**Tuukka Pitkänen**

# Android-laitteiden tietoturvariskit ja niiden ehkäiseminen

Tietotekniikan kandidaatintutkielma

30. huhtikuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

**Tekijä:** Tuukka Pitkänen

**Yhteystiedot:** tuukka.p.pitkanen@student.jyu.fi

**Ohjaaja:** Tuomo Rossi

**Työn nimi:** Android-laitteiden tietoturvariskit ja niiden ehkäiseminen

**Title in English:** Security risks on Android devices and their prevention

**Työ:** Kandidaatintutkielma

**Sivumäärä:** 21+0

**Tiivistelmä:** Kandidaatintutkielma käsittelee Android-laitteiden tietoturvariskejä ja niiden ehkäisemistä. Tutkielmassa tuodaan esille sekä käyttöjärjestelmän, että sovellusten sisältämiä haavoittuvuuksia. Tietoisuuden lisääminen jo tunnistetuista haavoittuvuuksista on tärkeää, jotta tulevaisuudessa voidaan välttyä samankaltaisilta virheiltä helpommin. Androidin ollessa käytetyin käyttöjärjestelmä mobiililaitteissa, on tärkeää kiinnittää huomiota sen turvallisuuteen.

**Avainsanat:** Android, tietoturva, käyttöjärjestelmä, sovellus

**Abstract:** This bachelor´s thesis examines security risks and their prevention in context of Android-devices. Thesis shows different kinds of vulnerabilities in Android operating system but also in Android-apps. It is important to raise awareness of already known vulnerabilities in Android devices so such mistakes could be avoided in future with less effort. Android being the most used operation system in mobile devices it is important to focus on its security.

**Keywords:** Android, information security, operating system, application

# Kuviot

Kuvio 1. Androidiin kohdistuvien hyökkäysten luokittelu (Bhat ja Dutta 2019) ..... 3

# Taulukot

Taulukko 1. Vaarallisten käyttöoikeuksien ryhmittelyä mukailtuna Calciati ym. (2020) ja Alkindi, Sarrab ja Alzeidi (2019) taulukoista. ..... 10

# Sisällys

1 JOHDANTO ..... 1

2 ANDROID-KÄYTTÖJÄRJESTELMÄN TIETOTURVARISKIT ..... 3
2.1 Laitteistopohjaiset hyökkäykset ..... 4
2.2 Kerneliin kohdistuvat hyökkäykset ..... 4
2.3 HAL-kerrokseen kohdistuvat hyökkäykset ..... 6

3 ANDROID-SOVELLUSTEN TIETOTURVARISKIT ..... 7

4 TIETOTURVAN KEHITTÄMINEN ..... 11

5 YHTEENVETO ..... 14

LÄHTEET ..... 16

# 1 Johdanto

Android-käyttöjärjestelmä on Linux-ytimeen perustuva avoimen lähdekoodin käyttöjärjestelmä, joka on suunniteltu ensisijaisesti mobiililaitteille. Androidin kehittämisen aloitti vuonna 2003 Android, Inc, mutta vuodesta 2005 asti sen on omistanut Google. Ensimmäinen Android-käyttöjärjestelmällä varustettu puhelin julkaistiin 2008 (Javatpoint 2020). Käyttöjärjestelmä muodostuu useasta kerroksesta, jotka seuraavat ohjelmistopinomallia. Kerrokset koostuvat alijärjestelmistä, jotka ovat toteutettu Java- ja C/C++-kielillä (Mazuera-Rozo ym. 2019). Android-käyttöjärjestelmää hyödynnetään useissa eri laitteissa, kuten mobiililaitteissa, televisioissa ja autoissa. Aion kuitenkin Android-laitteella viitata tutkielmassani Androidiin pohjautuviin mobiililaitteisiin.

Älypuhelinkäyttäjiä on maailmanlaajuisesti jo yli 3 miljardia ja määrä jatkaa varmasti kasvua (Statista 2021b). Tammikuussa 2021 Android-käyttöjärjestelmällä varustettujen älypuhelimien osuus kaikista älypuhelimista oli 71.93% (Statista 2021a). Älypuhelimilla suoritetaan toimintoja yleisistä arkisista asioista luottamuksellisiin työtehtäviin. Molemmissa tapauksissa käsiteltävä data on tärkeää pitää yksityisenä, oli kyse sitten yrityksen liiketoiminnasta, tai käyttäjän valokuvakokoelmasta. Lisäksi Android-laitteiden yksityisyyteen on panostettava sen ollessa yleisimmin käytetty älypuhelin käyttöjärjestelmä. Suosiosta johtuen Android kiinnostaa haittaohjelmien kehittäjiä ja haavoittuvuuksia löydetään jatkuvasti lisää. Tutkijat ja kehittäjät tekevät jatkuvasti töitä turvallisuuden parantamiseksi, mutta tutkimusta on tärkeä tehdä lisää, jotta haavoittuvuuksien määrää ja vaikutuksia saadaan rajoitettua entistä enemmän.

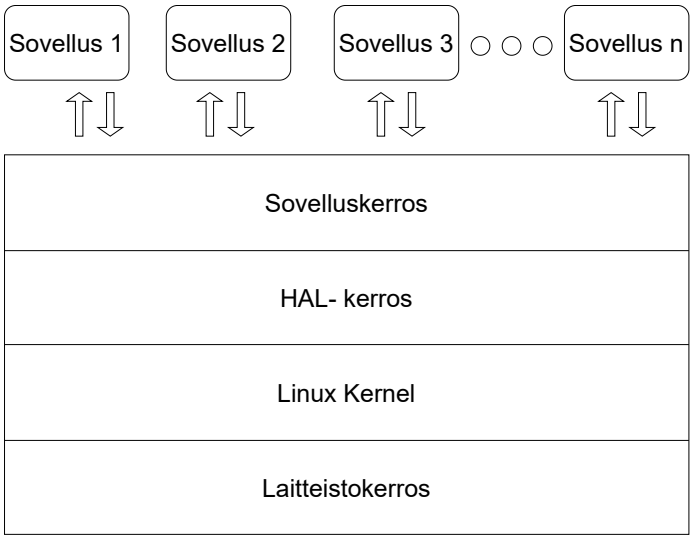
Tutkielmassani aion käydä läpi Android-laitteisiin liittyviä tietoturvariskejä ja mahdollisia tapoja kehittää niitä. Ensimmäiseksi käsittelen Android-laitteen käyttöjärjestelmän sisältämiä haavoittuvuuksia. Toiseksi esittelen Android-sovelluksien riskejä. Perustelen valintaani rajata sovellusten haavoittuvuudet omaan lukuunsa sillä, että sovellukset ovat oleellinen osa Android-laitteen käyttöä ja suuren sovellusten määrän lisäksi niistä on löydetty suuri määrä haavoittuvuuksia (Bhat ja Dutta 2019). Viimeiseksi ennen yhteenvetoa käyn läpi mahdollisia Android-käyttöjärjestelmän tietoturvan kehitystapoja. Androidin haavoittuvuuksista on tehty paljon tutkimuksia, johtuen mahdollisesti sen suosiosta. Androidista on tutkielman kirjoitus-

hetkellä julkaistu vuosittain tai useammin uusi versio (Javatpoint 2020). Uusien versioiden mukana on tullut muutoksia käyttöjärjestelmän vanhoihin ominaisuuksiin ja sen tietoturvaa on päivitetty. Tästä johtuen useat Androidiin liittyvät tutkimukset saattavat sisältää vanhentunutta tietoa, näin ollen olen pyrkinyt rajaamaan lähteeni mahdollisimman tuoreisiin.

# 2 Android-käyttöjärjestelmän tietoturvariskit

Mazuera-Rozo ym. (2019) mukaan suurin osa (69.97%) haavoittuvuuksista on hyödynnettävissä etäyhteyden kautta. Lisäksi ainakin puolessa tapauksista haavoittuvuudet ovat hyödynnettävissä siten, että saadaan koko laitteen hallinta. Järjestelmän haavoittuvuudet ovat järjestelmässä huomaamatta keskimäärin 770 päivää ennen tunnistamista, mahdollista hyväksikäyttöä ja korjaamista. Suurin osa Androidin haavoittuvuuksista liittyy vääränlaiseen pääsynvalvontaan, vääränlaisiin raja- ja syöttötarkastuksiin tai tyyppivirheisiin (Mazuera-Rozo ym. 2019).

Bhat ja Dutta (2019) luokittelevat artikkelissaan Android-järjestelmään kohdistuvat hyökkäykset neljään luokkaan. Luokat muodostuvat Androidin arkkitehtuurin kerrosten pohjalta. Kerrokset ovat käyttäjää lähimmästä tasosta lähtien: Sovelluspohjaiset hyökkäykset, HAL-kerrokseen (Hardware Abstraction Layer) kohdistuvat hyökkäykset, kerneliin kohdistuvat hyökkäykset ja Laitteistopohjaiset hyökkäykset. Sovelluspohjaisista hyökkäyksistä aion kertoa luvussa 3. Nämä tasot ovat esitetty Kuviossa 1. Esitetyssä kuviossa järjestelmässä on käynnissä useita sovelluksia, jotka voivat olla järjestelmässä oletusarvoisesti olevia tai erinäisistä lähteistä asennettuja.




Kuvio 1. Androidiin kohdistuvien hyökkäysten luokittelu, mukaillen Bhat ja Dutta (2019) tekemää kuviota

## 2.1 Laitteistopohjaiset hyökkäykset

Laitteistopohjaiset hyökkäykset kohdistuvat laitteistokomponenttien haavoittuvuuksiin. Laitteessa saattaa olla haavoittuvia komponentteja jo valmistajan toimesta. Haavoittuvia kolmannen osapuolen komponentteja on myös voitu asentaa jälkikäteen, esimerkiksi rikkoutuneen alkuperäisen komponentin tilalle (Bhat ja Dutta 2019).

Esimerkki laitteistopohjaisesta hyökkäyksestä on GLitch, joka on viimeisin versio Rowhammer-hyökkäyksestä. Rowhammer-hyökkäykset hyödyntävät haitallista JavaScript-koodia webselaimessa. Tätä kautta hyökkääjät pääsevät käsiksi laitteen muistiin (DRAM-komponenttiin) ja pystyvät muuttamaan tai vuotamaan sitä. GLitch käyttää hyväkseen WebGL-kirjastoa, joka renderöi grafiikkakoodia sitä tukevissa selaimissa. Kirjasto sisältää tunnetun muistisiruissa sijaitsevan häiriön (engl. glitch), joka saattaa laueta grafiikoita käsitellessä. Tästä on peräisin nimi GLitch (Bhat ja Dutta 2019).

## 2.2 Kerneliin kohdistuvat hyökkäykset

Kerneliin kohdistuvien hyökkäyksien tarkoitus on saada hyökkääjälle pääkäyttäjän oikeudet, jolloin hyökkääjä pystyy hallitsemaan laitetta (Mazuera-Rozo ym. 2019). Kerneli on Android-käyttöjärjestelmän ydin, jonka tehtävänä on muun muassa muistin ja laitteistokomponenttien hallinta (Tutorialspoint 2021). Kernel sisältää suuren osan Android-käyttöjärjestelmän tietoturva-aukoista. Useimmat kerneliin liittyvistä haavoittuvuuksista liittyvät laitteistokomponenttien ajureihin, joista yleisimmin haavoittuneita ovat muun muassa Wi-Fi, kamera ja grafiikkaprosessori (Mazuera-Rozo ym. 2019). Bhat ja Dutta (2019) luokittelevat kerneliin kohdistuvat hyökkäykset neljään luokkaan kernelin osien mukaan:

- **Pääkäyttäjän oikeuksia tavoittelevat hyökkäykset.** Käyttöoikeuksia tavoittelevissa hyökkäyksissä hyökkääjät käyttävät hyväkseen kernelistä löydettyjä haavoittuvuuksia. Gooligan-hyökkäys on Gooligan haittaohjelmaan perustuva pääkäyttäjän oikeuksia tavoitteleva hyökkäys. Gooligan käyttää hyväkseen 'zero-day'-haavoittuvuutta. Gooligan tekee kernelin muistiin sopimattomia luku- ja kirjoitusoperaatioita, jonka myötä hyökkääjät saavat lisää oikeuksia ja mahdollisuuden sijoittaa haitallista koodia (Bhat ja Dutta 2019).

- **Muistiin kohdistuvat hyökkäykset.** Muistiin kohdistuvat Code Reuse-hyökkäykset käyttävät hyväkseen muistin heikkouksia, jotka Bhat ja Dutta (2019) määrittelee tilanteiksi, joissa haitallinen koodi pystyy kirjoittamaan muistipaikkoihin epäsuorasti, esimerkiksi sijoittamalla varattua muistia suuremman määrän dataa sille tarkoitettuun paikkaan tai viittaamalla taulukon rajojen yli. ROP-hyökkäys (Return-oriented Programming) on esimerkki Code Reuse-hyökkäyksestä. ARM-arkkitehtuuri, johon Android-käyttöjärjestelmä perustuu, on haavoittuvainen ROP-hyökkäykselle. Hyökkäyksessä konetason käskyjen suorittamista sotketaan, jonka myötä hyökkääjä pääsee käsiksi käskyjen ulkopuolella oleviin muistipaikkoihin (Bhat ja Dutta 2019).
- **Bootloaderiin kohdistuvat hyökkäykset.** Bootloader on ohjelma, joka huolehtii tärkeistä tehtävistä systeemin käynnistyttyä, esimerkiksi käyttöjärjestelmän alustamisesta. Qualcommin bootloaderista löydetyt haavoittuvuudet CVE-2014-9798 ja CVE-2015-8893 antoivat hyökkääjille mahdollisuuden tehdä palvelunestohyökkäyksiä, muistivuotoja ja mahdollisuuden syöttää mielivaltaista koodia. Haavoittuvuudet johtuvat tarkastamattomista osoitteista (Bhat ja Dutta 2019).
- **Laitteiston ajureihin kohdistuvat hyökkäykset.** Laitteiston ajuri on käyttöliittymä, joka mahdollistaa tiedonsiirron ohjelmiston ja laitteiston välillä. Jokaiselle laitteistokomponentille on oma uniikki ajurinsa. Laitteiston ajureiden haavoittuvuudet antavat hyökkääjälle mahdollisuuden vahingoittaa ajuria ja sitä kautta kaataa koko järjestelmä tai saada kernelin ja sen myötä koko laitteen hallinta. Haavoittuvuuksia löytyy pääasiassa laitteeseen valmiiksi asennettujen sovellusten kuten Kameran tai Bluetoothin ajureista. Laitteistoajureiden pääasiallinen syy on kehittäjien keskittyminen ajureiden toiminnallisuuteen, tietoturvan sijaan. CVE-2016-2435 on NVIDIA:n näytönohjaimista löytynyt, Nexus-laitteissa vaikuttanut laitteiston ajureiden haavoittuvuus. Haavoittuvuuden avulla hyökkääjät pystyivät vioittamaan muistiarvoja, jonka myötä he saivat lopulta tarpeeksi käyttöoikeuksia koko järjestelmän hallitsemiseksi (Bhat ja Dutta 2019).

## 2.3 HAL-kerrokseen kohdistuvat hyökkäykset

HAL-kerros (Hardware Abstraction Layer) sisältää kirjastomoduuleja, jotka toteuttavat rajapinnat esimerkiksi kameralle, GPS:lle ja muille Android-järjestelmän osille. Tähän kerrokseen kohdistuvissa hyökkäyksissä hyökkääjät kohdentavat rajapinnan ja käyttävät komponentteja käyttäjän seuraamiseen. Mazuera-Rozo ym. (2019) löysivät tutkimuksessaan useita haavoittuvuuksia HAL-kerroksesta, mutta ne muodostivat vain 1.70% Android-käyttöjärjestelmän haavoittuvuuksista. Toisaalta, vaikka haavoittuvuuksien määrä ei ole suuri, voivat ne olla käyttäjän yksityisyyden kannalta vakavia. Esimerkiksi Vanhoef ja Piessens (2017) ovat paljastaneet Wi-Fi:n turvallisuusprotokollassa, WPA2:ssa, vakavia haavoittuvuuksia. He kertovat Key Reinstallion-hyökkäyksestä; Key Reinstallion-hyökkäyksessä Wi-Fi-kantaman päässä oleva hyökkääjä pystyy varastamaan yksityistä dataa uhrilta. Hyökkääjä voi hyödyntää laitteen mikrofonia, Kameraa ja muita komponentteja vakoillakseen laitteen käyttäjää. Bhat ja Dutta (2019) kuitenkin toteavat, että HAL-kerroksen turvallisuutta oli paranneltu huomattavasti Androidin versiossa Oreo, joka oli Androidin viimeisin versio tutkimuksen julkaisuhetkellä vuonna 2019.

# 3 Android-sovellusten tietoturvariskit

Sovelluspohjaisissa hyökkäyksissä haittaohjelmien kehittäjät hyödyntävät oletussovellusten sekä järjestelmään asennettujen sovellusten haavoittuvuuksia. Bhat ja Dutta (2019) kertovat, että Google on havainnut tietoturva-aukkoja yli 275 000 Google Play-Kaupasta olevista sovelluksessa. Mazuera-Rozo ym. (2019) havaitsivat tutkimuksessaan, että käyttöoikeus- ja pääsynvalvontaongelmat ovat haavoittuvuuksien pääasiallinen lähde sovelluspohjaisissa hyökkäyksissä. Toisaalta heidän tekemässään tutkimuksessa haavoittuvuudet oli eroteltu Androidin rakenteen mukaan useampaan luokkaan, jolloin toisin kuin jaossa jota käyttivät Bhat ja Dutta (2019), natiivikirjastot ja sovelluskehys ovat jaettu omiin luokkiinsa. Mazuera-Rozo ym. (2019) tekemässä jaossa sovelluspohjaisiin hyökkäyksiin sisältyy myös natiivi- ja kolmannen osapuolen kirjastojen haavoittuvuuksien hyödyntäminen. Bhat ja Dutta (2019) mainitsevat hyökkääjien hyödyntävän kirjastoja saadakseen oikeuksia laitteen komponentteihin. Sufatrio ym. (2015) kertovat, että Android-sovellusten takaisinmallinnus on suhteellisen helppoa verrattaessa esimerkiksi Windows- ja Unix-ohjelmiin. Tämän seurauksena Android-sovellukset voidaan pakata uudelleen, mukanaan haittaohjelmia.

Android-sovellukset vaativat toimiakseen käyttöoikeuksia, joita sovelluksen on pyydettävä käyttöjärjestelmältä. Käyttöoikeuksia ovat esimerkiksi oikeus käyttää laitteen sijaintia tai laitteen kameraa. Näin Android-käyttöjärjestelmä pyrkii estämään käyttäjän arkaluontoisten tietojen ja systeemin kriittisten ominaisuuksien väärinkäyttöä. Tapaa, jolla Android-käyttöjärjestelmä käsittelee sovellusten käyttöoikeuksia, kutsutaan lupamalliksi (eng. permission model) (Calciati ym. 2020).

Calciati ym. (2020) kertovat Android sovellusten lupamallin historiasta. Androidin lupamallin historia voidaan jakaa kahteen osaan julkaistujen Android-versioiden mukaan. Vuoden 2015 syksyllä julkaistussa Androidin versiossa 6 (Marshmallow) lupamalliin tuli aiempaan verraten oleellinen muutos. Aiemmissa Androidin versioissa sovelluksen vaatimat käyttöoikeudet pyydettiin ennen sovelluksen asentamista. Vain päivitysten myötä sovellukselle oli mahdollista saada lisää käyttöoikeuksia. Vanha lupamalli sisälsi useita epäkohtia: Käyttäjä ei välttämättä osannut arvioida sovelluksen pyytämien oikeuksien merkitystä, jolloin käyttäjä asensi sovelluksen, tietämättä liittyvätkö oikeudet sovelluksen toimintaan. Toisaalta, jos

käyttäjä havaitsisikin sovelluksen pyytävän epäilyttäviä oikeuksia, vaihtoehtoina oli joko asentaa sovellus huolimatta epäilyttävistä käyttöoikeuksista tai olla käyttämättä sovellusta. Sovelluksen elinkaaren aikana kehittäjien lisätessä ominaisuuksia, ja sen mukana lisää vaadittavia käyttöoikeuksia, sovelluksen oli pyydettävä lisää oikeuksia päivitysten mukana. Tämä johti siihen, että käyttäjät saattoivat jättää asentamatta sovellusten uudemmat päivitykset. Sovelluskehittäjät alkoivatkin usein pyytämään sovelluksen aikaisimmista versioista asti ylimääräisiä oikeuksia, jotta kerran asennetun sovelluksen ei tarvitsisi pyytää lisää oikeuksia, ominaisuuksien lisääntyessä. Tämän seurauksena useilla sovelluksilla oli liikaa oikeuksia, jotka aiheuttivat mahdollisia haavoittuvuuksia. Android version 6 mukana esiteltiin nykyinen lupamalli, jossa sovelluksen vaatimat oikeudet kysytään dynaamisesti, ja lisäksi eri oikeudet ovat jaettu luparyhmiin (eng. permission groups).

Uudesta lupamallista kertovat Alkindi, Sarrab ja Alzeidi (2019). Uudessa lupamallissa erityyppiset käyttöoikeudet ovat jaettu eri luokkiin vaarallisuutensa mukaan. Vaarattomin luokka on 'normaali'-luokka, johon kuuluvat oikeudet käyttöjärjestelmä hyväksyy automaattisesti kysyttäessä. Tällaisia ovat muun muassa oikeus internetyhteyteen, bluetoothiin tai taustakuvan vaihtamiseen. Seuraava luokka on 'vaarallinen'-luokka, jotka vaativat hyväksynnän käyttäjältä. Nämä oikeudet koskevat toimintoja, joilla sovellus voi loukata käyttäjän yksityisyyttä helposti, kuten oikeus kameraan tai viestien lukemiseen/lähettämiseen. Vaarallisia käyttöoikeuksia on esitelty ryhmittäin taulukossa 1. Jos käyttäjä hyväksyy yhden ryhmän oikeuksista, on sovelluksella sen jälkeen oikeus ryhmän muihin käyttöoikeuksiin. Viimeisenä 'signature'-luokka, joka on vaarallisin. Tämän luokan oikeudet saavat vain ne sovellukset, jotka ovat kirjautuneet samalla sertifikaatilla, kuin luvan myöntäjä. Nämä oikeudet myönnetään jo sovellusta asentaessa automaattisesti niille, jotka niitä pyytävät ja joilla on oikeus niihin. Näillä oikeuksilla on mahdollista esimerkiksi poistaa käyttäjän dataa tai kirjoittaa laitteen asetuksia uudelleen.

Uudessa lupamallissa käyttäjän on hyväksyttävä vaaralliset oikeudet ja käyttäjän on mahdollista myöhemmin evätä hyväksymiään oikeuksia sovelluksilta. Sovellus kysyy vaarallisia oikeuksia vasta tarvittaessa, toisin kuin vanhassa mallissa, jossa oikeudet piti hyväksyä jo ennen sovelluksen asentamista. Näin käyttäjällä on mahdollisuus arvioida mihin sovellus käyttää oikeutta ja esimerkiksi jättää käyttämättä vain osaa sovelluksesta (Calciati ym. 2020)

(Alkindi, Sarrab ja Alzeidi 2019). Alkindi, Sarrab ja Alzeidi (2019) kuitenkin toteavat, että käyttäjä ei vieläkään voi tietää mihin sovellus tarvitsee oikeutta. Lisäksi normaalien oikeuksien, joita käyttäjä ei siis voi kontrolloida, joukossa on oikeuksia, joiden väärinkäytöllä voi olla vakavia seurauksia. Vaikka käyttäjän on tarkoitus olla kontrollissa vaarallisten oikeuksien suhteen, Calciati ym. (2020) näyttivät, että usein sovellukset pyytävät uusia vaarallisia oikeuksia ja Android myöntää ne kysymättä käyttäjältä. Heidän analyysinsä paljasti, että sovellukset käyttävät 56% automaattisesti myönnetyistä vaarallisista oikeuksista. Lisäksi he näyttivät, että automaattisesti hyväksyttyjä oikeuksia on käytetty käyttäjän yksityisen datan saamiseksi. Sovellus on sitten välittänyt saatua dataa eteenpäin kolmansille osapuolille tai sovelluksen kehittäjille.

| **Käyttöoikeusryhmä** | **Käyttöoikeudet** |
|---|---|
| Kalenteri | READ_CALENDAR, WRITE_CALENDAR |
| Mikrofoni | RECORD_AUDIO |
| Kamera | CAMERA |
| Puhelin | READ_PHONE_STATE, CALL_PHONE, READ_CALL_LOG, WRITE_CALL_LOG, ADD_VOICEMAIL, USE_SIP, PROCESS_OUTGOING_CALLS |
| Sensorit | BODY_SENSORS |
| Sijainti | ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION |
| Tallennustila | READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE |
| Tekstiviestit | SEND_SMS, RECEIVE_SMS, RECEIVE_MMS, READ_SMS, RECEIVE_WAP_PUSH |
| Yhteystiedot | READ_CONTACTS, WRITE_CONTACTS, GET_ACCOUNTS |

Taulukko 1. Vaarallisten käyttöoikeuksien ryhmittelyä mukailtuna Calciati ym. (2020) ja Alkindi, Sarrab ja Alzeidi (2019) taulukoista.

# 4 Tietoturvan kehittäminen

Android-laitteet siis sisältävät useita haavoittuvuuksia, vaikka huomioitaisiin vain toistaiseksi tunnistetut. Tutkijat ovat ehdottaneet useita kehitysvaihtoehtoja Android-laitteiden tietoturvalle. Huomioimalla eri tutkimuksissa esiin nostetut ehdotukset, on mahdollista kehittää Androidin tietoturvaa siten, että vältetään tulevaisuudessa tarpeettomia tietoturvariskejä.

Sufatrio ym. (2015), Bhat ja Dutta (2019) ja Garg ja Baliyan (2021) kertovat haittaohjelmien tunnistusmenetelmistä: Tutkijat ovat kehittäneet työkaluja ja tekniikoita haitallisen toiminnan tunnistamiseksi. Haitallisen toiminnan tai sovellusten havaitsemiseksi ja estämiseksi on kaksi päämenetelmätyyppiä: Staattiset menetelmät ja dynaamiset menetelmät:

Staattisessa menetelmässä työkalu voi havaita haittaohjelman tai haitallisen toiminnan ennen sovelluksen suorittamista. Analyysin suorittaminen ilman, että ohjelmaa tarvitsee suorittaa, tekee menetelmästä tehokkaan ajan ja resurssien puolesta.

Dynaamisessa menetelmässä työkalu havaitsee haitallisen toiminnan ajon aikana. Dynaamisten menetelmien etu on niiden kyky tunnistaa 'koodin hämärtämis'-hyökkäykset, jolloin toisin kuin staattisia menetelmiä, dynaamista ei ole yhtä helppo kiertää.

Yleisin eri artikkeleiden esiin nostama kehitystapa Android-laitteiden tietoturvalle onkin erilaisten automaattisten analysointimenetelmien hyödyntäminen. Eri menetelmiä tulisi hyödyntää sekä laitteeseen, että sovelluksiin niin kehitys- kuin käyttövaiheissa. Bhat ja Dutta (2019) ehdottavat, että Google Playn tulisi kehittää staattinen analyysimenetelmä, joka hyödyntäisi koneoppimista ja tekoälyä. Tällöin sovellusten tietoturvariskien hyväksikäyttö voitaisiin mahdollisesti estää jo ennen sovelluksen asentamista. Koska Google Play-kauppaan toteutettu staattinen menetelmä ei todennäköisesti onnistuisi havaitsemaan kaikissa sovelluksissa olevia haavoittuvuuksia, tulisi jo asennettuihin sovelluksiin hyödyntää dynaamista tunnistusmenetelmää. Dynaamisen tunnistusmenetelmän keräämää dataa voitaisiin käyttää kouluttamaan Google Play-kaupan staattista menetelmää, jolloin pitkällä aikavälillä riskejä saataisiin ehkäistyä tehokkaammin. Garg ja Baliyan (2021) toteavatkin erityyppisten analyysimenetelmien yhdistämisen olevan mahdollinen ratkaisu tarkemman havaitsemisen saamiseksi. Lisäksi Garg ja Baliyan (2021) huomauttavat Android-sovellusten sisältävän koodia

useissa eri muodoissa, kuten Java-koodina, natiivina C/C++-koodina ja Dalvik-tavukoodina. Näin ollen tulee huolehtia, että analyysi kattaa eri muodoissa olevat koodit.

Mazuera-Rozo ym. (2019) painottavat haavoittuvuuden tunnistusmenetelmien pakollista käyttöä laitteiston ohjelmakoodiin. He osoittivat, että haavoittuvuuksien havaitsemiseen saattaa kulua paljon aikaa, jonka takia on tärkeää hyödyntää automatiikkaa tunnistamisessa. Haavoittuvuuden tunnistaminen voitaisiin keskittää eniten haavoittuvaisten kerrosten heikkouksiin, tehokkaiden tulosten saamiseksi.

Mazuera-Rozo ym. (2019) kertovat myös aiempaan tutkimukseen pohjautuen, automaattisen koodikorjauksen toimivan 9% bugikoodeista, ja he uskovat osuuden olevan mahdollisesti suurempi ongelmien ollessa yksinkertaisia. Tästä syystä Mazuera-Rozo ym. (2019) ehdottavat automaattisten koodausvirheiden korjaajien käyttöä, sillä useat koodissa sijaitsevat haavoittuvuudet ovat korjattavissa yksinkertaisilla muutoksilla, esimerkiksi raja- ja nulltarkastuksilla.

Toisena julkaisuja yhdistävänä havaintona on kehittäjän rooli tietoturvassa. Bhat ja Dutta (2019) kertovat Android-sovellusten kehittäjien jättävän käyttämättä SSL-sertifikaattia, vaikka Android-sovelluskehitys-paketti tarjoaa paketit sen käyttämiseen. Tämä johtuu SSL-sertifikaattia käyttävien sovellusten aikaa vievästä debuggaamisesta. SSL-protokollan pakollinen noudattaminen kuitenkin vähentäisi mahdollisia langattomiin yhteyksiin kohdistuvia hyökkäyksiä.

Mazuera-Rozo ym. (2019) korostavat myös koodauskäytänteiden parantamista. He toteavat Android-käyttöjärjestelmän eri haavoittuvuuksien yleisyyden vaihtelevan eri kerrosten perusteella, mutta yleisimmät tapaukset liittyvät pääasiassa tarkastamattomiin muistitarkastuksiin, poikkeustilanteiden käsittelyihin tai muihin koodausvirheisiin.

Muita ilmi tulleita kehityskohteita Android-laitteiden turvallisuudelle: Bhat ja Dutta (2019) vaativat Android-sovellusten käyttöoikeuksien tarkentamista tehtäväkohtaiseksi. Näin voitaisiin välttyä antamasta oikeuksia kolmannen osapuolen kirjastoille, jotka saattavat väärinkäyttää oikeuksia. Lisäksi lupajärjestelmää tulisi kehittää vankemmaksi, jotta vähemmän vaarallisina pidettyjen oikeuksien hyväksikäyttö olisi vaikeampaa. Bhat ja Dutta (2019) ehdottavat, että sovellusten asentaminen olisi mahdollista vain Google Play-kaupasta, jolloin voitaisiin

varmistaa sovellusten turvallisuus ennen niiden käyttöönottoa, olettaen että Googlen sovellusten analyysimenetelmä kehitettäisiin nykyistä paremmaksi. Lisäksi Bhat ja Dutta (2019) esittelevät muutamia työkaluja, joilla voitaisiin välttää käyttöoikeuksien väärinkäyttöä. Esimerkiksi AdSplit erottaa sovelluksen toiminnan ja sovelluksen sisältämien mainosten tarvitsemat kirjastot eri prosesseihin, jolloin on mahdollista seurata ja rajoittaa tarkemmin mainosten tekemiä toimintoja. Viimeisenä, Garg ja Baliyan (2021) korostavat kokonaisuuden huomioimista ja eri menetelmien yhdistämistä Android-laitteiden turvallisuuden kehittämisessä.

# 5 Yhteenveto

Tutkielmassani pyrin esittelemään lähdekirjallisuuteen pohjautuen Android-laitteissa tunnistettuja tietoturvariskejä, ja joitain esiteltyjä menetelmiä tietoturvariskien ehkäisemiseksi tulevaisuudessa. Android on ensisijaisesti mobiililaitteille suunniteltu käyttöjärjestelmä, jolla on jo yli 2 miljardia käyttäjää. Android perustuu avoimeen lähdekoodiin ja on nykyään Googlen omistama. Erilaiset sovellukset ovat käytännössä tapa, jolla laitteen käyttäjä hyödyntää Android-laitettaan. Käyttöjärjestelmän suuri käyttäjämäärä on kannustanut hyökkääjiä löytämään ja hyödyntämään laitteiden haavoittuvuuksia. Suuresta käyttäjämäärästä johtuen haavoittuvuudet vaikuttavat laajasti, ja on tärkeää tuoda esiin löydettyjä haavoittuvuuksia ja niiden vaikutuksia, jotta tulevaisuudessa kehittäjät voivat kiinnittää tarkempaa huomiota eri kehitysvaiheiden turvallisuuteen.

Tavanomaisesti Android-käyttöjärjestelmän haavoittuvuudet jaetaan käyttöjärjestelmän kerrosten mukaan. Kerroksiin pohjautuvaa haavoittuvuuksien luokittelua käyttävät muun muassa Mazuera-Rozo ym. (2019), Bhat ja Dutta (2019) ja Garg ja Baliyan (2021). Android-käyttöjärjestelmä koostuu kerroksista, jotka toimivat toisistaan erillään, ja eri kerroksiin vaikuttavat yleisimmät haavoittuvuustyypit ovatkin erilaisia (Mazuera-Rozo ym. 2019). Erityisesti nostan esiin käyttöjärjestelmän ytimen eli kernelin sisältämät haavoittuvuudet. Kernelin sisältämät haavoittuvuudet ovat yleisimpiä, ja jatkossa tulisikin käyttää enemmän resursseja sen turvallisuuden parantamiseksi (Mazuera-Rozo ym. 2019).

Sovellusten ollessa oleellinen osa Android-laitteen käyttöä, toin esille tutkielmassani myös niihin liittyviä tietoturvariskejä. Bhat ja Dutta (2019) kertovat Googlen havainneen tietoturva-aukkoja yli 275 000 Google Play-Kaupassa olevista sovelluksessa. Toisaalta myös järjestelmäsovellukset voivat sisältää haavoittuvuuksia. Lisäksi Androidin sovellusten käyttöoikeuksien hallintajärjestelmä sisältää puutteita: Lupa vaarallisiin käyttöoikeuksiin, joiden avulla sovelluksen on helppo loukata käyttäjän yksityisyyttä, saatetaan hyväksyä käyttöjärjestelmän toimesta, vaikka niiden tulisi vaatia käyttäjän hyväksyntää (Calciati ym. 2020).

Eri tutkijat ovat kuitenkin esittäneet ratkaisukeinoja haavoittuvuuksien estämiseksi Android-laitteissa. Tärkeimpänä tutkimuksia yhdistävänä kohtana on automatiikan hyödyntäminen.

Esimerkiksi automaattisia haavoittuvuuksien tunnistamismenetelmiä tulisi hyödyntää, sekä sovelluksiin, että käyttöjärjestelmän eri kerroksiin. Tunnistusmenetelmien käytössä voitaisiin myös hyödyntää tekoälyä, jotta voitaisiin saavuttaa parempia tuloksia tulevaisuudessa.

Tietoturvaan ei voi koskaan panostaa liikaa, ja Android-laitteet eivät ole ainut tietoturvariskejä sisältävä teknologia, mutta tulevaisuudessa on vielä paljon tehtävää Android-käyttäjien tietoturvan takaamiseksi.

# Lähteet


Alkindi, Zainab, Mohamed Sarrab ja Nasser Alzeidi. 2019. “Android Application Permission Model Issues and Privacy Violation”. Huhtikuu. https://www.researchgate.net/publication/332401070_Android_Application_Permission_Model_Issues_and_Privacy_Violation.

Bhat, Parnika, ja Kamlesh Dutta. 2019. “A Survey on Various Threats and Current State of Security in Android Platform”. *ACM Comput. Surv.* (New York, NY, USA) 52, numero 1 (helmikuu). ISSN: 0360-0300. https://doi.org/10.1145/3301285. https://doi.org/10.1145/3301285.

Calciati, Paolo, Konstantin Kuznetsov, Alessandra Gorla ja Andreas Zeller. 2020. “Automatically Granted Permissions in Android Apps: An Empirical Study on Their Prevalence and on the Potential Threats for Privacy”. Teoksessa *Proceedings of the 17th International Conference on Mining Software Repositories,* 114–124. MSR ’20. Seoul, Republic of Korea: Association for Computing Machinery. ISBN: 9781450375177. https://doi.org/10.1145/3379597.3387469. https://doi.org/10.1145/3379597.3387469.

Garg, Shivi, ja Niyati Baliyan. 2021. “Android security assessment: A review, taxonomy and research gap study”. *Computers  Security* 100:102087. ISSN: 0167-4048. https://doi.org/https://doi.org/10.1016/j.cose.2020.102087. https://www.sciencedirect.com/science/article/pii/S0167404820303606.

Javatpoint. 2020. *Android Versions.* https://www.javatpoint.com/android-versions.

Mazuera-Rozo, Alejandro, Jairo Bautista-Mora, Mario Linares-Vásquez, Sandra Rueda ja Gabriele Bavota. 2019. “The Android OS stack and its vulnerabilities: an empirical study”. *Empirical Software Engineering* (tammikuu). https://doi.org/10.1007/s10664-019-09689-7. https://doi.org/10.1007/s10664-019-09689-7.

Statista. 2021a. *Mobile operating systems’ market share worldwide from January 2012 to January 2021.* https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/#statisticContainer.

Statista. 2021b. *Number of smartphone users worldwide from 2016 to 2021.* https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

Sufatrio, Darell J. J. Tan, Tong-Wei Chua ja Vrizlynn L. L. Thing. 2015. "Securing Android: A Survey, Taxonomy, and Challenges". *ACM Comput. Surv.* (New York, NY, USA) 47, numero 4 (toukokuu). ISSN: 0360-0300. https://doi.org/10.1145/2733306. https://doi-org.ezproxy.jyu.fi/10.1145/2733306.

Tutorialspoint. 2021. *Android - Architecture.* https://www.tutorialspoint.com/android/android_architecture.htm.

Vanhoef, Mathy, ja Frank Piessens. 2017. "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2". Teoksessa *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security,* 1313–1328. CCS '17. Dallas, Texas, USA: Association for Computing Machinery. ISBN: 9781450349468. https://doi.org/10.1145/3133956.3134027. https://doi.org/10.1145/3133956.3134027.