online] 1 13, 2009. http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=269032.

[149] Android/Radiocutter.A. [Online] 1 13, 2009. http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=269033.

[150] . Android/SilentMutter.A. [Online] 1 13, 2009. http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=269034.

[151] Android/StiffWeather.A. [Online] 4 25, 2009. http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=269036.

[152] SEO Press Release. “Android Mobile Spy Released to Silently Monitor GPS Locations, SMS Messages and Calls.” [Online] November 6, 2009 .http://www.seopressreleases.com/android-mobile-spy-released-silently-monitor-gps-locations-sms-messages-calls/5080.

[153] Retina-X Studios. “Android Mobile Spy Software.” [Online] http://www.mobile-spy.com/android.html.

[154] Vennon, Troy. “Android Malware: A Study of Known and Potential Malware Threats.” [Online] February 24, 2010. http://globalthreatcenter.com/wp-content/uploads/2010/03/Android-Malware-Whitepapër.pdf.

[155] Travis Credit Union. “Phishing scam targeting.” [Online] 1 12, 2010. https://www.traviscu.org/news.aspx?blogmonth=12&blogyear=2009&blogid=112.

[156] F-Secure. “Warning On Possible Android Mobile Trojans.” [Online] 1 11, 2010. http://www.f-secure.com/weblog/archives/00001852.html.

[157] Cluley, Graham. “Banking malware found on Android Marketplace.” [Online] 1 11, 2010. http://nakedsecurity.sophos.com/2010/01/11/banking-malware-android-marketplace/.

[158] Papathanasiou, Christian and Përcoco, Nicholas J. DEF CON. [Online] June 21, 2010. https://www.defcon.org/images/defcon-18/dc-18-presentations/Trustwave-Spiderlabs/DEFCON-18-Trustwave-Spiderlabs-Android-Rootkit-WP.pdf.

[159] Maslennikov, Dennis. "First SMS Trojan for Android." [Online] Kaspërsky, August 10, 2010. http://www.securelist.com/en/blog/2254/First_SMS_Trojan_for_Android.

[160] Dunham, Ken. "Chapter 5: Taxonomy of Mobile Malware." Mobile Malware Attacks and Defense. s.l. : Syngress, 2009.

[161] Symantec. AndroidOS. "Tapsnake: Watching Your Every Move." [Online] August 16, 2010. http://www.symantec.com/connect/blogs/androidostapsnake-watching-your-every-move.

[162] Hypponen, Mikko. "Angry Birds Trojan." [Online] November 13, 2010. http://www.f-secure.com/weblog/archives/00002063.html.

[163] Wyatt, Tim. "Security Alert: Geinimi, Sophisticated New Android Trojan Found in Wild." [Online] December 2010, 2010. https://blog.mylookout.com/2010/12/geinimi_trojan/.

[164] "Geinimi Trojan Technical Analysis." [Online] January 2011, 2011. http://blog.mylookout.com/2011/01/geinimi-trojan-technical-analysis/.

[165] Asrar, Irfan. "The 'Mobile' Little Shop of Horrors." [Online] January 05, 2011. http://www.symantec.com/connect/blogs/mobile-little-shophorrors.

[166] Security Week. "Multiple Variants of Android Malware 'Hong Tou Tou' Surface in China." [Online] February 22, 2011. https://www.securityweek.com/multiple-variants-android-virus-hong-tou-tou-surface-china.\

[167] Aegis Lab. "Security Alert 2011-02-14: New Android Trojan 'ADRD' Was Found in the Wild by Aegislab." [Online] February 14, 2011. http://blog.aegislab.com/index.php?op=ViewArticle&articleId=75&blogId=1.

[168] Ballano, Mario. "Android Threats Getting Steamy." [Online] February 28, 2011. http://www.symantec.com/connect/blogs/android-threatsgetting-steamy.

[169] Lookout. "Update: Security Alert: DroidDream Malware Found in Official Android Market." [Online] March 1, 2011. https://blog.mylookout.com/2011/03/security-alert-malware-found-in-official-android-market-droiddream/.

[170] Lompolo. “Someone just ripped off 21 popular free apps from the market, injëcted root exploits into them and republished.” 50k-200k downloads combined in 4 days. [Online] March 1, 2011. http://www.reddit.com/r/netsec/comments/fvhdw/someone_just_ripped_off_21_popular_free_apps_from/.

[171] Gingrich, Aaron. “The Mother Of All Android Malware Has Arrived: Stolen Apps Released To The Market That Root Your Phone, Steal Your Data, And Open Backdoor.” [Online] March 1, 2011. http://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-hasarrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor/.

[172] Google. “An Update on Android Market Security.” [Online] March 5, 2011. http://googlemobile.blogspot.com/2011/03/update-on-androidmarket-security.html.

[173] Ballano, Mario. “Android.Bgserv Found on Fake Google Security Patch.” [Online] March 9, 2011. http://www.symantec.com/connect/ko/blogs/androidbgserv-found-fake-google-security-patch.

[174] Aegis Lab. “Security Alert 2011-03-04: Yet Another Repackaged Trojan ‘Fake10086’ Leaks User Privacy.” [Online] March 4, 2011. http://blog.aegislab.com/index.php?op=ViewArticle&articleId=81&blogId=1.selectively-blocks-sms-step-by-step/.

[175] Asrar, Irfan. “Android Threat Tackles Piracy Using Austere Justice Measures.” [Online] March 31, 2011. http://www.symantec.com/connect/blogs/android-threat-tackles-piracy-using-austere-justice-measures.

[176] AegisLab, Zsone. “Security Alert 2011-05-11: New SMS Trojan ‘zsone’ was Took Away from Google Market.” [Online] May 11, 2011. http://blog.aegislab.com/index.php?op=ViewArticle&articleId=112&blogId=1.security-alert-zsone-trojan-found-in-android-market/.

[177] Asrar, Irfan. “Android Threat Set to Trigger On the End of Days, or the Day’s End.” [Online] May 25, 2011. http://www.symantec.com/connect/blogs/android-threat-set-trigger-end-days-or-day-s-end.

[178] Apville, Axelle. “Android.Smspacem under the microscope.” [Online] May 30, 2011. http://blog.fortinet.com/android-smspacem-under-themicroscope/.

[179] Wyatt, Tim. “Update: Security Alert: DroidDreamLight, New Malware from the Developërs of DroidDream.” [Online] May 30, 2011. http://

blog.mylookout.com/2011/05/security-alert-droiddreamlight-new-malware-from-the-developërs-of-droiddream/.

[180] AVG Mobilation. “Malware information: DroidDreamLight.” [Online] May 30, 2011. http://www.avgmobilation.com/securitypost_20110601.html#tabs-2.

[181] NetQin. “Security Alert: Fee-Deduction Malware on Android Devices Spotted in the Wild.” [Online] May 30, 2011. http://www.prnewswire.com/news-releases/security-alert-fee-deduction-malware-on-android-devices-spotted-in-the-wild-122822179.html.

[182] AVG Mobilation. “Malware information: BaseBridge.” [Online] May 23, 2011. http://www.avgmobilation.com/securitypost_20110605.html#tabs-2.

[183] Apvrille, Axelle. “Android/DroidKungFu uses AES encryption.” [Online] June 9, 2011. http://blog.fortinet.com/androiddroidkungfu-usesaes-encryption/.

[184] SQLite. SQLite. [Online] http://www.sqlite.org/.

[185] Case, Justin. [Updated] “Exclusive: Vulnerability In Skype For Android Is Exposing Your Name, Phone Number, Chat Logs, And A Lot More.” [Online] April 14, 2011. http://www.androidpolice.com/2011/04/14/exclusive-vulnerability-in-skype-for-android-is-exposing-your-namephone-number-chat-logs-and-a-lot-more/.

[186] Asher, Adrian. “Privacy vulnerability in Skype for Android fixed.” [Online] April 20, 2011. http://blogs.skype.com/security/2011/04/privacy_vulnerability_in_skype_1.html.

[187] “Guide, Android Developërs--Dev. Application Fundamentals.” [Online] http://developer.android.com/guide/topics/fundamentals.html.

[188] android4me. “J2ME port of Google’s Android.” [Online] October 9, 2008. http://code.google.com/p/android4me/downloads/detail?name=AXMLPrinter2.jar&can=2&q=.

[189] dex2jar. “A tool for converting Android’s .dex format to Java’s .class format.” [Online] http://code.google.com/p/dex2jar/.

[190] Decompiler, Java. “Yet another fast java decompiler.” [Online] http://java.decompiler.free.fr/

[191] Gabor, Paller. “MY LIFE WITH ANDROID :-).” [Online] January 9, 2009. http://mylifewithandroid.blogspot.com/2009/01/disassembling-dexfiles.html.

[192] smali. “An assembler/disassembler for Android’s dex format.” [Online] http://code.google.com/p/smali/.

[193] M. Becher, “Security of smartphones at the dawn of their ubiquitousness,” Ph.D. dissertation, University of Mannheim, Oct. 2009

[194] M. Hypponen, “State of Cell Phone Malware in 2007,” 2007, http://www.usenix.org/events/sec07/tech/hypponen.pdf.

[195] Sophos, “Security Threat Report,” 2010. [Online]. Available: http://www.sophos.com/sophos/docs/eng/ papërs/ sophos-security-threat-report-jan-2010-wpna.pdf

[196] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C.Wolf, “Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices,” in Proceedings IEEE Security and Privacy, May 2011.

[197] A. Makhlouf and N. Boudriga, “Intrusion and anomaly detection in wireless networks,” in Handbook of Research on Wireless Security, Y. Zhan, J. Zheng, and M. Ma, Eds. Information Science Publishing, 2008.

[198] K. Haataja, “Security threats and countermeasures in Bluetoothenabled systems,” Ph.D. dissertation, Department of Computer Science, University of Kuopio, 2009.

[199] Y. L. Ho and S.-H. Heng, “Mobile and ubiquitous malware,” in MoMM ’09: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia. NewYork,NY, USA: ACM, 2009, pp. 559–563.

[200] A. Bose, X. Hu, K. G. Shin, and T. Park, “Behavioral detection of malware on mobile handsets,” in MobiSys ’08: Proceeding of the 6th international conference on Mobile systems, applications, and services. New York, NY, USA: ACM, 2008, pp. 225–238.

[201] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. C¸ amtepe, and S. Albayrak, “Monitoring smartphones for anomaly detection,” Mob. Netw. Appl., vol. 14, no. 1, pp. 92–106, 2009.

[202] L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu, “pBMDS: a behaviorbased malware detection system for cellphone devices,” in Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24, 2010. ACM, 2010, pp. 37–48.

[203] M. Becher and F. C. Freiling, “Towards Dynamic Malware Analysis to Increase Mobile Device Security,” in Sicherheit 2008: Sicherheit, Schutz und Zuverl¨assigkeit. Konferenzband der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft f¨ur Informatik e.V. (GI), 2.- 4. April 2008 im Saarbr¨ucker Schloss, ser. LNI, vol. 128. GI, 2008, pp. 423–433.

[204] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta, “On cellular botnets: measuring the impact of malicious devices on a cellular network core,” in CCS ’09: Proceedings of the 16th ACM conference

on Computer and communications security. New York, NY, USA: ACM, 2009, pp. 223–234.

[205] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, “Exploiting open functionality in SMS-capable cellular networks,” in Proceedings of the 12th ACM conference on Computer and communications security, ser. CCS ’05. New York, NY, USA: ACM, 2005, pp. 393–404.

[206] C. Xenakis, “Malicious actions against the GPRS technology,” Journal in Computer Virology, vol. 2, no. 2, pp. 121–133, 2006.

[207] J. R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely, “Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards,” in Proceedings of the 2002 IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2002, pp. 31–.

[208] G. Kambourakis, C. Kolias, S. Gritzalis, and J. H. Park, “DoS attacks exploiting signaling in UMTS and IMS,” Computer Communications, vol. 34, no. 3, pp. 226–235, 2011.

[209] A. Bose and K. G. Shin, “Proactive security for mobile messaging networks,” in WiSe ’06: Proceedings of the 5th ACM workshop on Wireless security. New York, NY, USA: ACM, 2006, pp. 95–104.

[210] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, “A Social Network Based Patching Scheme for Worm Containment in Cellular Networks,” in INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janëiro, Brazil, 2009, pp. 1476–1484.

[211] A. Bose and K. G. Shin, “On Mobile Viruses Exploiting Messaging and Bluetooth Services,” in Securecomm and Workshops, 2006, Sept 2006, pp. 1–10.

[212] U. Meyer and S. Wetzel, “A man-in-the-middle attack on UMTS,” in Proceedings of the 3rd ACM workshop on Wireless security, ser. WiSe ’04. New York, NY, USA: ACM, 2004, pp. 90–97.

[213] M. Khan, A. Ahmed, and A. R. Cheema, “Vulnerabilities of UMTS Access Domain Security Architecture,” in Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. Washington, DC, USA: IEEE Computer Society, 2008, pp. 350–355.

[214] C. Guo, H. J. Wang, and W. Zhu, “Smart-phone attacks and defenses,” in HotNets III. Citeseer, 2004.

[215] J. W. Mickens and B. D. Noble, “Modeling epidemic spreading in mobile environments,” in WiSe ’05: Proceedings of the 4th ACM workshop on Wireless security. New York, NY, USA: ACM, 2005, pp. 77–86.

[216] C. Rhodes and M. Nekovee, “The opportunistic transmission of wireless worms between mobile devices,” Physica A: Statistical Mechanics and its Applications, vol. 387, no. 27, pp. 6837–6844, 2008. [Online]. Available: http://www.science- direct.com/science/article/pii/S0378437108007772

[217] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes, “Can you infect me now?: malware propagation in mobile phone networks,” in WORM ’07: Proceedings of the 2007 ACM workshop on Recurring malcode. New York, NY, USA: ACM, 2007, pp. 61–68.

[218] G. Yan, H. D. Flores, L. Cuellar, N. Hengartner, S. Eidenbenz, and V. Vu, “Bluetooth worm propagation: mobility pattern matters!” in ASIACCS ’07: Proceedings of the 2nd ACM symposium on Information, computer and communications security. New York, NY, USA: ACM, 2007, pp. 32–44.

[219] G. Yan and S. Eidenbenz, “Bluetooth Worms: Models, Dynamics, and Defense Implications,” in ACSAC ’06: Proceedings of the 22nd Annual Computer Security Applications Conference. Washington,DC, USA: IEEE Computer Society, 2006, pp. 245–256.

[220] Y. Bulygin, “Epidemics of Mobile Worms,” Përformance, Computing, and Communications Conference, 2002. 21st IEEE International, vol. 0, pp. 475–478, 2007.

[221] J. W. Mickens and B. D. Noble, “Analytical Models for Epidemics in Mobile Networks,” in WIMOB ’07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. Washington, DC, USA: IEEE Computer Society, 2007, p. 77.

[222] G. Yan and S. Eidenbenz, “Modeling Propagation Dynamics of Bluetooth Worms (Extended Version),” IEEE Transactions on Mobile Computing, vol. 8, pp. 353–368, 2009.

[223] J. Su, K. K. W. Chan, A. G. Miklas, K. Po, A. Akhavan, S. Saroiu, E. de Lara, and A. Goel, “A preliminary investigation of worm infections in a bluetooth environment,” in WORM ’06: Proceedings of the 4th ACM workshop on Recurring malcode. NewYork,NY, USA: ACM, 2006, pp. 9–16.

[224] M. Becher, F. C. Freiling, and B. Leider, “On the Effort to Create Smartphone Worms in Windows Mobile,” in Information Assurance and Security Workshop, 2007. IAW ’07. IEEE SMC, june 2007, pp. 199–206.

[225] A. Lelli, “A Smart Worm for a Smartphone WinCE.PmCryptic.A,” 2009. [Online]. Available: http://www.symantec.com/connect/blogs/smart-worm-smartphone-wincepmcryptica

[226] A. R. Flø and A. Jøsang, “Consequences of botnets spreading to mobile devices,” in 14th Nordic Conference on Secure IT Systems, 2009, pp. 37–43.

[227] K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee, “Evaluating Bluetooth as a medium for botnet command and control,” in Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment, ser. DIMVA’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 61–80.

[228] K. G. S. Yuanyuan Zeng, Xin Hu, “How to Construct a Mobile Botnet?” in The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010), 2010.

[229] C. Mulliner and J.-P. Seifert, “Rise of the iBots: Owning a telco network,” in Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on, Oct 2010, pp. 71–80.

[230] P. A. Porras, H. Sa¨ıdi, and V. Yegneswaran, “An Analysis of the iKee.B iPhone Botnet,” in Security and Privacy in Mobile Information and Communication Systems - Second International ICST Conference, MobiSec 2010, Catania, Sicily, Italy, May 27-28, 2010, Revised Selected Papërs, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. U. Schmidt, G. Russello, A. Lioy, N. R. Prasad, and S. Lian, Eds., vol. 47. Springer, 2010, pp. 141–152.

[231] A. Apvrille, “Symbian Worm Yxes: Towards Mobile Botnets?” in The 19th EICAR Annual Conference, May 2010, pp. 31–54.

[232] M. Ballano, “Android Threats Getting Steamy,” Feb 2011. [Online]. Available: http://www.symantec.com/connect/blogs/android-threats-getting-steamy

[233] M. Becher, “Security of Smartphones at the Dawn of their Ubiquitousness,” Ph.D. dissertation, Universit¨at Mannheim, 2009.

[234] Techie Buzz, “Android Data Theft Vulnerability Detailed,” 2011. [Online]. Available: http://techie-buzz.com/mobile-news/android-data-theft-vulnerability-detailed.html

[235] L. Whitney, “Apple sued over privacy in iPhone, iPad apps,” 2011. [Online]. Available: http://news.cnet.com/8301-13579\3-20026677-37.html

[236] N. Seriot, “iPhone Privacy,” Black Hat DC, 2010. [Online]. Available: http://seriot.ch/resources/talks papërs/iPhonePrivacy.pdf

[237] S. Bhatt, R. Sion, and B. Carbunar, “A përsonal mobile DRM manager for smartphones,” Computers & Security, vol. 28, no. 6, pp. 327–340, 2009.

[238] R. P. Minch, “Privacy Issues in Location-Aware Mobile Devices,” in Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS’04) - Track 5 - Volume 5. Washington, DC, USA: IEEE Computer Society, 2004, pp. 50 127.2–.

[239] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones,” in Proceedings of the 9th USENIX conference on Opërating systems design and implementation, ser. OSDI’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[240] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, “PiOS: Detecting Privacy Leaks in iOS Applications,” in Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, February 2011.

[241] S. Whitehead, J. Mailley, I. Storer, J. McCardle, G. Torrens, and G. Farrell, “IN SAFE HANDS: A Review of Mobile Phone Antitheft Designs,” European Journal on Criminal Policy and Research, vol. 14, pp. 39–60, 2008.

[242] A. Portnoy, “Pwn2Own 2010,” 2010. [Online]. Available: http://dvlabs.tippingpoint.com/blog/2010/ 02/15/pwn2own2010

[243] L. Cai, S. Machiraju, and H. Chen, “Defending against sensor-sniffing attacks on mobile phones,” in Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds, ser. MobiHeld ’09. New York, NY, USA: ACM, 2009, pp. 31–36.

[244] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, “Stealthy video capturer: a new video-based spyware in 3G smartphones,” in Proceedings of the second ACM conference on Wireless network security, ser. WiSec ’09. New York, NY, USA: ACM, 2009, pp. 69–78.

[245] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, “Soundminer: A Stealthy and Context-Aware Sound Trojan for Smartphones,” in Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS), Feb. 2011.

[246] W. Enck, M. Ongtang, and P. McDaniel, “On lightweight mobile phone application certification,” in CCS ’09: Proceedings of the 16th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2009, pp. 235–245.

[247] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, “Semantically Rich Application-Centric Security in Android,” in Computer Security Applications Conference, 2009. ACSAC ’09. Annual, dec. 2009, pp. 340–349.

[248] E. Naone, “SMS of Death Could Crash Many Mobile Phones,” 2011. [Online]. Available: http://www.technologyreview.com/printerfriendly article.aspx?id =27021

[249] H. Kim, J. Smith, and K. G. Shin, “Detecting energy-greedy anomalies and mobile malware variants,” in MobiSys’08: Proceeding of the 6th international conference on Mobile systems, applications, and services. New York, NY, USA: ACM, 2008, pp. 239–252.

[250] L. Liu, X. Zhang, G. Yan, and S. Chen, “Exploitation and threat analysis of open mobile devices,” in Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ser. ANCS ’09. New York, NY, USA: ACM, 2009, pp. 20–29.

[251] R. Racic, D. Ma, and H. Chen, “Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone’s Battery,” in Securecomm and Workshops, 2006, aug. 2006, pp. 1–10.

[252] D. Johnston and J. Walker, “Overview of IEEE 802.16 security,” Security Privacy, IEEE, vol. 2, no. 3, pp. 40–48, may-june 2004.

[253] T. Engel, “Remote SMS/MMS Denial of Service - Curse Of Silence for Nokia S60 phones,” Dec 2008. [Online]. Available: http://berlin.ccc.de/∼tobias/cursesms.txt

[254] C. Mulliner and C. Miller, “Injëcting SMS messages into smart phones for security analysis,” in WOOT’09: Proceedings of the 3rd USENIX conference on Offensive technologies. Berkeley,CA,USA: USENIX Association, 2009, pp. 5–5.

[255] C. Xenakis and L. Merakos, “Vulnerabilities and Possible Attacks Against the GPRS Backbone Network,” in Critical Information Infrastructures Security, ser. Lecture Notes in Computer Science, J. Lopez, Ed. Springer Berlin / Heidelberg, 2006, vol. 4347, pp.262–272.

[256] Android.Bgserv Found on Fake Google Security Patch. http://www.symantec.com/connect/blogs/androidbgservfound-fake-google-security-patch.

[257] WAPS. http://www.waps.cn/.

[258] GGTracker Technical Tear Down. http://blog.mylookout.com/wp-content/uploads/2011/06/GGTracker-Teardown Lookout-Mobile-Security.pdf.

[259] Malicious QR Codes Pushing Android Malware. https://www.securelist.com/en/blog/208193145/Its time for malicious QR codes.

[260] First SpyEye Attack on Android Mobile Platform now in the Wild. https://www.trusteer.com/blog/first-spyeye-attackandroid-mobile-platform-now-wild.

[261] ZeuS-in-the-Mobile - Facts and Theories. http://www.securelist.com/en/analysis/204792194/ZeuS in the Mobile Facts and Theories.

[262] QR code. http://en.wikipedia.org/wiki/QR code.

[263] Using QR tags to Attack SmartPhones (Attaging). http://kaoticoneutral.blogspot.com/2011/09/using-qr-tags-toattack-smartphones 10.html.

[264] G. Portokalidis, P. Homburg, N. FitzRoy-Dale, K. Anagnostakis, and H. Bos, "Protecting smart phones by means of execution replication," Vrije Universiteit Amsterdam, Tech. Rep., 2009.

[265] A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets," in MobiSys '08: Proceeding of the6th international conference on Mobile systems, applications, andservices. New York, NY, USA: ACM, 2008, pp. 225–238.

[266] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. C¸ amtepe, and S. Albayrak, "Monitoring smartphones for anomaly detection," Mob.Netw. Appl., vol. 14, no. 1, pp. 92–106, 2009.

[267] L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu, "pBMDS: a behaviorbased malware detection system for cellphone devices," in Proceedingsof the Third ACM Conference on Wireless Network Security,WISEC 2010, Hoboken, New Jersey, USA, March 22-24, 2010. ACM, 2010, pp. 37–48.

[268] M. Becher and F. C. Freiling, "Towards Dynamic Malware Analysis to Increase Mobile Device Security," in Sicherheit 2008: Sicherheit,Schutz und Zuverl¨assigkeit. Konferenzband der 4. Jahrestagung desFachbereichs Sicherheit der Gesellschaft f¨ur Informatik e.V. (GI), 2.-4. April 2008 im Saarbr¨ucker Schloss, ser. LNI, vol. 128. GI, 2008, pp. 423–433.

[269] A. Bose and K. G. Shin, "Proactive security for mobile messaging networks," in WiSe '06: Proceedings of the 5th ACM workshop onWireless security. New York, NY, USA: ACM, 2006, pp. 95–104.

[270] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, "A Social Network Based Patching Scheme for Worm Containment in Cellular Networks," in INFOCOM 2009. 28th IEEE International Conference onComputer Communications, Joint Conference of the IEEE Computerand Communications Societies, 19-25 April 2009, Rio de Janëiro,Brazil, 2009, pp. 1476–1484.

[271] A. Bose and K. G. Shin, “On Mobile Viruses Exploiting Messaging and Bluetooth Services,” in Securecomm and Workshops, 2006, Sept 2006, pp. 1–10.

[272] L. Cai, S. Machiraju, and H. Chen, “Defending against sensor-sniffing attacks on mobile phones,” in Proceedings of the 1st ACM workshopon Networking, systems, and applications for mobile handhelds, ser. MobiHeld ’09. New York, NY, USA: ACM, 2009, pp. 31–36.

[273] W. Enck, M. Ongtang, and P. McDaniel, “On lightweight mobile phone application certification,” in CCS ’09: Proceedings of the 16thACM conference on Computer and communications security. New York, NY, USA: ACM, 2009, pp. 235–245.

[274] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, “Semantically Rich Application-Centric Security in Android,” in ComputerSecurity Applications Conference, 2009. ACSAC ’09. Annual, dec. 2009, pp. 340–349.

[275] H. Kim, J. Smith, and K. G. Shin, “Detecting energy-greedy anomalies and mobile malware variants,” in MobiSys ’08: Proceeding ofthe 6th international conference on Mobile systems, applications, andservices. New York, NY, USA: ACM, 2008, pp. 239–252.

[276] C. Mulliner and C. Miller, “Injëcting SMS messages into smart phones for security analysis,” in WOOT’09: Proceedings of the 3rdUSENIX conference on Offensive technologies. Berkeley,CA,USA: USENIX Association, 2009, pp. 5–5.

[277] McAfee, “WaveSecure,” 2011. [Online]. Available: https://www.wavesecure.com/

[278] Norton, “Norton Mobile Security Lite,” 2011. [Online]. Available:http://us.norton.com/mobile-security/

[279] IIT-CNR, “iCareMobile,” 2011. [Online]. Available: http://icaremobile.iit.cnr.it/

[280] BullGuard Ltd, “BullGuard Mobile Security 10,” 2011. [Online].Available: http://www.bullguard.com

[281] Kaspërsky Lab ZAO, “Kaspërsky Mobile Security 9,” 2011. [Online].Available: http://www.kaspërsky.com/kaspërsky mobile security

[282] Lookout, “Lookout Mobile Security,” 2011. [Online]. Available:https://www.mylookout.com

[283] B. Sun, Y. Xiao, and K. Wu, “Intrusion Detection in CellularMobile Networks,” in Wireless Network Security, ser. Signals and Communication Technology, Y. Xiao, X. S. Shen, and D.-Z. Du, Eds. Springer US, 2007, pp. 183–210.

[284] G. W. Chow and A. Jones, “A Framework for Anomaly Detection inOKL4-Linux Based Smartphones,” in Australian Information SecurityManagement Conference, 2008.

[285] A. Shabtai, U. Kanonov, and Y. Elovici, “Intrusion detection formobile devices using the knowledge-based, temporal abstraction method,” J. Syst. Softw., vol. 83, no. 8, pp. 1524–1537, 2010.

[286] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss,“”Andromaly”: a behavioral malware detection framework for android devices,” Journal of Intelligent Information Systems,pp. 1–30, 2011, 10.1007/s10844-010-0148-x. [Online]. Available: http://dx.doi.org/10.1007/s10844-010-0148-x

[287] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, “Crowdroid:behavior-based malware detection system for Android,” in Proceedings of the 1st ACM workshop on Security and privacyin smartphones and mobile devices, ser. SPSM ’11. NewYork, NY, USA: ACM, 2011, pp. 15–26. [Online]. Available:http://doi.acm.org/10.1145/2046614.2046619

[288] G. A. Jacoby, R. Marchany, and N. J. D. IV, “How Mobile HostBatteries Can Improve Network Security,” IEEE Security and Privacy,vol. 4, pp. 40–49, 2006.

[289] L. Liu, G. Yan, X. Zhang, and S. Chen, “VirusMeter: PreventingYour Cellphone from Spies,” in RAID ’09: Proceedings of the 12thInternational Symposium on Recent Advances in Intrusion Detection. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 244–264.

[290] Q. Yan, R. H. Deng, Y. Li, and T. Li, “On the Potential of Limitationoriented Malware Detection and Prevention Techniques on Mobile Phones,” International Journal of Security and Its Applications(IJSIA), vol. 4(1), pp. 21–30, Jan 2010.

[291] D. Venugopal, G. Hu, and N. Roman, “Intelligent virus detection on mobile devices,” in PST ’06: Proceedings of the 2006 InternationalConference on Privacy, Security and Trust. New York, NY, USA: ACM, 2006, pp. 1–4.

[292] L. Xie, H. Song, T. Jaeger, and S. Zhu, “A systematic approach for cell-phone worm containment,” in WWW ’08: Proceeding of the 17thinternational conference on World Wide Web. NewYork,USA: ACM, 2008, pp. 1083–1084.

[293] L. Xie, X. Zhang, A. Chaugule, T. Jaeger, and S. Zhu, “Designing System-Level Defenses against Cellphone Malware,” in SRDS ’09:Proceedings of the 2009 28th IEEE International Symposium on ReliableDistributed Systems. Washington, DC, USA: IEEE Compute Society, 2009, pp. 83–90.

[294] G. Zyba, G. M. Voelker, M. Liljenstam, A. M´ehes, and P. Johansson, “Defending Mobile Phones from Proximity Malware,” in INFOCOM2009. 28th IEEE International Conference on Computer Communications,Joint Conference of the IEEE Computer and CommunicationsSocieties, 19-25 April 2009, Rio de Janëiro, Brazil. IEEE, 2009, pp. 1503–1511.

[295] C. Bauckhage, T. Alpcan, and A.-D. Schmidt, “A Probabilistic Diffusion Scheme for Anomaly Detection on Smartphones,” in Proceedingsof the Fourth International Workshop in Information Security Theoryand Practice 2010 (WISTP’10), ser. Lecture Notes in Computer Science. Springer, 2010, pp. 31–46.

[296] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, “A behavioral approach to worm detection,” in WORM ’04: Proceedingsof the 2004 ACM workshop on Rapid malcode. NewYork,NY, USA: ACM, 2004, pp. 43–53.

[297] A. Castrucci, F. Martinelli, P. Mori, and F. Ropërti, “Enhancing Java ME Security Support with Resource Usage Monitoring,” in Informationand Communications Security, 10th International Conference, ICICS 2008, Birmingham, UK, October 20-22, 2008, Proceedings, ser. Lecture Notes in Computer Science, L. Chen, M. D. Ryan, and G. Wang, Eds., vol. 5308. Springer, 2008, pp. 256–266.

[298] N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan, “Securityby- Contract: Toward a Semantics for Digital Signatures on Mobile Code,” in Public Key Infrastructure, 4th European PKI Workshop:Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June28-30, 2007, Proceedings, ser. Lecture Notes in Computer Science, vol. 4582. Springer, 2007, pp. 297–312.

[299] N. Dragoni, F. Martinelli, F. Massacci, P. Mori, C. Schaefer, T. Walter, and E. Vetillard, “Security-by-Contract (SxC) for Software and Services of Mobile Systems,” in At your service: Service Engineeringin the Information Society Technologies Program. MIT press, 2008.

[300] G. Costa, N. Dragoni, A. Lazouski, F. Martinelli, F. Massacci, and I. Matteucci, “Extending Security-by-Contract with Quantitative Trust on Mobile Devices,” Complex, Intelligent and Software IntensiveSystems, International Conference, vol. 0, pp. 872–877, 2010.

[301] J. Cheng, S. H. Wong, H. Yang, and S. Lu, “SmartSiren: virus detection and alert for smartphones,” in MobiSys ’07: Proceedingsof the 5th international conference on Mobile systems, applicationsand services. New York, NY, USA: ACM, 2007, pp. 258–271.

[302] E. V. Ruitenbeek, T. Courtney, W. H. Sanders, and F. Stevens, “Quantifying the Effectiveness of Mobile Phone Virus Response Mechanisms,” in DSN

’07: Proceedings of the 37th Annual IEEE/IFIPInternational Conference on Dependable Systems and Networks. Washington, DC, USA: IEEE Computer Society, 2007, pp. 790–800.

[303] M. Miettinen and P. Halonen, “Host-Based Intrusion Detection for Advanced Mobile Devices,” in AINA ’06: Proceedings of the 20thInternational Conference on Advanced Information Networking andApplications. Washington, DC, USA: IEEE Computer Society, 2006, pp. 72–76.

[304] C. Mulliner, G. Vigna, D. Dagon, and W. Lee, “Using Labeling to Prevent Cross-Service Attacks Against Smart Phones,” in Proceedingsof the Conference on Detection of Intrusions and Malware, andVulnerability Assessment (DIMVA), ser. LNCS, vol. 4064. Berlin, Germany: Springer, July 2006, pp. 91–108.

[305] M. Becher and R. Hund, “Kernel-Level Interception and Applications on Mobile Devices,” Department for Mathematics and Computer Science, University of Mannheim, Tech. Rep. TR-2008-003, 2009.

[306] A.-D. Schmidt, J. H. Clausen, S. A. Camtepe, and S. Albayrak, “Detecting Symbian OS Malware through Static Function Call Analysis,” in Proceedings of the 4th IEEE International Conference on Maliciousand Unwanted Software (Malware 2009). IEEE, 2009, pp. 15–22.

[307] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. H. Clausen, O. Kiraz, K. A. Y¨uksel, S. A. C¸ amtepe, and S. Albayrak, “Static Analysis of Executables for Collaborative Malware Detection on Android,” in Proceedings of IEEE International Conference on Communications,ICC 2009, Dresden, Germany, 14-18 June 2009. IEEE, 2009, pp. 1–5.

[308] T. Isohara, K. Takemori, and I. Sasase, “Anomaly Detection on Mobile Phone Based Opërational Behavior,” IPSJ Digital Courier, vol. 4, no. 0, pp. 9–17, 2008.

[309] S. Zahid, M. Shahzad, S. Khayam, and M. Farooq, “Keystroke-Based User Identification on Smart Phones,” in Recent Advances in IntrusionDetection, ser. Lecture Notes in Computer Science, E. Kirda, S. Jha, and D. Balzarotti, Eds. Springer Berlin / Heidelberg, 2009, vol. 5758, pp. 224–243.

[310] T. S. Yap and H. T. Ewe, “A Mobile Phone Malicious Software Detection Model with Behavior Checker,” in Web and CommunicationTechnologies and Internet-Related Social Issues - HSI 2005, 3rdInternational Conference on Human.Society@Internet, Tokyo, Japan,July 27-29, 2005, Proceedings, ser. Lecture Notes in Computer Science, vol. 3597. Springer, 2005, pp. 57–65.

[311] C. Mulliner and G. Vigna, “Vulnerability Analysis of MMS User Agents,” in Computer Security Applications Conference, 2006. ACSAC’06. 22nd Annual, dec. 2006, pp. 77–88.

[312] G. Yan, S. Eidenbenz, and E. Galli, “SMS-Watchdog: Profiling Social Behaviors of SMS Users for Anomaly Detection,” in RAID ’09:Proceedings of the 12th International Symposium on Recent Advancesin Intrusion Detection. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 202–223.

[313] S. M. Habib, C. Jacob, and T. Olovsson, “An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones,” Journal of Networks, vol. 4, no. 10, pp. 968–975, 2009.

[314] A.-D. Schmidt, H.-G. Schmidt, J. Clausen, K. A. Yksel, O. Kiraz, A. Camtepe, and S. Albayrak, “Enhancing Security of Linuxbased Android Devices,” in Proceedings of 15th International LinuxKongress. Lehmann, Oct 2008.

[315] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli, “Smartphone Malware Evolution Revisited: Android Next Target?” in Proceedings of the 4thIEEE International Conference on Malicious and Unwanted Software(Malware 2009). IEEE, 2009, pp. 1–7.

[316] A.-D. Schmidt, R. Bye, H.-G. Schmidt, K. A. Yksel, O. Kiraz, J. Clausen, K. Raddatz, A. Camtepe, and S. Albayrak, “Monitoring Android for Collaborative Anomaly Detection: A First Architectural Draft,” Technische Universit¨at Berlin - DAI-Labor, Tech. Rep. TUB-DAI 08/08-02, Aug 2008. [Online]. Available: http://www. dai-labor.de/fileadmin/files/publications/080 8-02 DAI TechReport Monitoring Android.pdf

[317] W. Enck, M. Ongtang, and P. McDaniel, “Understanding Android Security,” IEEE Security and Privacy, vol. 7, pp. 50–57, January 2009.

[318] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, “Google Android: A Comprehensive Security Assessment,” IEEESecurity and Privacy, vol. 8, pp. 35–44, 2010.

[319] A. Shabtai, Y. Fledel, and Y. Elovici, “Securing Android-Powered Mobile Devices Using SELinux,” IEEE Security and Privacy, vol. 8, pp. 36–44, May 2010.

[320] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, “A study of Android application security,” in Proceedings of the 20thUSENIX conference on Security, ser. SEC’11. Berkeley, CA, USA: USENIX Association, 2011, pp.

21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028088

[321] S. Dai, Y. Liu, T. Wang, T. Wei, and W. Zou, “Behavior-Based Malware Detection on Mobile Phone,” in Wireless CommunicationsNetworking and Mobile Computing (WiCOM), 2010 6th InternationalConference on, Sept 2010, pp. 1–4.\

[322] Y. Ikebe, T. Nakayama, M. Katagiri, S. Kawasaki, H. Abe, T. Shinagawa, and K. Kato, “Efficient Anomaly Detection System for Mobile Handsets,” in SECURWARE ’08: Proceedings of the 2008 SecondInternational Conference on Emerging Security Information, Systemsand Technologies. Washington, DC, USA: IEEE Computer Society, 2008, pp. 154–160.

[323] X. Zhang, O. Aciic¸mez, and J.-P. Seifert, “A trusted mobile phone reference architecture via secure kernel,” in STC ’07: Proceedings ofthe 2007 ACM workshop on Scalable trusted computing. NewYork, NY, USA: ACM, 2007, pp. 7–14.

[324] D. Muthukumaran, A. Sawani, J. Schiffman, B. M. Jung, and T. Jaeger, “Measuring integrity on mobile phone systems,” in SACMAT’08: Proceedings of the 13th ACM symposium on Access controlmodels and technologies. New York, NY, USA: ACM, 2008, pp. 155–164.

[325] X. Zhang, O. Aciic¸mez, and J.-P. Seifert, “Building Efficient Integrity Measurement and Attestation for Mobile Phone Platforms,” in Securityand Privacy in Mobile Information and Communication Systems,First International ICST Conference, MobiSec 2009, Turin, Italy, June3-5, 2009, Revised Selected Papërs, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 17. Springer, 2009, pp. 71–82.

[326] Thahn, L., (2013) “Analysis of Malware Families on Android Mobiles: Detection Characteristics Recognizable by ordinary Phone Users and How to Fix It” Journal of Information Security 213-224 https://www.scirp.org/journal/jis/

[327] F - Secure web site: https://www.f-secure.com/v descs/trojan_android_basebridge.shtml#technicaldetails.

[328] Zhou, Y., Jiang, X. (2011) “An analysis of the AnserverBot Trojan” NetQin U.S Security Reseatch Center.

[329] Anity Labs (2011) “ Analysis Report on Android Trojan Hong TouTou (ADRD)”

[330] J. Grossschadl, T. Vejda, and D. Page, “Reassessing the TCG specifications for trusted computing in mobile and embedded systems,” in Proceedings of

the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust. Washington, DC, USA: IEEE Computer Society, 2008, pp. 84–90.

[331] McAfee https://home.mcafee.com/virusinfo/virusprofile.aspx?key=522281#none

[332] Introduction to System Calls : http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html

[333] M.Aranitasi, G.Daci, I.Tafa, "Today's Security Threats on Android Operating System" International Journal of Computer Science and Management Studies (IJCSMS) April 2015 *ISSN: 2231 - 5268*

[334] M.Aranitasi G.Daci, E.Bejko, I.Tafa "Parallelization algorithm for Android OS using OpenMP" International Journal of Computer Science and Management Studies (IJCSMS) April 2015 *ISSN: 2231 – 5268*

[335] M. Aranitasi E. Sheme M.Ibro – "Online identification of business users." International Conference ISTI'2012, 8-9 June 2012, UT – Tirana ISBN 978-9995-6377-8-1

[336] M.Aranitasi. L.Jani – "The Kernel Ring. Self-tuning Linux Kernel using Support Vector Machines". 5th International Conference ICT Innovations Ohrid 2013 ISSN 1857 7288

[337] M. Aranitasi E. Sheme M.Ibro – "Online identification of bussines users" International Journal of Science, Innovation and New Technology June 2012 ISSN 2223-2257

[338] https://stuff.mit.edu/afs/sipb/project/linux/kernel/doc/syscall/syscall.tex

[339] M.Aranitasi M.Ibro, B.Shehu – "Using PKI to increase the security of internet Transactions" 8th South East European Doctoral Student Conference (DSC 2013) Thessaloniki Greece ISBN 978-960-9416-06-1