Markus Muranen

Android-käyttöjärjestelmän tietoturvauhat ja niiden torjuminen

Tietotekniikan kandidaatintutkielma

29. huhtikuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Markus Muranen

Yhteystiedot: markus.p.i.muranen@student.jyu.fi

Työn nimi: Android-käyttöjärjestelmän tietoturvauhat ja niiden torjuminen

Title in English: Security threats in Android and how to defend from them

Työ: Kandidaatintutkielma

Sivumäärä: 28+0

Tiivistelmä: Googlen Android on noussut suosituimmaksi mobiilikäyttöjärjestelmäksi 2010-luvulla. Mobiililaitteet sisältävät runsaasti arkaluontoista tietoa käyttäjästään ja muodostavat näin ollen houkuttelevan kohteen hyökkääjille. Tämän kandidaatintutkielman tarkoituksena on selvittää, mitä uhkia ja haavoittuvuuksia Androidiin liittyy ja kuinka niitä voidaan torjua.

Avainsanat: mobiililaitteet, Android, sovellusten tietoturva, haavoittuvuudet

Abstract: Google's Android has grown to be the most popular mobile operating system in the 2010s. Mobile devices contain huge amounts of sensitive data about the user which makes them a tempting target for attackers. The purpose of this bachelor's thesis is to find out, what kind of threats and vulnerabilities are associated with Android and how can one defend himself succesfully from them.

Keywords: mobile devices, Android, software security, vulnerabilities

Kuviot

Taulukot

Sisältö

1 Johdanto

Älypuhelimet ja tabletit ovat 2010-luvun aikana integroituneet osaksi jokaista elämämme osa-aluetta. Nämä mobiililaitteet ja niiden sovellukset ovat muuttaneet maailmaa ja vallankumous jatkuu yhä. Moni osapuoli haluaa hyötyä tästä murroksesta ja kilpailu älypuhelinten käyttäjistä käy kuumana valmistajien välillä. Ostaessaan älypuhelimen kuluttaja tulee tehneeksi valinnan laitteen lisäksi myös käyttöjärjestelmän osalta. Käyttöjärjestelmien markkinaosuuksista kilpailevat niin Applen iOS, Googlen Android kuin Microsoftin Windows Phonekin. Mitä enemmän hyödynnämme mobiililaitteiden tarjoamia mahdollisuuksia elämässämme, sitä enemmän altistamme itseämme myös niihin kohdistuville uhille. Toisin kuin tietokonemaailman aikoinaan vallanneet haittaohjelmat, muodostaa käyttäjästään arkaluontoisia tietoja runsaasti sisältävä älypuhelin haittaohjelmille vieläkin houkuttelevamman kohteen. Mobiilisovellusten kautta toimitetaan nykyään yhä useammin mm. pankkiasioita ja maksetaan verkkokauppaostoksia. Mobiililaitteet sisältävät runsaasti arkaluontoista tietoa, kuten sähköpostin ja muiden verkkopalveluiden salasanoja ja käyttäjätunnnuksia sekä pikaviestinten kuten WhatsAppin viestihistoriaa. Näillä tiedoilla voidaan mobiililaitteen käyttäjästä muodostaa hyvinkin tarkka profiili.

Android on Googlen vuonna 2007 julkaisema mobiilikäyttöjärjestelmä, joka on noussut 2010-luvulla suurimmaksi mobiilikäyttöjärjestelmäksi. Gartnerin helmikuussa 2016 julkaiseman raportin mukaan Android-käyttöjärjestelmän markkinaosuus älypuhelimista oli 80,7 prosenttia vuoden 2015 viimeisellä neljänneksellä. Vuonna 2015 kaksi suurinta Android-älypuhelinvalmistajaa, Samsung ja Huawei, myivät yhteensä yli 424 miljoonaa Android-älypuhelinta. (Gartner 2016) Android perustuu avoimeen lähdekoodiin, ja sen käyttäjät pystyvät lataamaan puhelimiinsa sovelluksia hyvin monenlaisista lähteistä. Toimintamalli on erilainen kuin esimerkiksi Applella, jonka ympäristö on suljetumpi. Avoimuus on kuitenkin avannut ovia myös haittaohjelmille, ja Android-käyttäjän kiusana ovatkin hyvin monenlaiset haavoittuvuudet ja tietoturvauhat. Erilaisia hyökkäystyyppejä käyttäjän tietoturvaa kohtaan on lukuisia. Kehittäjien käytettävissä olevien suojausprotokollien implementointi

saattaa olla hyvinkin puuttellista ja altistaa sovelluksen useille erilaisille hyökkäystyypeille. Myöskin Android-laitteiden suuri määrä tekee siitä houkuttelevan kohteen hyökkääjille.

Android on mobiilikäyttöjärjestelmänä vasta elinkaarensa alussa. Sen haavoittuvuuksien tutkiminen ja niihin varautuminen on nyt erityisen ajankohtaista, mikä tekee siitä tutkimuskohteena houkuttelevan. Tutkielman tavoitteena on kartoittaa erilaisia haavoittuvuuksia ja kuinka niitä voidaan käyttää hyväksi. Tutkin myös kuinka näitä uhkia voidaan torjua ja Androidin tietoturvaa parantaa. Koska aiheesta on tehty jo jonkin verran tutkimusta, haluan myös selvittää, miten aiemmissa tutkimuksissa on kategorisoitu ja jaoteltu Androidiin kohdistuvia uhkia.

Toisessa luvussa kerron valitsemastani tutkimusmenetelmästä ja siihen liittyvistä ratkaisuista. Kolmannessa luvussa kuvailen Androidin rakennetta ja neljännessä luvussa käyn läpi Androidiin kohdistuvia uhkia ja haavoittuvuuksia. Viidennessä luvussa kuvailen uhkien ja haavoittuvuuksien torjuntakeinoja ja ehdotuksia Androidin tietoturvan parantamiseksi. Kuudennessa luvussa yhteenvedän tutkimukseni tulokset.

2 Kirjallisuuskartoitus

Kirjallisuuskartoitukseni on luonteeltaan systemaattinen, ja tavoitteena on kartoittaa erilaisia Androidiin kohdistuvia tietoturvauhkia ja ratkaisuja niiden estämiseksi. Tutkimusmetodina käytetään pääosin kvalitatiivisia menetelmiä ja hakusanoina toimivat Android, android security, android vulnerabilities ja android security issues. Systemaattinen kirjallisuuskatsaus on tiivistelmä tietyn aihepiirin aiempien tutkimusten olennaisesta sisällöstä. Systemaattinen kirjallisuuskartoitus on myös hyvä tapa esittää tutkimusten tuloksia tiiviissä muodossa. (Salminen 2011, s. 9.) Kirjallisuuskartoituksen hakukoneena toimii pääasiassa Google Scholar. Löysin artikkeleita myös backward ja forward searchin avulla. Suurin osa Androidin tietoturvaan liittyvästä tutkimuksesta nojautuu hyvin vahvasti johonkin aiempaan laajan tutkimuksen artikkeliin, kuten Fahl ym. (2012) tekemään tutkimukseen tietoliikenteen suojaukseen liittyvästä tietoturvasta.

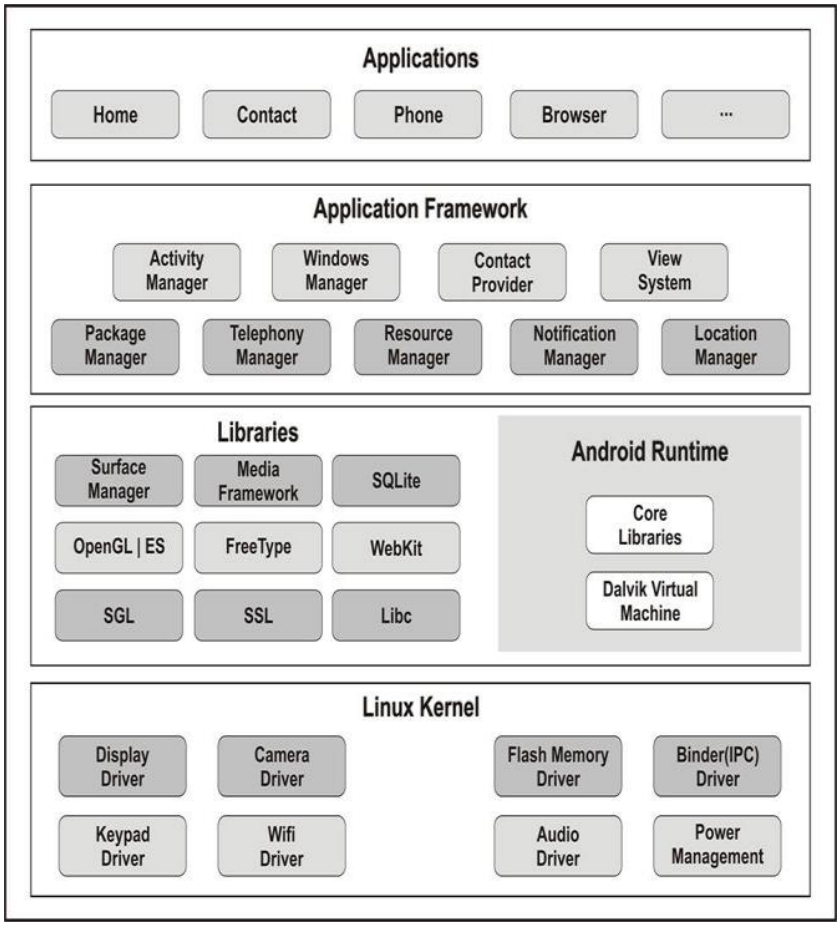
Koska Android on mobiilikäyttöjärjestelmänä kasvattanut suosionsa 2010-luvun aikana, rajasin lähteiden hakua niin, että otin mukaan vain 2010-luvulla ilmestyneet artikkelit. Vanhimmat lähteeni ovat vuodelta 2011 ja uusimmat vuodelta 2015. Monet hieman vanhemmissakin artikkeleissa esitetyt tietoturvaan liittyvät haavoittuvuudet ovat edelleen ajankohtaisia. Kirjallisuuskartoitusta tehdessä kävi selväksi, että tutkimusta Androidin tietoturvaan liittyvistä uhista ja ratkaisusta on tehty paljon. Tiedeyhteisö tuntuu olevan hyvin perillä Androidiin liittyvistä tietoturvauhista. Suurimmat eroavaisuudet eri artikkeleissa liittyivät siihen, kuinka Androidin haavoittuvuuksia ja haavoittuvuuksien torjuntakeinoja on kategorisoitu.

Taulukko 1. Kirjallisuuskartoituksen tuloksena löydetty lähdekirjallisuus

Android security permissions - can we trust them?	Kraxberger, Lackner, Leibetseder, Marsalek, Gissing, Orthacker, Prevenhueber ja Teufl (2011)
Android malware past, present, and future	Castillo (2011)
Why eve and mallory love android: an analysis of android SSL (in)security	Baumgärtner, Fahl, Freisleben, Harbach, Muders ja Smith (2012)
How secure is your smartphone: An analysis of smartphone security mechanisms	Khan, Nauman, Othman ja Shahrulniza (2012)
An Empirical Evaluation of the Android Security Framework	Armdo, Merlo ja Verderame (2013)
Critical Evaluation of Security Framework in Android Applications: Android-level security and Application-level security	Mall ja Gupta (2014)
Analysis of Android vulnerabilities and modern exploitation techniques	Desmukh, Patil ja Shewale (2014)
Android platform security issues	Padaryan ja Romaneev (2015)
Evaluation of Security Solutions for Android Systems	Elovici, Mimran ja Shabtai (2015)
Securing Android: A Survey, Taxonomy and Challenges	Darell, Sufatrio, Tong-Wei ja Vrizlynn (2015)
Android security: a survey of issues, malware penetration, and defenses	Faruki, Bharmal, Laxmi, Ganmoor, Gaur, Conti ja Rajarajan (2015)

3 Androidin rakenne

Jotta Androidiin kohdistuvia uhkia voi ymmärtää, täytyy ensin olla peruskäsitys Androidin rakenteesta ja toiminnasta. Desmukh ym. (2014, s. 863) kuvaavat artikkelissaan Androidin rakennetta, joka voidaan jakaa viiteen kerrokseen: Linuxin kerneliin (ytimeen), järjestelmäkirjastoihin, Android Runtime-ympäristöön, sovelluskehykseen (engl. Application Framework) ja sen yläpuolella olevaan sovelluskerrokseen (engl. Application Layer). Käyn seuraavissa kappaleissa Androidin kerrosteisen arkkitehtuurin läpi kuvaten jokaisen kerroksen perustehtävät.



Kuvio 1. Androidin kerrosteinen arkkitehtuuri. (Desmukh ym. (2014).

3.1 Sovelluskerros ja sovelluskehys

Sovelluskerros on Android-arkkitehtuurin ylin kerros. Käyttäjä kommunikoi tämän kerroksen kanssa esimerkiksi soittaessaan ja käyttäessään selainta sekä muita Android-sovelluksia. Android-sovellus koostuu .apk-päätteisestä paketista, joka sisältää sovelluksen koodin ja sovelluksen tarvitsemat resurssit. Sovelluskehys huolehtii puhelimen perustoiminnoista, kuten resurssinhallinnasta ja äänipuheluista. Ylemmän kerroksen sovellukset ovat vuorovaikutuksessa sovelluskehyksen kanssa. (Desmukh ym. 2014, s. 864.)

Vuorovaikutus tapahtuu neljän komponentin avulla: Activities-komponentti määrittelee sovelluksen käyttöliittymänäkymän. Activity-komponentti voi myös käynnistää toisen Activity-komponentin, mutta vain yksi komponentti voi vastaanottaa ja prosessoida dataa kerrallaan. Service-komponentti puolestaan huolehtii taustalla tapahtuvista palveluista ja prosesseista. Mikäli sovelluksen täytyy esimerkiksi ladata tiedostoja tai toistaa musiikkia, Activity-komponentti kutsuu Service-komponenttia, joka suorittaa pyydetyt toiminnot. Content provider-komponentti pitää yllä ja välittää dataa relaatiotietokannnan avulla. Neljäs ja viimeinen komponentti Brodacast receiver puolestaan toimii postilaatikkona sovelluksen lähettämille viesteille. (Padaryan ja Romaneev) 2015, s. 389.). Broadcast receiver-komponentin avulla voidaan reagoida järjestelmän tapahtumiin, kuten näytön sammuttamiseen, langattomaan verkkoon kytkeytymiseen tai puhelun vastaanottamiseen. (Castillo 2011, s. 10). Komponentit viestivät keskenään ja pyytävät toimintoja toisiltaan Intent-viestien avulla. (Padaryan ja Romaneev) 2015, s. 389).

3.2 Kirjastokerros, Android Runtime ja Linux Kernel

Kernelin päällä toimii kirjastokerros, jossa sijaitsevat Androidin natiivit C ja C++-kirjastot. Kirjastot ohjaavat eri laitteiston osien toimintaa, esimerkiksi mediakirjasto hallinnoi video-, kuva- ja äänitoistoa. Kirjastot ohjaavat laitteen toimintaa, esimerkiksi mediakehys (engl. Media Framework) ohjaa videoiden, audion ja kuvien toistoa ja tallennusta. (Desmukh ym. 2014, s. 863) Eräs tärkeimmistä kirjastoista on

SQLite, jonka avulla useimmat sovellukset muodostavat tietokantansa. Tähän tietokantaan sovellukset tallentavat usein pitkäkestoista, henkilökohtaista dataa. (Castillo 2011, s. 10.)

Sovelluskehittäjät käyttävät Java-ohjelmointikieltä kehittäessään sovelluksia Androidille. Android Runtime sisältää Java-kirjastoja ja Dalvik-virtuaalikoneen. Dalvik-virtuaalikone käyttäytyy kuin se olisi erillinen laite, jolla on oma käyttöjärjestelmänsä. Virtuaalikoneesta voi olla samanaikaisesti ajossa useampia ilmentymiä, mikä mahdollistaa muistinhallinnan, tietoturvan ja tuen useammalle säikeelle. Jokainen sovellus ajetaan omana, itsenäisenä prosessinaan virtuaalikoneessa, joten jos sovellus kaatuu, se ei vaikuta muihin laitteessa käynnissä oleviin sovelluksiin. Edellä mainitut ominaisuudet muodostavat ns. "hiekkalaatikon"(engl. sandbox). (Desmukh ym. 2014, s. 864.)

Android-käyttöjärjestelmä on rakennettu Linuxin kernelin päälle, joskin Google on tehnyt siihen joitakin muutoksia. Kernel tarjoaa ylemmille kerroksille järjestelmätoimintoja, kuten muistin-, laite- ja prosessihallinnan (Desmukh ym. 2014, s. 863). Kernel muodostaa siis sillan laitteiston ja sovelluskomponenttien välille. (Castillo 2011, s. 10).

4 Haavoittuvuudet ja uhat

Androidiin kohdistuvia uhkia ja haavoittuvuuksia voidaan kategorisoida monilla eri tavoilla. Desmukh ym. (2014) kategorisoivat Androidiin kohdistuvia haavoittuvuuksia sen mukaan, mistä Androidin kerroksesta ne ovat lähtöisin. Padaryan ja Romaneev) (2015) puolestaan käyttävät kategorisointia, joka jakaantuu kahdeksaan eri osaan. Padaryanin ja Romaneevin käyttämä kategorisointi on hyvin tarkasti rajattu, jakaen uhat sen mukaan, mistä ne ovat lähtöisin. Elovici, Mimran ja Shabtai) (2015) käyttävät jaottelua, joka on hyvin laaja-alainen. Heidän kategorisointinsa jakaantuu käytännössä kahteen osaan sen mukaan, millä metodilla hyökkääjä pääsee käsiksi Android-laitteeseen sekä sen mukaan, mitä hyökkäysmetodia käytetään. Myös Mall ja Gupta) (2014) jakavat uhat kahteen osaan sen mukaan mihin Androidin tietoturvan kerrokseen ne kohdistuvat: Android-tason tietoturvakerrokseen (engl. Android Level Security) tai sovellustason tietoturvakerrokseen (engl. Application Level Security).

Olen tässä tutkielmassa päätynyt kategorisoimaan uhkia kuten Desmukh ym. (2014), eli sen mukaan, mihin Androdin arkkitehtuurin kerrokseen uhat kohdistuvat. Myös tietoliikenne muodostaa olennaisen osan tietoturvasta. Tietoliikenteen haavoittuvuudet voivat kohdistua useisiin kerroksiin, minkä takia olen jakanut sen omaksi alaluvukseen.

4.1 Sovelluskerros

Sovelluskerroksella sijaitsee jokaisen sovelluksen tarvitsema tiedosto AndroidManifest.xml. Tiedosto sisältää kyseisen sovelluksen vaatimat oikeudet, joita se suoritukseensa tarvitsee. Asennuksen yhteydessä käyttäjää pyydetään antamaan sovellukselle sen vaatimat oikeudet. (Armando ym. 2013, s. 180.). Sovelluskerrokseen kohdistuvat hyökkäykset toteutetaan pääosin selainten kautta, joka antaa hyökkääjille mahdollisuuden ajaa haitallista koodia ja päästä käsiksi suojattuihin resursseihin. Esimerkkinä Googlen kuvankäsittely- ja arkistointiohjelma Picasassa ollut aukko,

jossa käyttäjänimi ja salasana lähetettiin suojaamattomana tekstinä autentikoinnin yhteydessä sisäänkirjautumisen jälkeen. Kuka tahansa pystyi pääsemään käsiksi käyttäjän galleriaan ja kuviin. Myös ohjelmointivirhe Androidin selaimessa mahdollisti "Man-in-the-Middle" hyökkäyksen ja käyttäjän toiminnan monitoroinnin. (Desmukh, Patil, Shewale & Singh 2014, s. 864.) Selaimiin kohdistuvissa hyökkäyksissä pyritään hyödyntämään niissä olevia haavoittuvuuksia tai niiden kautta avattavien ohjelmien haavoittuvuuksia, esimerkkinä Flash Playerin, PDF-lukijan tai kuvien esikatseluohjelman haavoittuvuudet. Joskus myöskin pelkästään haitallisella sivustolla käynti saattaa laukaista selaimen kautta toteutettavan hyökkäyksen, jossa asennetaan käyttäjän puhelimelle haittaohjelma tai suoritetaan joitakin haitallisia toimintoja. (Elovici, Mimran & Shabtai 2015, s. 3.)

4.2 Sovelluskehys

Desmukh ym. (2014, s. 864) mukaan sovelluskehykseen kohdistuvat uhat mahdollistavat haitallisen koodin ajamisen, sekä mahdollisuuden päästä käsiksi suojattuun informaatioon. Esimerkiksi Bluetoothissa esiintynyt haavoittuvuus mahdollisti pääsyn käsiksi laitteen käyttäjän arkaluontoiseen tietoon. Sovelluskehykseen kohdistuvilla hyökkäyksillä on myös onnistuttu ohittamaan käyttäjän sovelluksille asettamia luparajoitteita ja päästy käsiksi kameraan ja mikrofoniin. (Desmukh ym. 2014, s. 864.). Hyökkääjät ovat esimerkiksi käyttäneet hyväksi Flash Playeristä löytyvää haavoittuvuutta, jolla piilottamalla skripti kuvaan voidaan ottaa haltuun laitteen kamera tai mikrofoni.(Elovici, Mimran & Shabtai 2015, s. 3)

Android-sovellus sisältää neljä erilaista komponenttia, joille muut sovellukset voivat esittää pyyntöjä: Activites, Services, Broadcast Receivers ja Content Providers. Sovellukset esittävät pyyntöjä toisilleen Intent-viestien avulla. (Kraxberger ym. 2011, s. 42-43) Tähän sovellusten sisäisten komponenttien välillä tapahtuvaan viestintään (engl. Inter-component communication, ICC) sisältyy haavoittuvuuksia, joita kuvailevat mm. Padaryan & Romaneev (2015, s. 389). Hyökkäyksen toteuttava ohjelma voi saada käyttöönsä toisen ohjelman saamia oikeuksia (engl. permissions) ja päästä käsiksi arkaluontoiseen tietoon komponenttien välisen viestinnän avulla. (Pada-

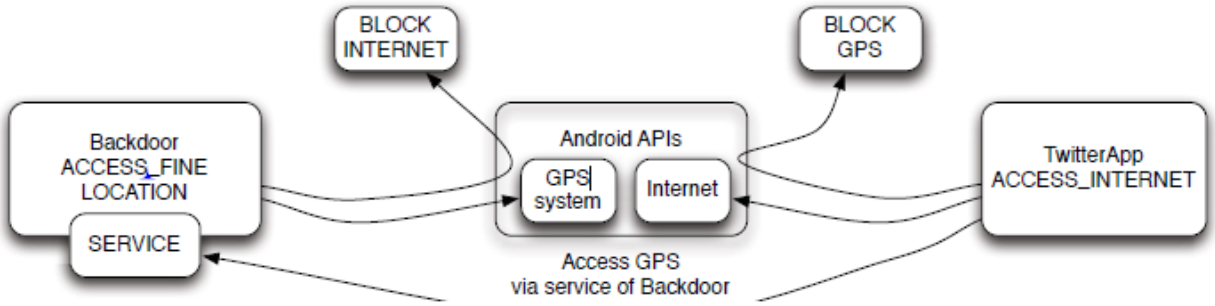
ryan ja Romaneev) 2015, s. 389.).

Oikeuksiin liittyviin haavoittuvuuksiin perehtyvät myös Kraxberger ym. (2011), jotka esittävät tutkimuksessaan, että sellaisiakin oikeuksia, joita käyttäjä ei hyväksynyt, on mahdollista siirtää sovellukselle komponenttien välisen viestinnän avulla. (Kraxberger ym. 2011, s. 41) Heidän mukaansa oikeuksia levittämällä (engl. Permission spreading) sellaisetkin sovellukset, joilla ei ole lainkaan oikeuksia, voivat kutsua toisten sovellusten funktioita, joilla on hyökkääjän kannalta halutunlaisia oikeuksia. Etenkin Services-komponentti on heidän mukaansa haavoittuva, sillä sen kutsuminen on helpoin tapa saavuttaa kahden sovelluksen välinen tiedonsiirto. (Kraxberger ym. 2011, s. 48-49.)

Esimerkiksi karttasovellukseksi naamioitunut haittaohjelma, joka pyytää oikeuksia päästä käsiksi käyttäjän sijaintitietoon, saattaa käyttäjän tietämättä toteuttaa Service-komponentin, joka tarjoaa sijaintitiedon muille sovelluksille. Toinen saastunut ohjelma, jolla puolestaan on oikeudet Internetin käyttöön, pystyy kutsumaan karttasovelluksen Service-komponenttia ja saamaan haltuunsa käyttäjän sijaintidatan. Sen jälkeen tämä sovellus voi levittää käyttäjän sijaintidataa Internetin välityksellä saamiensa oikeuksien avulla. (Kraxberger ym. 2011, s. 47.) Kraxberger ym. (2011) siis osoittivat tutkimuksessaan, että haittaohjelmat pystyvät sovellusten välisellä kommunikaatiolla pääsemään käsiksi myös oikeuksiin, joita niille ei ole annettu. Alla oleva kuva havainnollistaa edellä kuvattua oikeuksien leviämisestä syntyvää uhkaa.

4.3 Kirjastokerros, Android Runtime ja Linux Kernel

Myös Androidin kirjastoihin liittyy haavoittuvuuksia. Tähän kerrokseen kuuluviin haavoittuvuuksiin lukeutuvat vuoden 2011 ZergRush -haavoittuvuus, joka mahdollisti palvelunestohyökkäyksen toteuttamisen. ZergRush myös aiheutti muistiylivuotoa syöttämällä väärän määrän parametreja eräälle ohjelmointirajapinnalle (engl. Application programming interface, API). GingerBreak -nimistä haavoittuvuutta puolestaan käyttivät monet haittaohjelmat, joiden tavoitteena oli saada korkeimmat,



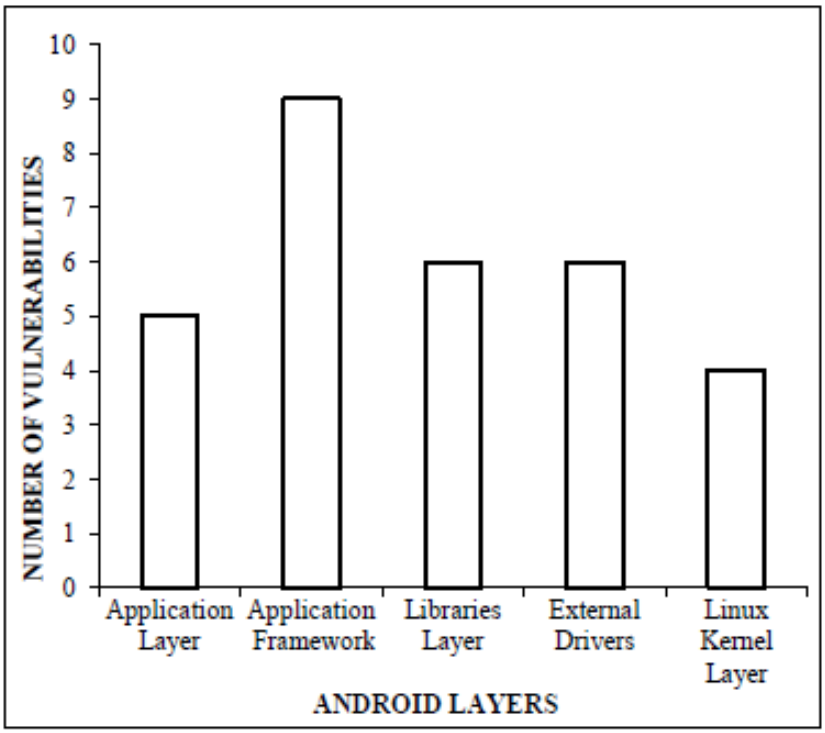
Kuvio 2. Esimerkki oikeuksien leviämisestä (engl. Permission Spreading) muodostuvasta uhasta. (Kraxberger ym. 2011).

juuritason oikeudet (engl. Root privileges). Haavoittuvuuksia on löytynyt myös yleisesti mobiililaitteissa käytettävien Qualcommin ja PowerVR:n laiteajureista. (Desmukh ym. 2014, s. 864-865.).

Androidin Runtime-ympäristön uhat kohdistuvat lähinnä Dalvik-virtuaalikoneeseen ja sen haavoittuvuuksiin. Esimerkiksi Zygote-funktio, joka luo uuden ilmentymän virtuaalikoneesta sovellusta avattaessa, on osoittautunut haavoittuvaksi. Zygoteen liittyvän haavoittuvuuden avulla mobiililaitteen resurssit oli mahdollista kuluttaa kokonaan luomalla sen avulla rajaton määrä virtuaalikoneen ilmentymiä. Toisena esimerkkinä on ZimperLich-haavoittuvuus, jonka avulla on mahdollista saada haitalliselle sovellukselle juuritason oikeudet. Plankton-niminen haittaohjelma puolestaan pesiytyy saastutetun ohjelman taustapalveluksi, joka suoritetaan kun itse ohjelma suoritetaan. Se kerää laitteesta tietoina laitteen id:n sekä saastutetun ohjelman saamat oikeudet ja lähettää ne verkon yli palvelimelle. Palvelin prosessoi saadut tiedot ja lähettää laitteeseen verkko-osoitteen, josta ladataan lisää suoritettavaa koodia. (Elovici, Mimran & Shabtai 2015, s. 3.)(F-Secure 2016)

Androidin ytimeen eli kerneliin kohdistuvat hyökkäykset pyrkivät saamaan juuritason oikeudet ja injektoimaan haitallista koodia ja sammuttamaan tietoturvasovelluksia. Esimerkiksi Galaxy S3:sen kernelistä löydettiin haavoittuvuus, jonka avulla kuka tahansa pystyi kirjoittamaan ja lukemaan laitteen sisältöä. (Elovici, Mimran & Shabtai 2015, s. 4.). Jotkin Android-käyttöjärjestelmää käyttävien laitteiden valmis-

tajat ovat myös lisänneet ytimeen omia moduulejaan paremman laitetuen saavuttamiseksi ja näissä moduuleissa voi olla haavoittuvuuksia.(Padaryan & Romaneev 2015, s. 389).



Kuvio 3. Androidin tunnetut haavoittuvuudet kerroksittain lajiteltuna (Desmukh ym. 2014).

4.4 Tietoliikenne

Android-sovellukset käyttävät tietoliikenteensä suojaamiseen SSL- (Secure Socket Layer) ja TLS (Transport Layer Security)-protokollia. Fahl, Harbach, Muders, Baumgärtner, Freisleben & Smith (2012) perehtyvät tutkimuksessaan Android-sovellusten tietoliikenteen suojaukseen sekä näiden protokollien käyttöön sovelluksissa. Tutkijoiden tavoitteena oli myös saada selville, kuinka hyvin sovellukset ilmoittavat SSL-protokollan käytöstä laitteen käyttäjälle, sillä Android itsessään ei anna SSL:n käytöstä minkäänlaista tietoa. He analysoivat sovelluksia kehittämänsä MalloDroid-työkalun avulla, joka tutki potentiaalisia aukkoja sovellusten tietoliikenteen suojauksessa ja sitä, onko niissä haavoittuvuuksia niin sanotuille Man-in-the-Middle

(MITM) hyökkäyksille. MITM-hyökkäyksissä hyökkääjä pääsee kuuntelemaan ja mahdollisesti kaappaamaan käyttäjän verkon yli siirtämää tietoliikenettä. Aktiivisessa MITM-hyökkäyksessä hyökkääjä pääsee käsiksi ja manipuloimaan käyttäjän verkon yli lähettämää dataliikennettä. Passivisissa MITM-hyökkäyksissä hyökkääjä pystyy vain kuuntelemaan käyttäjän lähettämää liikennettä. (Fahl ym. 2012, s. 50-52.)

Tutkimuksessa oli mukana 13 500 Googlen Play Marketissa saatavilla olevaa sovellusta, ja MalloDroid tutki valittujen Androi-sovellusten koodia kartoittaen, kuinka hyvin SSL:n ja TLS:n käytön periaatteita oli noudatettu. Tutkimuksissa kävi ilmi, että sovelluksista 1074 sisälsi haavoittuvuuksia MITM-hyökkäyksille. Haavoittuvuudet johtuivat SSL- ja TLS-protokollan erilaisista väärinkäytöistä ja tutkijat onnistuivat manuaalisessa tarkastelussa toteuttamaan MITM-hyökkäyksiä 41 sovellukselle sadasta valitusta sovelluksesta. Näistä 41 sovelluksesta saatiin kaapattua muun muassa American Express-, PayPal, Facebook- ja Twitter-tilien kirjautumistietoja. Tutkijat pystyivät myös huijaamaan laitteen antivirus-ohjelmistoa väärillä hälytyksillä tai sammuttamaan sen kokonaan. Kaiken kaikkiaan MITM-hyökkäyksille haavoittuvien sovellusten kumulatiivinen käyttäjäkunta oli 39,5 ja 185 miljoonan käyttäjän välillä. (Fahl ym. 2012, s. 50-51.)

5 Haavoittuvuuksien torjuminen

Siinä missä Applella on tiukat säännökset siitä, millaisia sovelluksia heidän kauppapaikassaan voi julkaista niin turvallisuuden kuin sisällönkin osalta, Google painottaa avoimempaa lähestymistapaa tarjoten Android-kehittäjilleen vapaampaa politiikkaa turvallisuuden ja sisällön suhteen. Googlen tapana on pikemminkin poistaa haitallisiksi havaitut ohjelmat jälkeenpäin sovellusten kauppapaikastaan. Androidin tietoturvapolitiikan selkärangan muodostaa oikeus-pohjainen järjestelmä (engl. Permission based system), jossa kehittäjä määrittelee sovelluksensa tarvitsemat resurssit ja käyttöoikeudet. Käyttäjän päätettävissä on sovelluksen asennuksen yhteydessä se, hyväksyykö hän sovelluksen vaatimat oikeudet vai ei. (Kraxberger ym. 2011, s. 41) Mikäli oikeuksia ei hyväksytä, sovellusta ei voi asentaa.

Näihin ja moniin muihinkin Androidin tietoturvapolitiikkaa koskeviin asioihin on kuitenkin esitetty useita parannusehdotuksia. Kuten uhkiakin, myös haavoittuvuuksien torjuntakeinoja voidaan kategorisoida monilla eri tavoilla.

Olen tässä tutkielmassa jaotellut Androiidin kohdistuvien haavoittuvuuksien torjuntakeinot kolmeen osa-alueeseen: haittaohjelmien tunnistus- ja torjuntaohjelmistoihin, Androidiin liittyviin päivityksiin ja Androidin tietoturvaa koskeviin kehitysehdotuksiin ja apuohjelmistoihin, sekä tietoliikenteen tietoturvan parannusehdotuksiin. Mielestäni näitä kolmea osa-aluetta hyödyntämällä voidaan torjua huomattavan suuri osa Androidiin kohdistuvista haavoittuvuuksista.

5.1 Haittaohjelmien tunnistus- ja torjuntaohjelmistot

Androidin haittaohjelmia voidaan torjua myös lukuisilla erilaisilla sovelluksilla. Nämä sovellukset analysoivat puhelimesta löytyviä sovelluksia ja muodostavat niistä riskianalyysin tutkimalla sovelluksen koodia tai toimintaa. Torjuntaohjelmat ovat tehokkaita tunnettujen haavoittuvuuksien torjunnassa, sillä tunnettujen haavoittuvuuksien tietokantoja päivitetään jatkuvasti. Android-ympäristössä on tarjolla sekä isojen tietoturvajättien, kuten McAfeen, F-Securen ja Nortonin torjunta- ja tunnis-

tusohjelmia, mutta myös lukuisia avoimeen lähdekoodiin perustuvia sovelluksia. Avoimeen lähdekoodiin perustuvien ratkaisujen puolesta puhuvat mm. Khan ym. (2012). Heidän mukaansa myös torjuntaohjelmistojen pitäisi noudattaa samaa avoimuutta kuin Androidin mobiilikäyttöjärjestelmänä. Khan ym. (2012) mukaan jokainen Androidin tietoturvamalliin kohdistuva parannus tulisi julkaista avoimen lisenssoinnin alaisena. Toisenlaiset ratkaisut saattavat heidän mukaansa olla toimivia, mutta toimivat Androidin avoimuutta vastaan, mikä saattaa hidastaa niiden käyttöönottoa. (Khan ym. 2012, s. 81.)

Torjuntaohjelmistojen tapa tunnistaa ja torjua haittaohjelmia jakaantuu yleensä kolmeen lähestymistapaan: staattiseen, dynaamiseen ja hybridiin, jossa yhdistetään molempien hyviä puolia. Staattisessa lähestymistavassa sovelluksen ohjelmakoodia tutkitaan suorittamatta itse sovellusta, joka on nopea tapa. Staattista lähestymistapaa käyttävät torjuntaohjelmistot kärsivät kuitenkin vääristä hälytyksistä. Dynaamisessa analyysissä ohjelma suoritetaan ja tutkitaan sen käyttäytymistä ja kanssakäymistä laitteen kanssa ajon aikana. Dynaamisen lähestymistavan etuna on, että sillä pystytään tutkimaan monimutkaisiakin haittaohjelmia. (Faruki ym. 2015, s. 1008.)

Haittaohjelmien tunnistus- ja torjuntaohjelmistot ovat käyttäjälle helpoin ja nopein tapa parantaa Android-laitteensa tietoturvaa. Niinpä niiden kehittäminen ja tutkiminen ovat tärkeä osa Androidin haavoittuvuuksien torjuntaa. Kaupallisten yritysten kehittämien torjuntaohjelmistojen etuna on niiden suuri tunnettujen haavoittuvuuksien tietokanta, hyvä saatavuus ja tekninen tuki. Tiedepiireissä on myös kehitetty ja esitelty lukuisia haittaohjelmien tunnistus- ja torjuntaohjelmistoja, jotka perustuvat yleensä avoimeen lähdekoodiin. Niitä ei kuitenkaan välttämättä julkaista suuren yleisön saataville, vaan osa jää vain tieteellisessä tutkimuksessa käytetyiksi resursseiksi. Esittelen seuraavaksi joitakin mielestäni keskeisiä ja mielenkiintoisia haittaohjelmien torjuntaohjelmistoja.

Fahl ym. (2012) esittävät käyttämänsä MalloDroid-työkalun integroimista sovellusten asennuksen yhteyteen. MalloDroid-työkalu voisi analysoida sovelluksen koodin asennuksen yhteydessä ja varoittaa mahdollisesti haitallisesta sovelluksesta. MalloDroid-työkalun voisi myös integroida suoraan sovelluskauppaan, jolloin se analysoisi so-

velluksen jo kauppaan vietäessä, jolloin potentiaalisesti haitallisen ohjelman pääsy kauppaan voitaisiin estää. MalloDroid-työkalun voisi myös julkaista omana sovelluksenaan verkossa, jolloin käyttäjillä olisi halutessaan mahdollisuus tarkastaa sovelluksen haitallisuus itsenäisesti. (Fahl ym. 2012, s. 59)

Luvussa 4.2 esiteltyyn oikeuksien levittämiseen (engl. Permission Spreading) Mall ja Gupta) (2014, Ongtang ym. (2012) mukaan) esittävät ratkaisuksi SAINT-sovellusta (Secure Application INTeraction). Sovelluksen avulla voidaan luoda monimutkaisiakin sääntöjä sovellusten väliselle kommunikoinnille ja näin tarjota parempi valvonta siihen, mihin oikeuksiin sovellus pääsee käsiksi. Heidän mukaansa sovelluksen tulisi kysyä oikeuksien hyväksymistä myös silloin, kun sovellus ilmoittaa haluavansa käyttää jonkin toisen sovelluksen ominaisuutta. Käyttäjälle tulisi tällöin myös ilmoittaa lisääntyneestä tietoturvariskistä helposti ymmärrettävällä asteikolla esimerkiksi yhdestä viiteen.(Mall & Gupta 2014, s. 4.)

Andromaly on Asaf Shabtain, Uri Kanonovin, Yuval Elovicin, Chanan Glezerin ja Yael Weissin vuonna 2011 tutkimuksessaan esittelemä ohjelmisto, joka tutkii Androidin haavoittuvuuksia. Andromalyn analyysi perustuu koneoppimiseen ja se monitoroi reaaliaikaisesti laitteen prosessorin käyttöastetta, internetin yli lähetetyn datan määrää, käynnissä olevien prosessien määrää ja akun käyttöä. Sen käyttämä lähestymistapa haavoittuuvuksien analysoinnissa on dynaaminen. (Faruki ym. 2015, s. 1011.)

Bouncer on dynaamista lähestymistapaa hyödyntävä ohjelmisto, jolla Google suojaa omaa sovelluskauppaansa, Google Playta, haittaohjelmilta. Bouncer on virtuaalikone, joka testaa sovelluskauppaan lähetettyjä ohjelmia suorittamalla ne ja tutkimalla, onko niiden käyttäytyminen epäilyttävää. Bouncer myös vertailee suorittamaansa sovelluksen käyttäytymistä aikaisemmin testaamiinsa sovelluksiin. (Faruki ym. 2015, s. 1013.)

Kirin on staattista lähestymistapaa hyödyntävä ohjelmisto, jonka tutkijat William Enck, Machigar Ongtang ja Patrick McDaniel esittelivät jo vuonna 2009. Kirinin tavoitteena on antaa käyttäjälle mahdollisuus Androidin oikeuspohjaisen järjestelmän

parantamiseen. Sen avulla voi määrittää säännöt sille, mitä haavoittuvuuksille alttiita oikeusyhdistelmiä sovellukset saavat käyttää. Mikäli sovelluksen oikeudet rikkovat käyttäjän määrittelemää säännöstöä, sovellusta ei asenneta. (Faruki ym. 2015, s. 1015.) Tällainen sovellus on edelleen ajankohtainen, sillä oikeuspohjainen järjestelmä pysyy yhtenä Androidin tietoturvan suurimmista heikkouksista.

5.2 Androidin päivitykset, kehitysehdotukset ja muut apuohjelmistot

Google julkaisee Androidiin päivityksiä tasaisin väliajoin. Google nimeää päivityksensä aina jonkin makeisen mukaan. Neljä tuoreinta päivitystä ovat koodinimeltään JellyBean, KitKat, Lollipop ja Marshmallow. JellyBean julkaistiin kesällä 2012, uusin versio Marshmallow syksyllä 2015. (Android Developer Portal 2016) Päivitykset ovat tuoneet uudistuksia myös tietoturvaan. Uusin niistä, Marshmallow, tuo muutoksia myös oikeuspohjaiseen järjestelmään. Sen sijaan, että käyttäjältä kysytään oikeuksien hyväksymistä sovelluksen asennuksen yhteydessä, ne kysytäänkin tulevaisuudessa sovelluksen ajon aikana. Marshmallow tuo mahdollisuuden myös oikeuksien poistoon milloin tahansa asetusten kautta. (Android Developer Portal 2016) Marshmallow siis tuo kaivattuja parannuksia oikeuspohjaiseen järjestelmään.

Androidiin päivityksiin liittyy olennaisena osana kuitenkin se, että ne ovat saatavilla välittömästi julkaisun jälkeen vain Googlen omille Android-laitteille. Muiden kuin Googlen nimen alla julkaistut mobiililaitteet joutuvat odottamaan päivityksiä alueesta ja valmistajasta riippuen useita kuukausia. Laajan laitekannan ja suuren valmistajakirjon takia päivitykset saavuttavatkin käyttäjät hitaammin kuin tietoturvan kannalta olisi toivottavaa. Hitaampi päivitysten tavoittavuus antaa hyökkääjille lisäaikaa uusien haavoittuvuuksien etsimiseen ja hyödyntämiseen. Myös Mall ja Gupta) (2014) esittävät tietoturvan parantamiseksi Androidin päivitysten (engl. patch) julkaisusyklin nopeuttamista. (Mall & Gupta 2014, s. 4, Vidas ym. (2011) mukaan)

Khan ym. (2012) esittävät, että Androidin oikeuspohjaista järjestelmää tulisi kehit-

tää. Heidän mukaansa käyttäjällä tulisi olla mahdollisuus jakaa sovellusten oikeuksia hienojakoisemmin. Käyttäjällä tulisi heidän mukaansa olla mahdollisuus estää sovellusta pääsemästä käsiksi arkaluontoiseen tietoon tietyssä kontekstissa, ilman että hänen täytyy poistaa ohjelma kokonaan. Käyttäjällä tulisi myös olla mahdollisuus valvoa sovellukselle annettujen oikeuksien käyttöä sovelluksen ajon aikana. Heidän mukaansa tulee tietoturvaratkaisuja kehiteltäessä myös kiinnittää erityistä huomiota siihen, että olemassaolevaan koodipohjaan ei tehtäisi kerralla hyvin radikaaleja ratkaisuja. Heidän mukaansa kynnys radikaalisti vanhaa koodipohjaa muuttavien tietoturvaratkaisujen käyttöönotto on niin korkea, että ne saattavat jäädä kokonaan toteuttamatta. Khan ym. (2012, s. 80-81) Myös Darell ym. (2015) painottavat tutkimuksessaan Androidin oikeuspohjaisen järjestelmän kehittämistä. Heidän mukaansa käyttäjällä tulisi olla mahdollisuus aiemmin myönnettyjen oikeuksien poistamiseen. Silloin tulisi myös huolehtia siitä, että tällainen mahdollisuus aiheuttaisi mahdollisimman vähän yhteensopivuusongelmia. (Darell ym. 2015, s. 39.)

Palomuurit ja roskapostisuotimet pystyvät pysäyttämään haitallisen liikenteen jo ennen kuin se saavuttaa käyttäjän. Androidille on jo saatavilla roskapostisuodattimia, kuten SpamDrain ja SMS Spam blocker. Roskapostia tulee nykyään joka lähteestä, niin tekstiviesteinä, pikaviestinsovellusten kuten WhatsAppin kautta, puhelinsoittoina sekä sähköposteina. Sähköpostin osalta roskapostia voidaan suodattaa joko palvelinpuolella esimerkiksi Gmailin toimesta, tai laitteen päässä sovelluksen toimesta. (Elovici, Mimran & Shabtai 2015, s. 6.)

Arkaluontoisen tiedon päätymistä vääriin käsiin voidaan estää salaamalla arkaluontoinen sisältö, sekä rajoittaa siihen pääsyä esimerkiksi salasanoin, vaatimalla käyttäjää piirtämään jokin tietty kuvio tai tunnistautumalla omalla sormenjäljellä. Myös älypuhelimen varkauden estäviä mekanismeja voidaan toteuttaa siten, että mahdollistetaan laitteen etähallinta mikäli se joutuu vääriin käsiin. Näin voidaan esimerkiksi saada selville laitteen sijainti, lukita laitteeseen sisäänpääsy ja tarpeen vaatiessa pyyhkiä sen sisältämä tieto kokonaan. Näiden mekanismien tulo mobiililaitteisiin on vielä kesken, vaikka älypuhelimet sisältävät runsaasti arkaluontoista dataa, kuten sijaintitietoja, dokumentteja, kontakteja, kalenteritietoja, pankkitunnuksia ja

niin edelleen. (Elovici, Mimran & Shabtai 2015, s. 6.) Myös Mall ja Gupta) (2014) esittävät, että laitteen tietojen tyhjentämiseen etänä tulisi olla mahdollisuus esimerkiksi sen jälkeen, kun salasana on annettu väärin kymmenen kertaa peräkkäin. (Mall & Gupta 2014, s. 4)

Sormenjälkitunnistimet ovat ominaisuus, joka on viime vuosina saanut tuulta alleen erityisesti korkeamman hintaluokan älypuhelimissa. Sormenjälkitunnistautumisella voidaan myös nopeuttaa jokapäiväistä käyttäjän tunnistautumista esimerkiksi pankkiasioita hoitaessa monimutkaisten salasanojen muistamisen ja syöttämisen sijaan.

5.3 Tietoliikenne

Tietoliikenteen suojauksessa on puutteita, kuten luvussa 4.4 osoitettiin. Fahl ym. (2012) esittävät tutkimuksessaan, että Androidin tietoturvaa ja tietoliikenteen suojausta voidaan parantaa monilla eri ratkaisuilla, jotka jakaantuvat kolmeen kategoriaan: Ratkaisut, jotka voidaan integroida osaksi Android-käyttöjärjestelmää, ratkaisut jotka voidaan integroida osaksi sovelluskauppoja ja ratkaisut, jotka toimivat itsenäisesti erillään Androidista ja laitevalmistajista, esimerkiksi kolmansien osapuolien tarjoamina sovelluksina. (Fahl ym. 2012, s. 58.)

Androidin käyttöjärjestelmään integroituvia ratkaisuja tietoliikenteen haavoittuvuuksien torjumiseksi Fahl ym. (2012) mukaan ovat SSL-protokollan käytön kustomoinnin estäminen ja kehittäjien pakottaminen käyttämään standardoitua SSL-protokollan implementointia sovelluskoodissaan. Fahl ym. (2012) esittävät, että tällä tavoin suurin osa vaarallisesta koodista ja haavoittuvuuksista voitaisiin välttää. Toinen käyttöjärjestelmään integroituva keino heidän mukaansa on niin sanottu HTTPS-Everywhere, jossa kaikki API:en välillä tapahtuva tietoliikenne salattaisiin HTTPS-protokollalla. Sovellusten tulisi myös Fahl ym. (2012) mukaan tarjota käyttäjälle visuaalista informaatiota siitä, kuinka hyvin käynnissä olevan sovelluksen tietoliikenne on suojattu. Tutkijoiden mukaan käyttöjärjestelmästä tulisi löytyä selkeä mittari siitä, mitkä sovellukset kommunikoivat internetin välityksellä ja kuinka hyvin suojattua se on.

(Fahl ym. 2012, s. 58-59.)

Myös Mall ja Gupta) (2014) esittävät tutkimuksessaan ratkaisuja tietoliikenteen haavoittuvuuksien torjumiseksi ja tietoturvan parantamiseksi. Heidän mukaansa asiakaspalvelin (engl. Client-Server) pohjaisen tietoliikenteen haavoittuvuuksiin voidaan puuttua käyttämällä tietoliikenteen salausprotokollia kuten SSL:ää ja TLS:tä. Kuten Fahl ym. (2012) tutkimuksissaan todistivat ja myös Mall ja Gupta) (2014) huomauttavat, protokollien oikeaoppinen käyttö on paljolti kiinni kehittäjän taitotasosta. Heidän mukaansa SSL:än käyttö ei kuitenkaan ole aina perusteltua, kun halutaan maksimoida laitteen suorituskyky, esimerkiksi pelisovelluksista puhuttaessa. He eivät myöskään näe tarvetta käyttää HTTPS-protokollaa koko ajan, vaan esittävät että sovellusten tietoliikenteen suojaustaso jaettaisiin kolmeen tasoon: Kolmas taso olisi kaikista turvallisin ja käyttäisi HTTPS-protokollaa kaikessa toiminnassaan. Toinen taso käyttäisi HTTPS-protokollaa vain osittain ja ensimmäinen taso ei käyttäisi SSL-protokollaakaan. Kehittäjän tulisi valita kategoria sovellukselleen ja käyttäjälle annettaisiin ilmoitus sovelluksen turvallisuustasosta asennuksen yhteydessä, jonka jälkeen hän voisi hyväksyä tai hylätä sovelluksen asennuksen. Kuitenkaan esimerkiksi pankkisovelluksissa protokollien käytössä ei tulisi Mallin ja Guptan (2014) mukaan tehdä kompromisseja. (Mall & Gupta 2014, s. 4.)

Fahl ym. (2012) esittävät Android-käyttöjärjestelmään integroituvan keinon oikeuspohjaisen järjestelmän parantamiseksi, jossa oikeuksia yksityiskohtaistettaisiin esimerkiksi koskemaan pelkän Internetiin pääsyoikeuden sijaan niin, että SSL-pohjaiselle, eli salatulle, internetinkäytölle olisi oma oikeutensa ja salaamattomalle internetinkäytölle omansa. Näin käyttäjällä olisi mahdollisuus tämän informaation pohjalta vältellä sovelluksia, jotka eivät käytä SSL- tai HTTPS-salauksia ollenkaan tietoliikenteessään. Tämä tapa ei kuitenkaan suojaisi puutteelliselta SSL-protokollan käytöltä, jonka Fahl ym. (2012) tutkimuksessaan havaitsivat. (Fahl ym. 2012, s. 58-59.).

6 Yhteenveto

Android on kasvanut suosituimmaksi mobiilikäyttöjärjestelmäksi 2010-luvulla. Kandidaatintutkielmani tavoite oli tarjota yleiskatsaus Androidiin kohdistuviin tietoturvauhkiin ja haavoittuvuuksiin sekä siihen, kuinka niitä voidaan torjua. Tutkimuksessani selvisi, että suurin osa haavoittuvuuksista kumpuaa tietoliikenteen puutteellisesta suojauksen, Androidin oikeuspohjaisen järjestelmän heikkouksien ja laitteiden välillä leviävien haittaohjelmien kautta. Androidiin kohdistuvat uhat ja haavoittuvuudet kohdistuvat johonkin Androidin kerrokseen. Tietoliikenteen puutteellisesta suojauksesta johtuvat haavoittuvuudet voivat kohdistua mihin tahansa näistä kerroksista. Päädyinkin lajittelemaan Androidiin kohdistuvat uhat tietoliikenteen puutteellisesta suojauksesta kumpuavien uhkien lisäksi sen mukaan, mihin Androidin kerrokseen haavoittuvuudet kohdistuvat.

Kolme osa-aluetta Androidin tietoturvan parantamiseksi ovat haittaohjelmien tunnistus- ja torjuntaohjelmistojen kehittäminen ja käyttäminen, Androidin ja oikeuspohjaisen järjestelmän kehittäminen päivitysten avulla sekä tietoliikenteen oikeaoppinen suojaaminen. Näihin kolmeen osa-alueeseen keskittymällä voidaan torjua suurin osa Androidiin kohdistuvista tietoturvauhista. On tärkeää, että asiaa tutkitaan aktiivisesti, sillä mobiilipuolella teknologia kehittyy ja käyttäjämäärät kasvavat edelleen hyvin nopeasti.

Googlenkin kannalta on edullista, että Androidin tietoturvaan perehdytään entistä tarkemmin sen suosion kasvaessa. Androidin maine on jo viime vuosina kehittynyt suuntaan, jossa sitä pidetään tietoturvan kannalta heikompana käyttöjärjestelmänä kuin sen suurinta kilpailijaa, Applen iOS:ia. Androidin valttikorttina pidetty avoimuus edellyttää valveutuneisuutta myös käyttäjältään. Ongelmaksi nousee kuitenkin se, kuinka paljon yksittäinen käyttäjä voi loppujen lopuksi valinnoillaan vaikuttaa. Oikeuspohjaisessa järjestelmässä oikeuksien hyväksymättä jättäminen tarkoittaa sitä, että sovellusta ei voi asentaa. Näin käyttäjä voi suojella tietoturvaansa, mutta toisaalta hän joutuu luopumaan suosituimmista sovelluksista. Oikeuspohjaista järjestelmää pitäisikin kehittää niin, että käyttäjä on tietoisempi siitä, mihin

tarkalleen ottaen tiettyjä oikeuksia käytetään. Oikeuspohjaista järjestelmää onkin paranneltu Androidin uusimmassa Marshmallow-päivityksessä, mikä osoittaa, että kehittäjät ovat tietoisia sen heikkouksista.

Myös sovelluskehittäjien tulee huolehtia siitä, että heidän sovelluksensa ovat ajan tasalla tietoturvan osalta. Etenkin tietoliikenteen suojauksessa on tärkeää, että sen suojaamiseen kehitettyjä protokollia hyödynnetään oikeaoppisesti. Google ei tutki sovelluskauppaansa ladattuja sovelluksia yhtä tarkasti kuin Apple, mikä edellyttää hyvää luottamusta kehittäjien ja käyttäjien välillä.

Tulevaisuudessa mielenkiintoinen tutkimuskohde olisi esimerkiksi pienempien startup-yritysten kehittämien sovellusten tietoturva. Mobiilipuolella hyvin pienikin määrä kehittäjiä pystyy julkaisemaan sovelluksen, joka kerää suuren määrän käyttäjiä ja nousee menestystarinaksi. Sovellukset ovat usein ulkoisesti hyvin laadukkaita, mutta tietoturvaan liittyvä panostus mietityttää. Onko nopeaa suosiota tavoitellessa olemassa riski jättää tietoturvakysymykset pienemmälle huomiolle?

Kirjallisuutta

Baumgärtner, L., Fahl, S., Freisleben, B., Harbach, M., Muders, T. & Smith, M. 2012. *Why eve and mallory love android: an analysis of android SSL (in)security*. Proceedings of the 2012 ACM conference on Computer and communications security, s. 12–45.

Padaryan, V. & Romaneev, M. 2015. *Android platform security issues*. International Journal of Computer Systems, Volume 02, Issue 09, September, 2015.

Mall, T. & Gupta, S. 2014. *Critical Evaluation of Security Framework in Android Applications: Android-level security and Application-level security*.International Research Journal of Computers and Electronics Engineering (IRJCEE) Vol. 2, Iss. 1, DEC 2014

Elovici, Y., Mimran D. & Shabtai, A. 2015. *Evaluation of Security Solutions for Android Systems*.

Darelll, J.J., Sufatrio, Tong-Wei C. & Vrizlynn L.L. 2015. *Securing Android: A Survey, Taxonomy, and Challenges*.ACM Computing Surveys, Vol. 47, No. 4, Article 58, Publication date: May 2015.

Desmukh, V., Patil, S., Shewale H. & Singh P. 2015. *Analysis of Android vulnerabilities and modern exploitation techniques*. ICTACT Journal on communcation technology, Volume 05, Issue 01, Publication date: March 2014.

Armando, A., Merlo, A. & Verderame L. 2013. *An Empirical Evaluation of the Android Security Framework*. Security and Privacy Protection in Information Processing Systems, Springer Berlin Heidelberg, 2013. s. 176-189.

Kraxberger, S., Lackner, G., Leibetseder J., Marsalek A., Gissing M., Orthacker C., Prevenhueber O. & Teufl P. 2011. *Android security permissions–can we trust them?*. Security and Security and Privacy in Mobile Information and Communication Systems, pp. 40-51. Springer Berlin Heidelberg, 2011.

F-Secure. 2016. *Threat Description, Android Plankton*. Saatavilla WWW-muodossa <URL: `https://www.f-secure.com/v-descs/trojan_android_plankton.shtml`, Viitattu 14.3.2016

Castillo, Carlos A. 2011. *Android Malware past, present, and future*. White Paper of McAfee Mobile Security Working Group (2011)

Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. 2015. *Android security: a survey of issues, malware penetration, and defenses*. Communications Surveys & Tutorials, IEEE 17.2 (2015): 998-1022.

Khan, S., Nauman, M., Othman, A.T., & Shahrulniza, M. 2012. *How secure is your smartphone: An analysis of smartphone security mechanisms*. Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on. IEEE, 2012.

Android. 2016. *Requesting permissions at runtime, Android Marshmallow*. Saatavilla WWW-muodossa <URL: http://developer.android.com/training/permissions/requesting.html/>. Viitattu 26.4.2016

Android. 2016. *Android version history*. Saatavilla WWW-muodossa <URL: http://developer.android.com/about/versions/>. Viitattu 26.4.2016

Gartner. February 2016. *Gartner report, worldwide smartphone sales 2015*. Saatavilla WWW-muodossa <URL: http://www.gartner.com/newsroom/id/3215217/>. Viitattu 26.4.2016

Salminen, A. 2011. *Mikä kirjallisuuskatsaus? Johdatus kirjallisuuskatsauksen tyyppeihin ja hallintotieteellisiin sovelluksiin*. Vaasan yliopisto, 2011.