Janita Sallanko

Tietoturvauhat ja niiden ehkäiseminen Android-sovelluksissa

Tietotekniikan kandidaatintutkielma

28. huhtikuuta 2017

Jyväskylän yliopisto

Tietotekniikka

Tekijä: Janita Sallanko

Yhteystiedot: janita.h.sallanko@student.jyu.fi

Työn nimi: Tietoturvauhat ja niiden ehkäiseminen Android-sovelluksissa

Title in English: Security threats and their prevention in Android applications

Työ: Kandidaatintutkielma

Sivumäärä: 20+0

Tiivistelmä: Kandidaatintutkielma käsittelee Android-sovellusten tietoturvauhkia ja näiden ehkäisemistä. Tutkielma rajautuu älypuhelimien tarkasteluun kaikkien Android-laitteiden sijasta. Kyberrikollisuuden lisääntymisen myötä on sovellusten turvallisuuden varmistamisen merkitys noussut. Kandidaatintutkielma käsittelee tietoturvaa ensisijaisesti sovelluksen ohjelmoitsijan näkökulmasta.

Avainsanat: android, sovellus, tietoturva

Abstract: This study investigates security threats and their prevention in Android applications. The study concentrates especially on Android mobile devices instead of all Android devices. Security is an essential priority in applications since the cyber criminal acitivity is on the increase. The study focuses primarily on the software developer's point of view.

Keywords: android, application, security

Kuviot

Sisältö

1 Johdanto

Android-mobiilikäyttöjärjestelmän käyttö on muuttunut yhä suositummaksi vuosi vuodelta ja sitä käytetään jo yli 190 maassa. Googlen luomat mobiilikäyttöjärjestelmät ovat pääasiassa suunniteltu älypuhelimille ja tableteille, mutta sitä on integroitu jopa älykelloihin ja autoihin. (Android Developers, 2017.) Android on saanut miljoonia uusia käyttäjiä vime vuosien aikana ja luvut ovat kasvaneet tasaisesti viime vuosina (Statista, 2017).

Androidin yleistyessä sovellusten tarjonta on lähtenyt jyrkästi nousuun (Statista, 2016). Google Play Storen latausten lukumäärä on ylittänyt jo miljoonan joka kuukausi. Viimeisen viiden vuoden aikana on julkaistu Androidin eri versioita vuosittain nopeassa tahdissa. Useat älypuhelimien valmistajat kuten Huawei, LG, Samsung sekä Sony Ericsson käyttävät androidia mobiilikäyttöjärjestelmänään.

Mobilikäyttöjärjestelmämarkkinoilla Googlen suurin kilpailija Apple on luonut iOS-käyttöjärjestelmän jota he soveltavat puhelimien valmistuksessa. Toistaiseksi iOS ei ole onnistunut tavoittamaan yhtä suurta käyttäjämäärää kuin Android (Nielsen, 2015). Androidin suuresta suosiosta huolimatta käyttäjistä jopa 20 prosenttia on huolissaan älypuhelimensa turvallisuudesta, kun taas Applen puhelimia käyttävistä vastaava luku on viisi prosenttia (Obiri-Yeboah & Qi, 2016).

Androidin ja sen sovellusten turvallisuutta on tärkeä tutkia monestakin syystä. Vaikka uudemmissa versioissa on parannettu aikaisempien versioiden vikoja, on olemassa edelleen heikkouksia jotka altistavat käyttäjän tietoturvamurroille. Yhteisiä tietoturvauhkia eri versioiden kesken on olemassa vielä nykypäivänäkin (Benítez-Mejía & Sánchez-Pérez & Toscano-Medina, 2016). Järjestelmien lisäksi tietomurroissa on mahdollista päästä käyttäjän tietoihin käsiksi sovellusten avulla (Benítez-Mejía ym., 2016).

Riskialttiit sovellukset puhelimessa voivat tehdä paljon tuhoa riippumatta siitä, onko käyttöjärjestelmä itsessään turvallinen vaiko ei. Sovelluksen kehittäjän kannalta on olennaista ohjelmoida sovelluksia, jotka eivät vaaranna käyttäjän yksityisyyt-

tä. Erilaisten palveluiden kuten verkkopankin mobilisoituessa on käyttäjän kannalta hyödyllistä tiedostaa mitä uhkia ladattavat sovellukset voivat sisältää (Haapala, 2016).

Käsittelen kandidaatintutkielmassani Android-sovellusten aiheuttamaa tietoturvauhkaa älypuhelimille sekä kuinka voidaan ehkäistä näitä. Tutkielmassani vastaan seuraaviin tutkimukselle asetettuihin kysymyksiin: Mitä vaaroja sovellukseen voi kohdistua? Mitkä asiat tekevät sovelluksesta vaarallisen ja kuinka näitä voidaan ottaa jo huomioon sovelluksen kehitysvaiheessa? Miten käyttäjä voi suojautua tietoturvahyökkäyksiltä? Kandidaatintutkielmani toteutuu systemaattisena kirjallisuuskatsauksena. Lähteistön olen rajannut aihepiirin perusteella.

Aluksi tutustutaan Android-käyttöjärjestelmän arkkitehtuuriin ja Android-sovelluksen rakenteeseen, jonka jälkeen tarkastellaan muutamia tyypillisiä tietoturvauhkia sovelluksissa. Tietoturvauhkia käsitellään sekä käyttäjän että sovelluksen kehittäjän näkökulmia ajatellen. Viimeiseksi esitetään tapoja, joilla sekä sovelluksen ohjelmoija että käyttäjä voi suojautua tietoturvamurroilta.

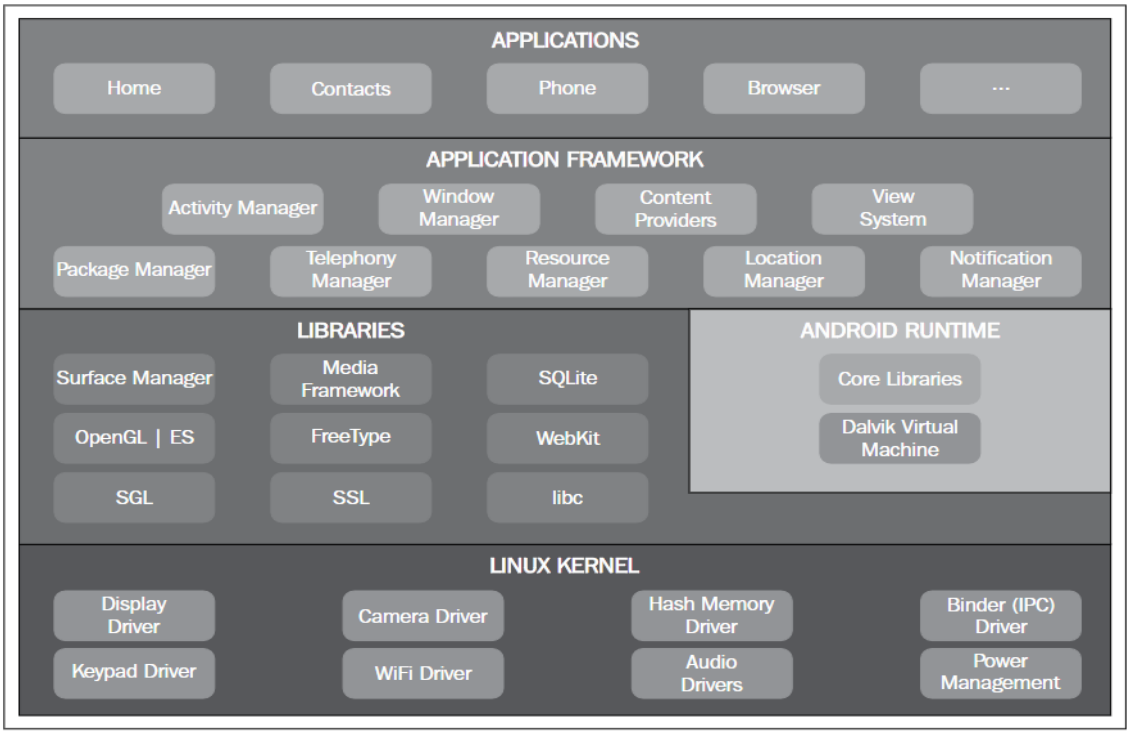
2 Android-sovellus

Android-älypuhelimissa sovellukset ovat keskeinen osa käyttäjän toimintaa. Sovellukset ovat yleensä ladattavissa Googlen Play Storesta, joka löytyy automaattisesti puhelimen oletussovelluksista. Play Storen lisäksi on myös mahdollista ladata sovelluksia internetistä. Sovellusten tarjonta on laaja ja vaihtelee maksullisten ja maksuttomien välillä. Java on yleisin ohjelmointikieli Android-sovelluksissa.

Käsittelen aluksi Androidin arkkitehtuuria, sillä mobiilikäyttöjärjestemän hahmoittaminen auttaa myös ymmärtämään sovelluksen toimintaa. Tämän jälkeen tarkastelen sovelluksen rakennetta.

2.1 Arkkitehtuuri

Pystyäkseen ymmärtämään sovelluksen toimintaa on tärkeää ymmärtää Androidin arkkitehtuuria. Androidin rakenne voidaan hahmottaa nelitasoisena. Alla oleva kuvio (kuvio 1) on Rain (2013) teoksesta Android Application Security Essentials ja auttaa hahmottamaan tämän selkeämmin.



Kuvio 1. Androidin rakenne (Rai, 2013)

Arkkitehtuurin alin taso on Linux kernel eli vapaasti suomennettuna Linuxin ydin huolehtii puhelimen laitteistosta. Tällöin pohjatason vastuulla ovat akun ja muistin hoitaminen, laiteohjaimet, verkoittuminen ja turvallisuus. Sovellusta asennettaessa Androidille sovellus saa omat UID ja GID-tunnuksensa. UID on lyhenne sanasta User Identification. Se on tunnus, jolla voidaan tunnistaa sovellus muiden joukosta. GID sen sijaan on viittaa Group Indetificationiin, ja sekä UID ja GID ovat olemassa niin kauan kunnes sovelluksen asennus poistetaan. Jotta Androidin käyttöjärjestelmä suostuu ajamaan sovelluksen pitää GID ja UID olla olemassa. (Gunasekera, 2012.)

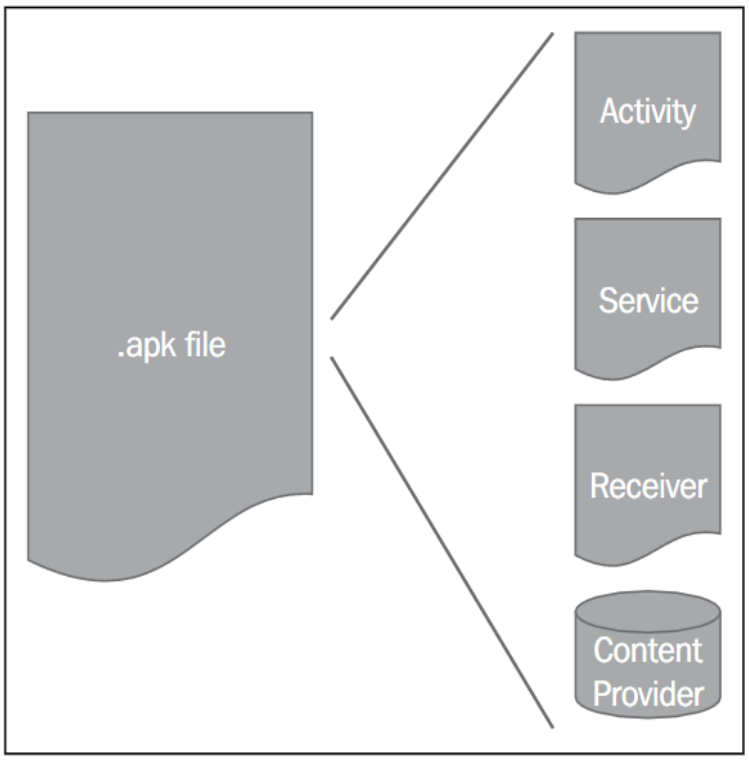
Alimman tason jälkeen on kirjastotaso, joka myös sisältää ajoympäristön sovelluksille. Kirjastotaso nimensä mukaisesti sisältää kirjastot koodin suorittamiseen. Ydinkirjastot ovat luotu C tai C++ kielillä (Gunasekera, 2012). Ajoympäristö sisältää Dalvikin virtuaalikoneen joka on vuorovaikutuksessa sovellusten kanssa. Virtuaalikoneen kehitti Dan Bornstein ja se pohjautuu Apache Harmony-nimiseen kirjastoon.

Jokainen sovellus ajetaan omalla virtuaalikoneellaan. Dalvik on toiminnaltaansa hyvin riippuvainen Linux alustastaan. Esimerkiksi virtuaalikoneen muistin hallinta on suoraan Linuxista saatu toimiva ominaisuus. Poikkeuksena on Dalvikin roskien kerääjä jokaiselle virtuaalikoneelle. (Rai, 2013.)

Sekä sovelluksen kehityskerros että sovelluskerros ovat aikaisemmista tasoista poikkeavasti toteutettu Javalla. Sovelluksen kehityskerros tunnetaan paremmin toiselta nimeltään API, joka tulee sanoista Application Programming Interface. Kehityskerros tarjoaa erilaisia palveluita ja systeemejä sovellusten kehittäjälle, kuten käyttöliittymän komponentteja kuten nappeja tai tekstikenttiä. Näiden lisäksi kehityskerrokseen kuuluu yleiset sisällöntuottajat, ilmoitusmanageri sekä aktiviteettien manageri. (Gunasekera, 2012.) Sovelluskerrokseen kuuluvat nimensä mukaisesti kaikki Androidin sovellukset, sekä oletussovellukset että käyttäjän lataamat. Mobiilikäyttöjärjestelmän oletussovelluksia ei ole käyttäjän tarkoitus poistaa, mutta käyttäjän itse lataamien sovelluksien asennukset ovat poistettavissa. (Rai, 2013.)

2.2 Sovelluksen rakenne

Sovelluksen rakenteen ymmärtäminen on tärkeää sen takia, että se helpottaa turvallisuusnäkökulman hahmottamista Androidin sovelluskerroksella. Kuten oheisesta kuviosta (kuvio 2) pystyy huomaamaan, sovelluksen rakenne periaatteessa muodostuu useista komponenteista jotka kerätään pinoon.



Kuvio 2. Sovelluksen rakenne (Rai, 2013)

Sovelluksen komponenteista muodostuminen mahdollistaa sen, että komponentit ovat itsenäisiä osia joita jopa muutkin sovellukset pystyvät kutsumaan halutessaan. On olemassa neljä eri komponenttia, joita Android tukee: toiminnallisuus, palvelu, radiovastaanotin ja sisällöntuottaja (kuvio 2). Kaikki nämä neljä komponenttia alustetaan tyypillisesti AndroidManifest.xml-tiedostossa. Toiminnallisuuskomponentti on tyypillisesti sovelluksen osa, joka vastaa käyttöliittymästä sovelluksessa. Toisin sanoen kaikki käyttäjän tuottama toiminnallisuus kuuluu tähän osaan. Tästä hyvänä esimerkkinä on kirjautuminen sisään sovellukseen. Palveluosa hoitaa taustalla toimivat prosessit, kuten esimerkiksi videon toistaminen. Radiovastaanotin komponenttia käytetään enimmäkseen sovelluksissa, joissa vastaanotetaan dataa kuten viestejä. Tyypillisesti tätä komponenttia hyödynnetään sähköpostisovelluksissa. Sisällöntuottajakomponentti toimii sovelluksen datavarastona, joka pystyy jakamaan dataa myös muiden sovellusten kanssa. (Rai, 2013.)

Kaikki neljä edellä mainitut komponentit pystyvät kommunikoimaan keskenään

niin kutsutun Intentin kautta. Intentillä viitataan Androidin asykronista mekanismia IPC:lle eli inter-process communicationille. Intent syntyy kun komponentti ilmoittaa aikovansa suorittaa toiminnan ja toinen komponentti reagoi vastaanottaessaan aikeen. On olemassa useita eri mekanismeja intentin lähettämiseen. Intentit eivät kuitenkaan ole suojattuja, vaan niitä voi seurata kuka tahansa. (Rai, 2013.)

3 Tietoturvauhat

Tietoturvahaavoittuvuudet Android-älypuhelimissa harvemmin poikkeavat tietokoneiden haavoittuvuuksista. Aivan kuten tietokonehakkeroinnissa, hyökkääjinä yleensä toimivat hakkerit, terroristit, kyberrikolliset ja bottiverkkojen operaattorit (Benítez-Mejía ym., 2016). Hyökkääjien motivaatio monesti vaikuttaa siihen, millä tavalla hyökkääjä hyödyntää löytämäänsä haavoittuvuutta edukseen. Yleensä tietoturvauhat jaetaan joko suoriin tai epäsuoriin. Epäsuorat hyökkäykset hyötyvät esimerkiksi huonosti suunnitellun sovelluksen takia (Gunasekera, 2012).

Tässä luvussa käsitellään tyypillisempiä haavoittuvuuksia, joita sovellukset voivat sisältää tai saattavat uhata sovellusta. Ensiksi on käyttöoikeuksien tarkastelua, sillä käyttöoikeuksilla on paljon merkitystä Androidin tietoturvan kannalta. Aina kun käyttäjä haluaa ladata älypuhelimeensa uuden sovelluksen, hänen pitää myöntää sovellusten vaatimille ominaisuuksille käyttöoikeus. Käyttöoikeuksien lisäksi tarkastellaan eri käyttötapauksia, jotka ovat aikaisemmin todettu olevan haitallisia Androidin sovellusten tietoturvan kannalta.

3.1 Käyttöoikeudet

Aikaisemmin kävimme Androidin arkkitehtuuria, jonka päällimäisenä kerroksena oli sovelluskerros. Käyttöoikeudet ajavat Androdin sovelluskerroksen turvallisuuspuolta, mutta vastuu on käyttäjällä joka lataa sovelluksia älypuhelimeensa. Käyttöoikeudet voidaan jakaa seuraavasti: normaali, vaarallinen, signeeraus tai signorsystem. Normaalit käyttöoikeudet ovat näistä harmittomimpia eivätkä yleensä aiheuta tietoturvauhkaa. Normaalit käyttöoikeudet eivät myöskään tarvitse käyttäjän lupaa, vaan heillä on suora yhteys tiettyihin älypuhelimen ominaisuuksiin. Näitä ovat esimerkiksi tärinä, taskulamppu tai taustakuva. Normaalin käyttöoikeudelle sallitut ominaisuudet eivät voi kuitenkaan aiheuttaa käyttäjälle vanhinkoa, korkeintaan ärsyttävät käyttäjää. (Rai, 2013.)

Vaaralliset käyttöoikeudet tarvitsevat aina käyttäjän luvan ennen sovelluksen asen-

tamista. Tällaisille luvan antaessa sovelluksella on pääsy älypuhelimen ominaisuuksiin kuten kameraan, mikrofoniin tai GPS:ään (Benítez-Mejía ym., 2016). Myös dataan käsiksi pääseminen on mahdollista. On olemassa sovelluksia, jotka tarvitsevat vaarallisiin käyttölupiin kuuluvan oikeuden mutta eivät hyödynnä sitä sovelluksessa. Esimerkiksi käyttäjän sijaintitietojen käyttäminen herättää kysymyksiä käyttäjän yksityisyyteen liittyen jos sovellus ei tarvitse sijaintietoa toimiakseen. Sovellukset, jotka haluavat luvan päästä käyttäjän tekstiviesteihin tai kalenteriin ovat mahdollisesti bottiverkkojen tai Troijanin hyödyntämiä sovelluksia (Rai, 2013). Signeerausoikeuksilla puhutaan siitä, kun kahdella sovelluksella on samat sertifikaatit ja täten saavat automaattisesti luvan päästä käsiksi toistensa komponentteihin (Benítez-Mejía ym., 2016). Parhaimmillaan signeerausoikeuksilla on mahdollista päästä käsiksi puhelimen systeemitason ominaisuuksiin kuten katkaisijaan tai tekstiviestien lähettämiseen. Signorsystem on tarkoitettu Androidin järjestelmäkuvan sovelluksille sekä sovelluksille, joilla on sama sertifikaatti kuin sovelluksella, joka on käyttöoikeuden määrittänyt. Tällaista käyttöoikeutta käytetään pääasiassa mm. laitevalmistajien luomissa sovelluksissa. Signorsystemin avulla sovellus pystyy esimerkiksi uudelleen käynnistämään laitteen. (Rai, 2013.)

Käyttäjien tietoturvan suojelemiseksi on ehdotettu, että tulevaisuudessa käyttäjä pystyy itse rajoittamaan sovelluksen haluamia käyttöoikeuksia tai nykyistä arkkitehtuuria muokattaisiin (Benitez-Mejia ym., 2016). Androidin käyttöoikeuksien rakennetta muuttamalla voitaisiin varmistaa, että käyttäjät ymmärtäisivät paremmin käyttöoikeuksien merkitystä (Obiri-Yeboah & Qi, 2016).

3.2 Remote Access Trojan

RAT eli Remote Access Trojan on haittaohjelma, jotka pyrkivät pääsemään käyttäjän henkilökohtaisiin tietoihin käsiksi. RAT sisältää sekä asiakkaan että serverin. Asiakaspuoli vastaa komennoista, kun taas serveripuoli hoitaa hakkerin haluamasta informaatiosta. Henkilökohtaisia tietoja, joita RAT käyttää hyödyksi ovat esimerkiksi käyttäjän sijainti, käyttäjätunnukset ja -salasanat, tekstiviestit. Hyökkäys onnistuu silloin, kun sovellus ladataan käyttäjän älypuhelimeen ja tätä kautta hyökkääjä

saa pääsyn uhrin puhelimeen. Sovellukset, jotka sisältävät Remote Access Troijanin, ovat tyypillisesti harmittomalta vaikuttavia sovelluksia kuten taskulamppu- tai pelisovellukset. Hyökkääjä saa nämä tiedot etänä ilman, että sovelluksen luojan tarvitsee olla käyttäjän lähistöllä. (Benitez-Mejia ym., 2016.)

3.3 Viestit

Eräs suosittu tapa hyökätä käyttäjän laitteeseen on lähettämällä käyttäjälle esimerkiksi teksti- tai sähköpostiviestin, joka sisältää haitallisen linkin. Linkin avaaminen johtaa sivustolle, joka heikentää puhelimen turvallisuutta. Linkin lisäksi voi olla myös viestin liittenä tiedostoja, joiden avaaminen altistaa puhelimen hakkerille. Huonoimmassa tapauksessa hyökkääjä voi päästä tätä kautta älypuhelimen tietoihin käsiksi. Sähköpostiviestien kautta on tämän lisäksi mahdollista ottaa käyttöön etähallinnan (engl. remote control) käyttäjän puhelimesta vakoilemiseen tai tietojen hyväksikäyttöön. Älypuhelimessa näkyvät pop-up viestit ovat niin kutsuttuja WAP PUSH-viestejä, joiden merkkimäärä on vähintään 160. WAP PUSH-viestit ovat hakkerille siinä mielessä hyödyllisiä, että viestissä näkyvä linkki aloittaa automaattisesti lataamisen käyttäjän laitteeseen ja sitä kautta vahingoittaa sitä. (Benitez-Mejia ym., 2016.)

3.4 SQL-injection attack

Jotkut sovellukset käyttävät tietokantaa esimerkiksi sovelluksen käyttäjien tietojen datavarastona. SQL-injektiohyökkäyksessä (engl. SQL-injection attack) hakkerit voivat pyytämättä antamaan SQL-komentoja sovelluksessa, joka ei pitäisi olla mahdollista. Tätä kautta hakkerit pystyvät pääsemään käsiksi sovelluksen sisällöntuottajaan, joka on hakkerin kannalta hyödyllisin sovelluksen osa. Sisällöntuottajaan käsiksi pääseminen tarkoittaa käytännössä sitä, että sovellukseen hyökkäävä pystyy hyödyntämään sovelluksen dataa enemmän kuin pitäisi. Sovelluksen koodissa SQLiteDatabase.rawQuery()-metodin käyttäminen altistaa sovelluksen SQL-injektiohyökkäyksille. (Makan & Scott, 2013.)

4 Tietoturvauhkien ehkäisy

Kaikkien tietoturvauhkien ehkäiseminen on haastavaa, mutta on asioita joihin sovelluksen kehittäjä voi vaikuttaa. Huolimattomuusvirheet sovelluksen kehitysvaiheessa altistavat sovelluksen hakkeroinnille. Sovelluksen kehittäjällä on myös mahdollisuus varmistaa, ettei sovellus toimita tietoja vääriin käsiin. Tässä luvussa käsitellään sitä, miten sovelluksen kehittäjä voi parantaa älypuhelinsovellusten tietoturvaa. Tarkastelen lyhyesti myös käyttäjän mahdollisuuksia tietoturvan parantamiseksi.

4.1 Kehittäjän näkökulma

Kehittäjällä on useita eri tapoja vaikuttaa oman sovelluksensa turvallisuuteen. Jo estääkseen epäsuoria tietoturvahyökkäyksiä on tärkeätä varmistaa, että sovelluksen rakenteessa ei ole heikkouksia. Sovelluksen turvallisuuden huomioiminen jo suunnitteluvaiheessa on olennainen osa toteutusta ja säästää monelta harmilta (Gunasekera, 2012). Koska Androidin pohja perustuu Linuxiin, olisi ohjelmoitsijan näkökulmasta kannattavaa käyttää sovelluksessa hiekkalaatikkotekniikkaa. Linuxin ohjelmissa hiekkalaatikkotekniikan käyttö on hyvin tyypillistä. Tälle tekniikalle ominaista on se, että sovellus erotetaan muista järjestelmäresursseista ja vaatii käyttöluvan ennen sovelluksen käyttöönottoa. Hiekkalaatikkotekniikan hyödyntäminen sovelluksessa on järkevää myös siksi, että jokaisella sovelluksella on oma hakemistonsa hiekkalaatikolle ja tarvitsee käyttäjältä luvan päästä tarvitsemaansa tietoon käsiksi (Tabassum & Kumari & Ghosh, 2014).

Sovelluksen komponenttien turvallisuus voidaan varmistaa kooditasolla. Tällöin huomio erityisesti kiinnittyy AndroidManifest.xml-tiedostoon, jossa määritetään komponentit. Määrittelemällä komponentin kuten kuviossa 3 voidaan varmistaa, ettei muut sovellukset pysty kutsumaan sitä. Tämä tapahtuu muuttamalla android:exported-attribuutin falseksi, joka estää ulkopuolisten sovellusten kykyä kutsua komponenttia. Samalla tavalla voi komponentin tilalle korvata haluamansa toiminnan. (Makan

& Scott, 2013.)

```
<[component name] android:exported="false">
</[component name]>
```

Kuvio 3. Komponentin suojaaminen (Scott, 2013)

AndroidManifest.xml-tiedostossa on myös hyvä määrittää komponenttien andoird:permissions attribuutti. Tämä johtuu siitä, että jos AndroidManifest.xml:ssä ei tätä määritetä, <application> lisää ne automaattisesti oletusarvojen mukaisesti. Tällöin jos tietää, ettei sovellus tai komponentit tarvitse luvatonta kommunikointia on nämä hyvä määritellä varmuuden vuoksi. Kun sovelluksen kehittäjä määrittää itse android:permissions-attribuutin, attribuutti ohittaa alkuperäiset oletusarvot. On myös mahdollista luoda omia käyttöoikeuksia, joissa voi määrittää itse käyttöoikeuden turvallisuustason. Edellisessä luvussa mainittiin jo aikaisemmin, että SQL-hyökkäyksille tietyt metodit, kuten SQLiteDatabase.rawQuery() altistaa koodin injektiohyökkäykselle. Sen sijaan kehittäjä voi esimerkiksi hyödyntää SQLiteDatabase-luokan binding-ominaisuutta. SQLiteDatabase-luokassa on vaihtoehtoisia metodeja, kuten query, insert ja update joiden käyttäminen ei aiheuta yhtä suurta uhkaa. (Makan & Scott, 2013.)

4.2 Käyttäjän näkökulma

Vaikka käyttäjällä on hyvin rajatut mahdollisuudet vaikuttaa oman älypuhelimensa turvallisuuteen, on tapa jonka avulla voi ehkäistä omaa älypuhelintaansa kyberhyökkäyksiltä. Kyseessä on käyttöoikeuksien tarkistaminen ennen asennusta. Jos sovellus pyytää lupaa päästä käsiksi laitteen ominaisuuksiin, joita sovellus ei tarvitse on turvallisempaa harkita tarvitseeko sovellusta ladata (Rai, 2013). Vaikka Androidin käyttäjät ovat enemmän tietoisia käyttöoikeuksista verrattuna muihin käyttöjärjestelmien käyttäjiin, saattaa osalle yhteys käyttöoikeuksien ja turvallisuuden välillä olla vielä epäselvää (Obiri-Yeboah & Qi, 2016). Käyttöoikeuksien hyväksymisen jälkeen käyttäjä ei voi juuri enään vaikuttaa sovellukseen (Rai, 2013).

5 Yhteenveto

Android-sovelluksien määrä on jatkuvasti kasvussa, ja sen takia on hyvä ottaa selville mitä vaaroja sovellukset voivat sisältää. Kandidaatintutkielma käsittelee Android-sovellusten turvallisuutta ja erityisesti sitä, kuinka Androidin käyttäjät ja ohjelmoijat pystyvät estämään tietoturvahyökkäyksiltä. Aluksi tarkasteltiin Androidin ja sovelluksen arkkitehtuuria, jossa huomataan mistä sovelluksen rakenne koostuu. Tällöin on helpompi ymmärtää kuinka sovellus toimii. Androidin arkkitehtuuri on monitasoinen, joista etenkin kaksi ylintä kerrosta ovat aktiivisesti vuorovaikutuksessa sovellusten kanssa. Sovellus rakentuu useasta itsenäisestä eri komponentista, joista kaikilla on eri vastuualueet. Komponentit pystyvät myös kommunikoimaan keskenään. AndroidManifest.xml-tiedosto on tärkeä sovelluksen kannalta, sillä siinä tapahtuu komponenttien alustukset ja määrittelyt.

Käyttäjän ja sovelluksen kehittäjän näkökulmia vertaillessa on nähtävissä suuri ero siinä, kuinka paljon kykenee vaikuttamaan sovelluksen turvallisuuteen. Sovelluksen käyttäjällä on tyypillisesti vähemmän keinoja päästä vaikuttamaan ladattavan sovelluksen turvallisuuteen, sillä sovellus pitää ladata joko sellaisenaan tai olla ilman (Obiri-Yeboah & Qi, 2016). Tärkeintä käyttäjän asemassa on varmistaa, ettei lataa epäluotettavia sovelluksia ja tarkistaa sovelluksen tarvitsemat käyttöoikeudet ennen asentamista. Sovelluksen kehittäjällä on sen sijaan enemmän vaihtoehtoja varmistaa, ettei ulkopuoliset pysty hyödyntämään sovellusta vääriin tarkoituksiin. Huolellisen suunnittelemisen ja käyttöoikeuksien rajoittamisen lisäksi kehittäjä pystyy soveltamaan jo Linuxista tuttua hiekkalaatikkomenetelmää. Hiekkalaatikkomenetelmä pitää sovelluksen erillään järjestelmäresursseista (Tabassum ym., 2014).

Käyttöoikeuksien merkitys tietoturvauhkien ehkäisemisessä on keskeinen. Nykyhetkellä Android-puhelimen tietoturva on pitkälti käyttäjästä kiinni. Sen vuoksi olisi syytä tarkastella lähemmin Androidin rakenteen muuttamista siten, ettei käyttäjä epätietoisena pysty tekemään vahinkoa laitteelleen sovelluksia asentaessaan. Rakenteen muuttaminen helpottaisi epäluotettavien sovellusten tunnistamista, sillä käyttäjä ei välttämättä pysty niitä päällisin puolin tunnistamaan. On olemassa

myös se vaara, että ulkopuolinen taho pääsee murtautumaan muutoin turvalliseen sovellukseen. (Benitez-Mejía ym., 2016; Obiri-Yeboah & Qi, 2016.)

Kirjallisuutta

Android Developers. *Android, the world's most popular mobile platform*. Saatavilla WWW-muodossa <URL: https://developer.android.com/about/android.html>. Viitattu 28.4.2017.

Benitez-Mejia, D. G. N. & Sanchez-Perez, G. & Toscano-Medina, L. K. 2016. *Android applications and security breach*. IEEE, s. 164–168.

Gommerstadt, H. & Long, D. 2012 *Android Application Security*. ACM, s. 1–9.

Gulista, T. & Shikha, K. & Nupur, G. 2014. *Android Application Security*. JETIR1407009. JETIR, s. 635–637.

Gunasekera, S. 2012. *Android Apps Security*. Apress.

Haapala, R. 2016. *Nordea aikoo luopua tunnuslukukorteista – tilalle tulee kaksi vaihtoehtoa*. Lappeenrannan uutiset, 12.6.2016. Saatavilla WWW-muodossa <URL: http://www.lappeenrannanuutiset.fi/artikkeli/403431-nordea-aikoo-luopua-tunnuslukukorteista-tilalle-tulee-kaksi- Viitattu 27.2.2017.

Makan, K. & Scott, A-B. 2013. *Android Security Cookbook : Practical Recipes to Delve Into Android's Security Mechanisms by Troubleshooting Common Vulnerabilities in Applications and Android OS Versions*. Birmingham, UK : Packt Publishing.

Nielsen. 2016. *TOPS OF 2015: DIGITAL*. Nielsen, 17.12.2015. Saatavilla WWW-muodossa <URL: http://www.nielsen.com/us/en/insights/news/2015/tops-of-2015-digital.html>. Viitattu 27.2.2017.

Obiri-Yeboah, J. & Qi, M. 2016 *Data Security of Android Applications*. IEEE, s. 1716–1721.

Rai, P. O. 2013. *Android Application Security Essentials*. Birmingham, UK : Packt Publishing.

Statista. 2017. *Number of Android smartphone users in the United States from 2014 to 2016 (in millions)*. Saatavilla WWW-muodossa <URL: https://www.statista.com/statistics/232786/forecast-of-andrioid-users-in-the-us/>. Viitattu 13.2.2017.

Statista. 2017. *Cumulative number of apps downloaded from the Google*

Play as of May 2016 (in billions)). Saatavilla WWW-muodossa <URL: https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/>. Viitattu 13.2.2017.

Saarelainen, A. 2001. *Android-sovellus voi lukea ja lähettää dataa luvatta*. Tivi, 13.4.2012. Saatavilla WWW-muodossa <URL: http://www.tivi.fi/Arkisto/2012-04-13/Android-sovellus-voi-lukea-ja-l%C3%A4hett%C3%A4%C3%A4-dataa-luvatta-3143393.html>. Viitattu 13.2.2017.