Android-laitteiden tietoturvatilanteen kartoittaminen data-analytiikan avulla

Diplomityö
Turun yliopisto
Tietotekniikan laitos
Ohjelmistotekniikka
2025
Elmo Kilkki

TURUN YLIOPISTO
Tietotekniikan laitos

Elmo Kilkki: Android-laitteiden tietoturvatilanteen kartoittaminen data-analytiikan avulla

Diplomityö, 55 s., 18 liites.
Ohjelmistotekniikka
Toukokuu 2025

Mobiililaitteet ovat nykyään yleinen apuväline työelämässä ja ne voivat siksi sisältää arkaluontoista tietoa sekä työntekijöistä että organisaatioista. Laitteiden tietoturvan valvominen on kuitenkin organisaatioille vaikeaa, koska laitteet liikkuvat usein valvottujen tilojen ulkopuolella ja työntekijöillä voi olla käytössä useita erilaisia laitteita.

Tämän tutkielman tavoitteena oli selvittää, millaista dataa organisaatioiden tulisi kerätä työntekijöidensä Android-laitteista tietoturvatilanteen kartoittamiseksi ja miten dataa voitaisiin kerätä laitteeseen asennetun sovelluksen kautta. Tutkielmassa määriteltiin Android-laitteiden resursseja, joista hyökkääjä voisi olla kiinnostunut. Tässä käytettiin apuna sovelluksille myönnettäviä Android-permissioita, jotka suojaavat laitteen resursseja väärinkäytöltä. Laitteisiin kohdistuvia uhkia puolestaan kartoitettiin tarkastelemalla tietoturvaorganisaatioiden julkaisemia vuosikatsauksia ja tilastoja viime vuosien ajalta. Näiden lähteiden pohjalta päädyttiin perehtymään tarkemmin pankkipalveluihin kohdistuviin haittaohjelmiin. Haittaohjelmien käyttäymistä tutkittiin MITRE ATT&CK -mallin avulla, jolloin kyettiin määrittelemään data, jota Android-sovellusrajapinnan kautta tulisi kerätä haittaohjelmien tunnistamiseksi. Tuloksena tuotettiin java-koodinäytteet, joiden avulla pankkihaittaohjelmien tunnistamiseen liittyvää dataa voidaan kerätä Android-sovellusrajapinnan kautta.

Lopuksi tutkielmassa suoritettiin käytännön koe, jossa testattiin haittaohjelman tunnistamista Android-laitteesta kerätyn datan perusteella. Kokeen perusteella MITRE ATT&CK -mallin avulla tunnistettu data vaikutti lupaavalta ratkaisulta haittaohjelmien tunnistamiseen, mutta datan analysoiminen vaatisi jatkokehitystä väärien hälytysten minimoimiseksi.

Asiasanat: Tietoturva, Android, MITRE ATT&CK, haittaohjelmat, data-analytiikka

Sisällys

1 Johdanto **1**

2 Tietoturva Android-laitteissa **3**

2.1 Laitteistotason tietoturva 3

2.2 Keystore ja avainten käsittely 5

2.3 Käyttöjärjestelmä 6

2.3.1 Prosessien ja sovellusten eristäminen 6

2.3.2 SELinux 6

2.3.3 Lupien myöntäminen sovelluksille 7

2.4 Tunnistautuminen 8

2.5 Datan suojaus 9

2.6 Päivitykset 10

3 Hyökkäykset ja uhat **11**

3.1 Android-laitteiden resurssit 11

3.1.1 Sovellusten permissiot 12

3.1.2 Yhteenveto resursseista 13

3.2 Hyökkäysten mallintaminen 14

3.2.1 MITRE ATT&CK 15

3.2.2 Cyber Kill Chain 17

3.2.3 STRIDE 18

3.3 Mobiililaitteisiin kohdistuvat uhat 18
3.4 Pankkihaittaohjelmien käyttämät MITRE-tekniikat 21

4 Hyökkäyksiin viittaavan datan kerääminen 29
4.1 Laitteen resurssien käytön tunnistaminen 29
4.2 Sovellusten käyttämien permissioiden kerääminen 31
4.3 MITRE-tekniikoden käytön tunnistaminen 32
4.3.1 Tietojen syöttäminen UI:lle - T1516 32
4.3.2 Järjestelmänvalvojan oikeudet - T1626.001 34
4.3.3 Kontaktien keräys - T1636.003 35
4.3.4 Notifikaatiot - T1517 35
4.3.5 Kommunikaatiokanavat - T1644 36
4.3.6 Verkkotietojen keräys - T1422 ja T1422.001 37
4.3.7 Palvelunesto - T1642 37
4.3.8 SMS (lukeminen) - T1636.004 38
4.3.9 SMS (hallinta) - T1582 38
4.3.10 Luotettava nimeäminen - T1655.001 39
4.3.11 Näppäilyjen tarkkailu - T1417.001 39
4.3.12 GUI syötteiden keräys - T1417.002 41
4.4 Yhteenveto kerättävästä datasta 41

5 Hyökkäysten visualisointi ja tunnistaminen 45
5.1 Koe datan keräämiseen ja visualisointiin 45
5.1.1 Koeasetelma 45
5.1.2 Haittaohjelmana toimiva Android-sovellus 46
5.1.3 Android-sovellus datan keräämiseen 48
5.2 Kokeen tulokset 48
5.2.1 Tulosten visualisointi 49

5.2.2 Tulosten arviointi 49

6 Yhteenveto **54**

Lähdeluettelo **56**

Liitteet

A Pankkihaittaohjelma (Android-sovellus) **A-1**

B Android-sovellus datan keräämiseen **B-1**

Kuvat

2.1 Secure-element (oikealla) on eristetty Android-käyttöjärjestelmästä sekä suojatusta suoritusympäristöstä (TEE). 5

3.1 MITRE ATT&CK Android-matriisi, jossa taktiikat on esitelty ylimällä rivillä. Kunkin taktiikan alle on listattu niihin liittyvät tekniikat. 16

5.1 (Laite A) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä. 50

5.2 (Laite B) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä. 51

5.3 (Laite C) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä. 52

Taulukot

3.1 Kaikki Exobot ja SharkBot -pankkihaittaohjelmien käyttämät tekniikat, tekniikoiden kuvaukset ja tieto siitä, onko tekniikan käyttö havaittavissa. 27

3.2 Android-laitteen resurssit ja niihin kohdistuvat pankkihaittaohjelmien käyttämät MITRE-tekniikat. Taulukossa listattu ainoastaan havaittavissa olevat tekniikat, jotka kohdistuvat johonkin resurssiin. 28

4.1 MITRE-tekniikat ja data, jonka avulla voidaan havaita tekniikan käyttö Android-laitteessa. 42

4.2 Android-sovellusrajapintoja, joiden kautta voidaan kerätä tietoturvaan liittyvää dataa. 44

5.1 Laitteessa B käytettävien sovellusten versionumerot. 46

5.2 Kokeessa käytettävän haittaohjelman toimintojen kuvaukset ja toimintoihin liitetyt MITRE-tekniikat. 47

5.3 Laitteeseen A asennettujen käytettävyyssovellusten package-nimet. . . 53

5.4 Laitteeseen A asennettujen, notifikaatioita lukevien sovellusten package-nimet. 53

1 Johdanto

Älypuhelimet ovat nykyään tietotyössä tietokoneen ohella yleinen työväline monissa organisaatioissa. Tietokoneen tavoin mobiililaiteet voivat kuitenkin sisältää arkaluontoista tietoa sekä organisaatioista, että laittetta käyttävävistä työntekijöistä.

Pienikokoiset mobiililaitteet ovat alttiita varkauksille ja fyysiselle peukaloinnille. Lisäksi ne ovat usein kiinni vieraissa verkoissa, jolloin verkkoliikenteen tarkkailu ja verkon yli tapahtuvat hyökkäykset ovat mahdollisia [1]. Organisaatioiden tapauksessa mobiililaitteet kulkevat monesti työntekijöiden mukana valvottujen tilojen ulkopuolla. Monissa organisaatioissa on myös käytössä BYOD (Bring Your Own Device) -menettelytapa, jossa työntekijät voivat käyttää omia laitteitaan työasioiden hoitamiseen. Näistä syistä mobiililaitteiden tietoturvatilanteen arvioiminen voi olla organisaatioille varsin hankalaa.

Tässä tutkielmassa tavoiteena on tutkia, miten organisaatio voisi havaita sille kuuluvien Android-laitteiden vaarantumista data-analytiikan avulla. Tarkoituksena on arvioida, millaista tietoturvaan liittyvää dataa Android-laitteista voitaisiin kerätä laitteisiin asennetun sovelluksen avulla.

Aluksi tutkielmassa kartoitetaan Android-laitteiden tietoturvaominaisuuksia sekä niitä laitteen resursseja, joista hyökkääjä voisi olla kiinnostunut. Tämän jälkeen käsitellään olennaisia Android-laitteisiin kohdistuvia uhkia. Esimerkkinä perehdytään erityisesti pankkipalveluihin liittyviin haittaohjelmiin, jotka ovat yleistyneet viime vuosien aikana. Hyökkäyksien tunnistamiseen sovelletaan MITRE ATT&CK

-tietokantaa, joka on tunnettu uhkien mallintamiseen käytetty apuväline. Näin saadaan määriteltyä data, jota laitteesta tulisi kerätä ja selvittää, voiko sitä kerätä laitteeseen asennetun sovelluksen avulla käyttäen Android-käyttöjärjestelmän tarjoamia rajapintoja. Lopuksi tarkastellaan useasta laitteesta kerättyä dataa yhdessä ja arvioidaan, voiko datamassan seasta löytää helposti poikkeavuuksia ja siten helpottaa suuren laitemäärän valvontaa.

Tutkimuskysymykset:

TK1: Mitä resursseja organisaation työntekijöiden Android-laitteissa tulisi suojata hyökkäyksiltä?

TK2: Millaisiin uhkiin ja hyökkäyksiin tulisi kiinnittää eniten huomiota, kun tarkastelukohteena joukko organisaation työntekijöiden käyttämiä Android-laitteita?

TK3: Millaista tietoturvaan liittyvää dataa Android-laitteista voidaan kerätä laitteeseen asennetun sovelluksen kautta?

TK4: Miten Android-laitteisiin kohdistuvia hyökkäyksiä voidaan havaita niistä kerätyn datan perusteella?

2 Tietoturva Android-laitteissa

Tassä luvussa käsitellään Android-laitteiden tietoturvaa yleisellä tasolla. Pohjana käytetään kahta jukaisua, joista molemmat on päivätty vuodelle 2023 ja ottavat huomioon Android-käyttöjärjestelmän ominaisuudet versioon 14 asti. Mayrhofer ym. kuvasivat julkaisussaan (The Android Platform Security Model) Android-laitteiden tietoturvamallia, sen suunnitteluperiaatteita sekä uhkakuvia [1]. Googlen julkaisu (Android Security Paper 2023) puolestaan kuvaa kattavasti Android-laitteiden tietoturvaominaisuuksia yleisellä tasolla [2].

Android-käyttöjärjestelmän tietoturvamalli on kehitetty useiden erityyppisiä uhkakuvia ajatellen. Hyökkääjä voi päästä laitteeseen käsiksi fyysisesti tai tarkkailla verkkoyhteyden yli kulkevaa liikennettä. Haitalliset sovellukset voivat myös hyväksikäyttää käyttöjärjestelmän tarjoamia ohjelmistorajapintoja. Uhkien torjumiseksi Android-käyttöjärjestelmään on kehitetty useita tietoturvaominaisuuksia sekä laittesto- että ohjelmistotasolla [1]. Näitä on käsitelty tarkemmin seuraavissa aliluvuissa.

2.1 Laitteistotason tietoturva

Android-laitteet tarjoavat *luotetun suoritusympäristön* (Trusted Excecution Environment, TEE) tietoturvan kannalta kriittisiin toimenpiteisiin, kuten näytön lukituksen avaamiseen ja kryptografisten avainten käsittelyyn. TEE on erillinen pieni käyttöjärjestelmä, jota ajetaan normaalin Android-käyttöjärjestelmän rinnalla sa-

malla prosessorilla. TEE on kuitenkin eristetty Linux kernelistä sekä sovelluksista. Normaalin käyttöjärjestelmän pääsyä on rajattu sellaisten muistin osien kohdalla, joita käytetään salaisen tiedon, kuten avainten säilyttämiseen. Normaalin käyttöjärjestelmän sijasta salaisen tiedon käsittely ohjataan TEE:lle. Eristys suojaa luotettua suoritusympäristöä haitallisilta sovelluksilta sekä Android käyttöjärjestelmästä löytyviltä haavoittuvuuksilta. [1] [2] [3].

Android-käyttöjärjestelmän koskemattomuus tarkistetaan laitteen käynnistysvaiheessa. *Todennettu käynnistys* (verified boot) on ollut poikkeuksetta pakollinen vaatimus laitteille Android 9.0 versiosta alkaen. Sen avulla voidaan havaita mahdolliset muutokset järjestelmässä ja tarkistaa, että kaikki suoritettava järjestelmän koodi on luotettavasta lähteestä [1].

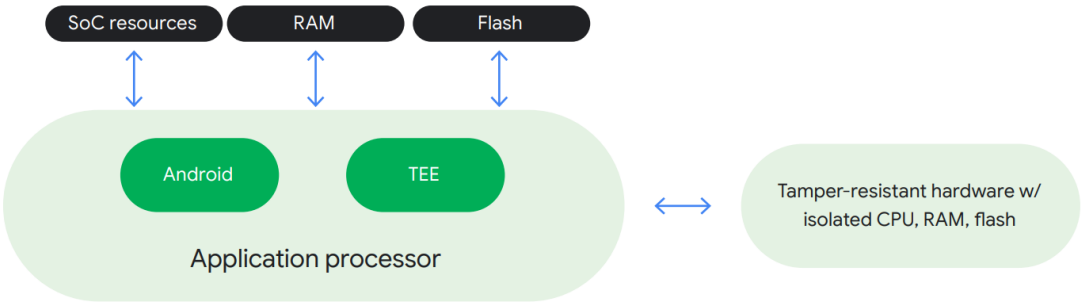
Tarkistus tehdään erikseen kaikissa käynnistysvaiheissa alkulatausohjelmasta (bootloader) eri partitioihin. Laitevalmistaja allekirjoittaa jokaisen laitteisiin päivitetyn Android-version omalla kryptografisella avaimellaan (root of trust), jonka alkulatausohjelma tarkistaa ennen Androidin käynnistämistä. Julkinen avain on tallennettu laitteeseen niin, ettei sitä voi muokata. Yksityinen avain puolestaan on vain laitevalmistajan tiedossa. Virheen tapahtuessa Androidin käynnistäminen voidaan estää kokonaan, mikäli alkulatausohjelma on lukitussa tilassa [4].

Todennettu käynnistys estää hyökkääjää tekemästä pysyviä järjestelmätason muutoksia laitteeseen, vaikka Android-käyttöjärjestelmä olisi laitteen käynnissä ollessa vaarantunut. Tämän lisäksi käyttöjärjestelmän voi päivittää ainoastaan uudempaan versioon (rollback protection). Hyökkäjä ei siis voi asentaa vanhaa versiota uuden tilalle ja siten hyödyntää vanhassa versiossa esiintyviä haavoittuvuuksia, vaikka molemmat versiot olisivat aidosta lähteestä [2].

2.2 Keystore ja avainten käsittely

Android Keystore mahdollistaa kryptografisten avainten turvallisen säilyttämisen laitteella. Keystore rajoittaa avaimien käyttöä mm. vaatimalla käyttäjää tunnistautumaan avainten käytön yhteydessä. Keystore yleensä tallentaa avaimet suojatuun suoritusympäristöön.

Android 9 versiosta alkaen Android tukee myös secure-elementin käyttöä avainten säilytykseen. Secure-element on laitekomponentti, joka sisältää oman prosessorin, muistin ja mekanismeja fyysisen peukaloimisen estämiseksi. Secure-element on siten vielä tehokkaammin eristetty muusta järjestelmästä kuin suojattu suoritusympäristö, joka käyttää samaa fyysistä prosessoria Android-käyttöjärjestelmän kanssa. [5]. (Kuva 2.1) havainnollistaa secure-elementin roolia. Kuvassa Android-käyttöjärjestelmä sekä luotettu suoritusympäristö (TEE) käyttävät samoja fyysisiä komponentteja (System on chip, RAM, Flash), kun taas secure-element käyttää omia, peukaloinnilta suojattuja komponentteja.



Kuva 2.1: Secure-element (oikealla) on eristetty Android-käyttöjärjestelmästä sekä suojatusta suoritusympäristöstä (TEE).

2.3 Käyttöjärjestelmä

2.3.1 Prosessien ja sovellusten eristäminen

Android-käyttöjärjestelmän tietoturvamalli on suunniteltu käyttäen pohjana Linux kernelin ominaisuuksia. Linux tarjoaa tuen useammalle käyttäjälle, jota Androidissa käytetään sovellusten eristämiseen toisistaan [2]. Jokaiselle sovellukselle annetaan oma tunniste (UID) ja ne ajetaan omassa prosessissaan, jolloin ne on kernel-tasolla eritelty toisistaan. Lähtökohtaisesti sovellukset eivät pysty kommunikoimaan toistensa kanssa ilman käyttöjärjestelmän lupaa ja niillä on rajattu pääsy käyttöjärjestelmään. Jos sovellus yrittää esimerkiksi lukea toisen sovelluksen tietoja tai soittaa puhelun ilman myönnettyä lupaa, järjestelmä estää kyseisen toiminnon [6]. Vastaavasti myös järjestelmän prosessit, kuten Wi-Fi -palvelut, Bluetooth-komponentit ja puhelinpalvelut on eristetty toisistaan. Nämä eristykset rajoittavat yksittäisen haavoittuvuuden vaikutuksia, jolloin hyökkääjä joutuu hyväksikäyttämään useampia järjestelmän haavoittuvuuksia saatuttaakseen tavoiteensa [1].

2.3.2 SELinux

SELinux on Linux-käyttöjärjestelmälle kehitetty kernel-moduuli, joka tarjoaa turvallisemman tavan järjestelmän resurssien käyttöoikeuksen määrittämiseen ja se on nykyään myös käytössä Android-käyttöjärjestelmässä [7].

Tavanomainen Linux hyödyntää *harkinnanvaraista oikeushallintaa* (discretionary access control, DAC), jossa kullakin resurssilla (tiedostot, portit) on omistaja, joka valvoo resurssin käyttöä. SELinux:issa on sen sijaan käytössä *pakollinen oikeushallinta* (Mandatory Access Control, MAC), jossa käyttöoikeuksia valvotaan keskitetysti ja käyttöoikeuksien eskaloiminen on siten vaikeampaa. Sallitut operaatiot määritellään *policy* -nimisinä sääntöinä, joiden perusteella päätetään, onko tietyn prosessin suorittama operaatio tietylle resurssille sallittu [7].

SELinux voi toimia kahdenlaisessa moodissa: *permissive* tai *enforcing*. Ensimmäisessä moodissa järjestelmä suorittaa policyn vastaiset operaatiot, mutta kirjaa ne ylös järjestelmän lokeihin. Enforcing moodissa sen sijaan järjestelmä estää policyn vastaiset operaatiot ja kirjaa ne lisäksi lokeihin. Enforcing moodi on tästä syystä huomattavasti turvallisempi. SELinux otettiin osaksi Android-käyttöjärjestelmää versiossa 4.3, jolloin käytössä oli ensin permissive-moodi. Versiosta 5.0 alkaen enforcing-moodi on kuitenkin ollut täysin käytössä kaikissa Android-versioissa [7].

2.3.3 Lupien myöntäminen sovelluksille

Android-sovellusten toimintaa rajoitetaan *Android-permissioilla*. Lähtökohtaisesti Android ei salli sovellusten ajaa sellaisia toimintoja, jotka vaikuttavat muiden sovellusten tai käyttöjärjestelmän toimintaan. Permissiolla suojatut arkaluontoiset tiedot ja järjestelmän ominaisuudet ovat saatavilla vain niille sovelluksille, joille tarvittavat permissiot on myönnetty. Permissiot on määritelty uniikkeina merkkijonoina, esim. *android.permission.READ_SMS*. Jokainen Android-sovellus sisältää *AndroidManifest.xml* -tiedoston, johon on listattu kaikki sovelluksen tarvitsemat permissiot [8].

Permissiot luokitellaan viiteen eri ryhmään sen mukaan, miten tärkeitä toimintoja ne suojaavat ja miten permissio myönnetään [1]:

1. **Normal permission** - resurssiin pääsy ei vaaranna käyttäjän yksityisyyttä, permissio myönnetään automaattisesti kun sovellus asennetaan laitteeseen.

2. **Dangerous permission** - myöntäminen edellyttää käyttäjän vahvistusta, joka voidaan kysyä sovelluksen käyttöliittymän kautta.

3. **Special pemission** - edellyttää käyttäjän vahvistusta, mutta toiminto jota permissio suojaa on alttiimpi väärinkäytölle eikä sitä voida suoraan kysyä sovelluksen käyttöliittymältä. Käyttäjän on myönnettävä permissio laitteen

asetuksissa. Näillä permissioilla käyttäjä voi esimerkiksi myöntää sovellukselle järjestelmänvalvojan oikeudet, tai antaa sovellukselle luvan muiden sovellusten asentamiseen.

4. **Priviledged permission** - permissio voidaan myöntää ainoastaan laitteen valmistajan määrittelemille sovelluksille.

5. **Signature permission** - permissio voidaan myöntää ainoastaan komponenteille, jotka on allekirjoitettu samoilla avaimilla kuin luvan myöntävä komponentti.

Näistä ryhmistä *Normal*, *Privileged* sekä *Signature* -tason permissiot myönnetään sovellukselle, kun se asennetaan laitteeseen. *Dangerous* ja *Special* -permissiot sen sijaan myönnetään sovellukselle vasta, kun käyttäjältä on pyydetty vahvistus niiden käytöstä [1].

2.4 Tunnistautuminen

Käyttäjän tunnistaminen tapahtuu Android-laitteilla pääosin näytön lukitusmekanismin kautta. Lukitus rakentuu kaksijakoisen mallin ympärille. Näytön lukituksen ollessa auki kaikki laitteen toiminnot ovat käytettävissä, kun taas lukitussa tilassa suurin osa laitteen toiminnoista ei ole lainkaan käytössä. Tunnistautumistapoja on usealla eri tasolla. Pääkeinona toimii salasana, PIN-koodi tai kuvio, jonka käyttäjä piirtää laitteen näytölle. Nämä menetelmät on parhaiten turvattuja, sillä avaaminen edellyttää käyttäjän tiedossa olevaa tunnusta (lukuunottamatta tilanteita, joissa näytölle jääneitä tahroja tarkastellaan kuvion tai PIN-koodin arvaamiseksi). Vahvan suojauksen vuoksi lukitus voidaan avata vain näillä menetelmillä sen jälkeen, kun laite on käynnistetty uudelleen. Käyttäjän antama tunnus verifioidaan luotettussa suoritusympäristössä (TEE) paremman turvan takaamiseksi.

Vaihtoehtoisena tunnistautumistapoina toimii biometrinen tunnistautuminen, esimerkiksi sormenjäljen tai kasvojentunnistuksen avulla. Nämä menetelmät ovat käyttäjän kannalta helpompia, mutta vähemmän turvallisia. Biometristä tunnistautumista on mahdollista huijata käyttämällä esimerkiksi tarpeeksi oikean näköistä sormenjälkeä, vaikkei se olisi oikea. Vastaavasti oikea sormenjälki ei ole aina 100-prosenttisella tarkkuudella tunnistettavissa. Laitetta ei voi avata uudelleen käynnistämisen jälkeen pelkällä biometrisellä tunnistautumisella. Tämän tapauksen lisäksi käyttäjän tulee myös avata laite vähintään kolmen vuorokauden välein päätunnuksella. Kolmannella tasolla tunnistautumiseen toimii luotettu Bluetooth-laite tai sijainti, jolloin käyttäjän ei tarvitse syöttää laitteeseen päätunnusta esimerkiksi ollessaan kotona. Tämän menetelmän osalta kuitenkin käyttäjää vaaditaan syöttämään salasana jo neljän tunnin välein [1].

2.5 Datan suojaus

Käyttäjän tiedot on salattu Android-laitteissa kryptografisilla avaimilla. Hyökkääjä ei siis kykene lukemaan tietoja suoraan laitteen muistista, ennen kuin näytön lukitus on avattu. Android 5.0 -versiossa laitteen koko user-partitio salattiin, mutta tämä esti mm. puhelinsoitot sekä hälytykset ennen salasanan syöttämistä. Android 7.0 -versiossa siirryttiin tiedostopohjaiseen salaukseen, jonka myötä puhelut ja hälytykset toimivat myös ennen salasanan syöttöä. Salauksessa käytetään hyväksi suojattua suoritusympäristöä sekä secure-elementtiä paremman turvallisuuden takaamiseksi. Android 10 -versiossa parannettiin salauksen suorituskykyä, jotta myös halvemman hintaluokan prosessorit kykenevät salaukseen ilman ongelmia. Tämän muutoksen myötä salaus on ollut pakollista kaikissa Android-laitteissa versiosta 10 alkaen [1].

2.6 Päivitykset

Käyttöjärjestelmän säännöllinen päivittäminen on olennainen osa myös Androidin tietoturvamallia. Google julkaisee kuukausittain tietoturvapäivityksiä, joissa on paikattu raportoituja Android-käyttöjärjestelmän haavoittuvuuksia ja korjattu vikoja. Laitevalmistajat eivät tosin välttämättä jaa päivityksiä kaikkiin laitteisiin yhtä nopealla aikataululla [1].

Android 10.0 -versiosta alkaen joitakin käyttöjärjestelmän komponentteja voidaan kuitenkin päivittää itsenäisesti ilman, että koko käyttöjärjestelmä asennetaan uusiksi. Tämä tapahtuu Google Play -palvelun kautta samaan tapaan kuin Android-sovellusten asentaminen, jolloin järjestelmän komponenttien päivitys tapahtuu samaan tapaan kuin sovellusten päivittäminen Google Play -kaupan kautta. Näin kriittisiä päivityksiä on mahdollista jakaa käyttäjille tavallista nopeammin [1].

3 Hyökkäykset ja uhat

3.1 Android-laitteiden resurssit

Ennen uhkien käsittelyä määritellään Android-laitteiden resurssit, joita halutaan suojata väärinkäytöltä. Tämän tutkielman osalta keskitytään ainoastaan Android-laitteisiin. Organisaatioiden palvelimet- ja verkkoympäristöt jätetään tarkastelun ulkopuolelle.

Android-laitteisiin kohdistuvia uhkia voidaan käsitellä monesta näkökulmasta. Bhat ja Dutta käsittelivät tutkimuksesssaan *A Survey on Various Threats and Current State of Security in Android Platform* uhkia kattavasti kernelin, sovellusten sekä laitekomponenttien näkökulmasta. Tutkimuksen mukaan käyttäjää on mahdollista vakoilla puhelimen kameran, mikrofonin sekä muiden sensorien avulla. Käyttäjän sijainti voi myös paljastua GPS:n välityksellä [9].

Laitteen prosessori, RAM-muisti, kiintolevy sekä akku voidaan laskea suojattaviksi resursseiksi myös saatavuuden osalta. Haitalliset sovellukset voivat aiheuttaa tavanomaista enemmän rasitusta prosessorille, käyttää enemmän muistia sekä kuluttaa enemmän akkua. Lopulta järjestelmä voi hidastua niin paljon, ettei käyttäjä enää kykene käyttämään laitteen toimintoja ollenkaan [9].

Alkulatausohjelma ja kappaleessa 2 kuvattu *root of trust* kuuluvat myös suojattavien resurssien piiriin. Alkulatausohjelman haavoittuvuuksia hyödyntämällä hyökkääjän on mahdollista päästä käsiksi suojattuun suoritusympäristöön.

Resurssien kartoittamiseksi tarvitaan kuitenkin tarkempi lista laitteen toiminnoista ja niiden käytöstä. Tämän tutkielman osalta keskitytään siis sovellusten permissioihin, joiden avulla valvotaan sovellusten pääsyä laitteen resursseihin. Permissioita tarkastelemalla havaitaan suoraan, minkälaisia resursseja ne suojaavat ja kuinka tärkeäksi mikäkin resurssi on määritelty Android-käyttöjärjestelmää suunnitellessa.

3.1.1 Sovellusten permissiot

Android-sovellusten permissioita käsiteltiin aiemmin luvussa 2.1, jossa listattiin eri tyyppisiä permissioita ja miten niitä myönnetään. Näistä Dangerous -tason permissiot käsittelevät yleensä käyttäjän yksityisyyteen liittyviä tietoja sekä laitteen toimintoja [1]. Tästä syystä tämän tutkielman osalta keskitytään erityisesti näiden permissioiden suojaamiin resursseihin.

Dangerous -tason permissiot on jaettu ryhmiin (*Permission group*) sen mukaan, mihin laitteen toiminnallisuuteen ne liittyvät. Sovelluksen pyytäessä jonkin ryhmän permissiota käyttäjältä, voi se samalla kyselyllä pyytää myös muut ryhmän permissiot [8]. Käytännössä permissioryhmät siis kuvaavat, millaisia permissioita kuhunkin laitteen resurssiin on liitetty.

Android 14 -versiossa on käytössä seuraavat permissioryhmät [10]:

1. ACTIVITY_RECOGNITION
2. CALENDAR
3. CALL_LOG
4. CAMERA
5. CONTACTS
6. LOCATION

7. MICROPHONE
8. NEARBY_DEVICES
9. NOTIFICATIONS
10. PHONE
11. READ_MEDIA_AURAL
12. READ_MEDIA_VISUAL
13. SENSORS
14. SMS
15. STORAGE

Resurssien tunnistamisen lisäksi sovellusten käyttämät permissiot itsessään voivat olla hyödyllistä dataa, jota kannattaisi kerätä laitteista. Tämä havaittiin tutkimuksessa *On the Effectiveness of Application Permissions for Android Ransomware Detection (2020)*, jossa onnistuttiin tunnistamaan kiristyssovelluksia tehokkaasti pelkästään tarkastelemalla sovellusten käyttämiä permissioita [11].

Laitteella olevien tietojen lisäksi voimme sisällyttää resursseihin käyttäjätunnukset eri palveluihin ja sovelluksiin, vaikka ne eivät välttämättä ole fyysisesti tallennettuna itse laitteella. Nämä kirjautumistiedot on kuitenkin mahdollista kaapata esimerkiksi näppäimistösovelluksen kautta [12], joten niiden suojaamiseen tulisi kiinnittää huomiota. Samaan resurssiin voidaan laskea myös henkilötunnukset sekä luottokorttien tiedot, joita käyttäjä mahdollisesti syöttää verkkopalveluihin mobiililaitteen kautta.

3.1.2 Yhteenveto resursseista

Android-laitteen resurssit:

R1: Käyttäjän fyysiseen tunnistamiseen liittyvä tieto

R2: Käyttäjän kalenteri

R3: Puhelujen lokitiedot

R4: Kamera

R5: Käyttäjän kontaktit

R6: Sijaintitiedot

R7: Mikrofoni

R8: Bluetooth-toiminnallisuus

R9: Notifikaatiot

R10: Puheluihin liittyvä toiminnallisuus, puhelinnumeron ja verkkojen tiedot

R11: Audiotiedostot jaetulla kiintolevyllä

R12: Kuva- ja videotiedostot jaetulla kiintolevyllä

R13: Laitteen sensorit

R14: SMS-viestit

R15: Jaettu kiintolevy

R16: Käyttäjätunnukset ja muut henkilökohtaiset tiedot

3.2 Hyökkäysten mallintaminen

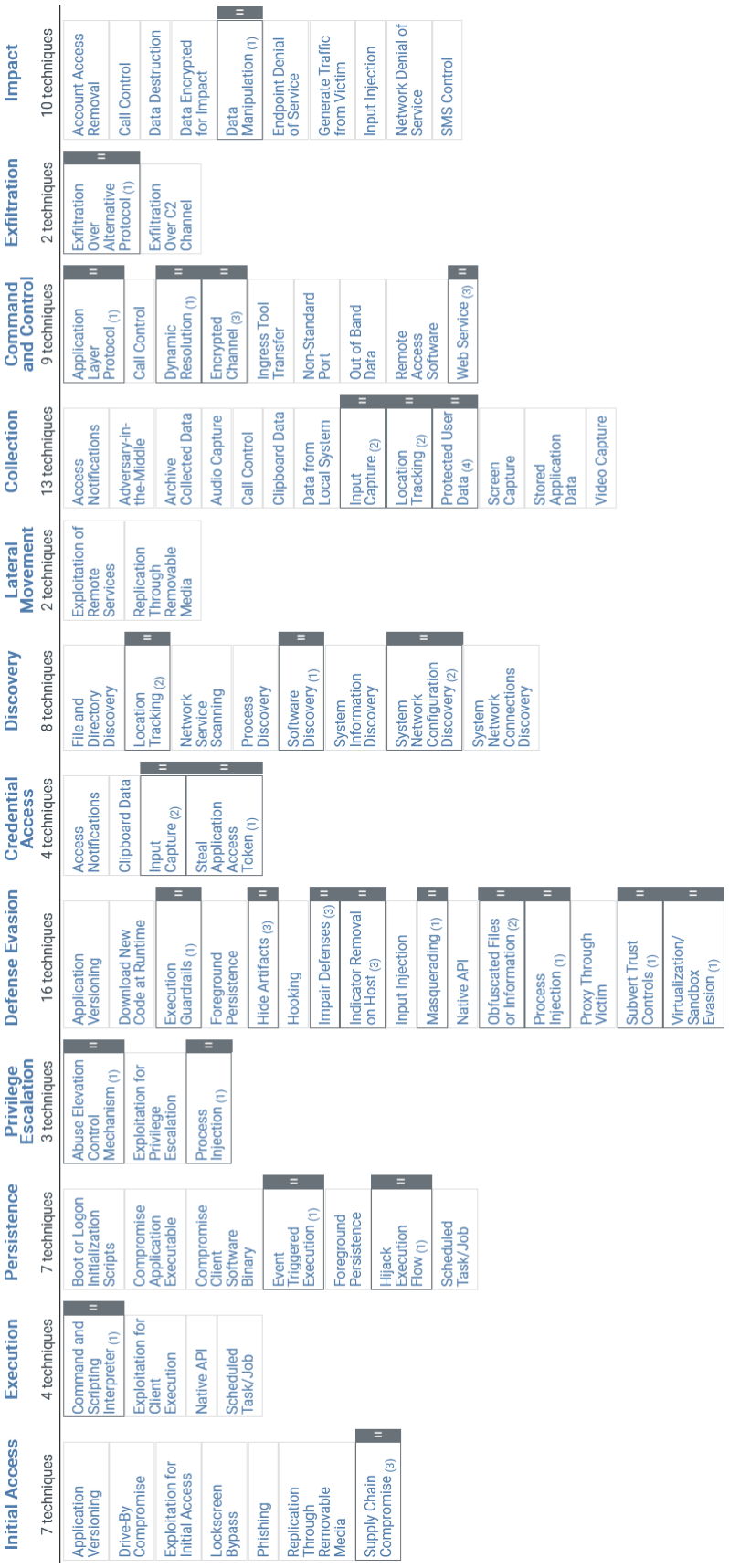
Tässä luvussa käsitellään tietoturvauhkien mallintamiseen käytettyjä tekniikoita. Tarkoituksena on valita menetelmä, jonka avulla voidaan tarkastella Android-laitteisiin kohdistuvia uhkia niin, että hyökkäysyritykset pystytään tunnistamaan.

3.2.1 MITRE ATT&CK

Yksi yleinen uhkien mallintamisessa käytetty apuväline on vuonna 2013 kehitetty MITRE ATT&CK (Adversial Tactics, Techniques and Common Knowledge). Kyseessä on tietokanta, johon on kerätty tietoa tosielämässä havaituista haitallisista aktiviteeteista. [13] MITRE ATT&CK -tietokanta esitetään usein matriisina, joka koostuu *taktiikoista* (Tactics) ja *tekniikoista* (Techniques). Taktiikat kuvaavat hyökkääjän lyhyen aikavälin tavoitteita, ja tekniikat puolestaan tapoja, joilla hyökkääjä voi saavuttaa kyseiset tavoitteet. Hyökkääjän toiminta pilkotaan taktiikoiden ja tekniikoiden avulla osiin, jolloin hyökkäyksen eteneminen on helpommin ymmärrettävissä. Tietokannasta löytyy kunkin tekniikan osalta kuvaus tekniikasta, sekä keinoja tekniikan käytön havainnointiin ja estämiseen. Tunnettujen haittohjelmien käyttämiä tekniikoita on myös listattu tietokantaan. [13]

MITRE ATT&CK -tietokanta on jaoteltu myös käytössä olevan teknologian mukaan useisiin eri matriiseihin, jotka sisältävät erilaisia tekniikoita ja taktiikoita toisiinsa nähden. Enterprise-matriisi sisältää Windows -ja Linux-laitteisiin sekä organisaatioiden tietoverkkoihin liittyviä taktiikoita ja tekniikoita. Vastaavasti teollisuuden ohjausjärjestelmille (ICS) ja mobiililaitteille on myös määritelty omat matriisit. [13] Oheisessa kuvassa (Kuva 3.1) on esimerkki matriisista, joka kuvaa Android-alustaan liittyviä tekniikoita ja taktiikoita.

Yksi tämän tutkielman tavoitteista on tunnistaa, minkälaista tietoturvaan liittyvää dataa Android-laitteista tulisi kerätä [TK3]. MITRE ATT&CK matriisia tarkastelemalla on mahdollista nähdä, mikälaista tietoa tarvitaan hyökkääjän käyttämien tekniikoiden havaitsemiseen. Mikäli tietyn tekniikan havainnointiin liittyvä tieto on mahdollista kerätä Android-sovellusrajapinnan kautta, voidaan se laskea tietoturvan kannalta olennaiseksi dataksi. Tätä dataa voidaan myöhemmin yhdistellä tunnistamaan hyökkäysten eri vaiheita. Tämän tutkielman osalta MITRE ATT&CK auttaa siis tunnistamaan niitä tietoja, joita Android-sovelluksen tulisi pystyä ke-



Kuva 3.1: MITRE ATT&CK Android-matriisi, jossa taktiikat on esitelty ylimällä rivillä. Kunkin taktiikan alle on listattu niihin liittyvät tekniikat.

räämään tietoturvatilanteen kartoittamiseen. Vertailun vuoksi käsitellään kuitenkin myös muita tarjolla olevia uhkien mallintamistapoja, joita ovat Cyber Kill Chain sekä STRIDE.

3.2.2 Cyber Kill Chain

Lockheed Martinin kehittämä Cyber Kill Chain -malli jakaa hyökkäykset seitsemään vaiheeseen, joita tarkastelemalla tietoturva-asiantuntijat voivat kehittää puolustuksia hyökkäysten pysäyttämiseen. Mallin mukaan hyökkäys voidaan pysäyttää, jos mikä tahansa sen seitsemästä vaiheesta saadaan pysäytettyä [14].

Hyökkäyksen vaiheet [14]:

1. Tiedustelu (Reconnaissance)
2. Aseistaminen (Weaponization)
3. Toimitus (Delivery)
4. Hyväksikäyttö (Exploitation)
5. Asentaminen (Installation)
6. Komento ja ohjaus (Command and control)
7. Tavoitteen saavuttaminen (Actions on objective)

Organisaatiot voivat käyttää Cyber Kill Chain -mallia tietoturva-aukkojen tunnistamiseen sekä tunnistamaan, miten suojamekanismit tulisi suunnitella jotta ne olisivat mahdollisimman tehokkaita. Mallissa ei kuitenkaan oteta huomioon iteratiivisia tai eri tekniikoita yhdistäviä hyökkäyksiä, jotka mukautuvat kohteena olevan ympäristön mukaan. Menetelmällä voidaan havaita hyökkäys tietyn vaiheen kohdalla, mutta hyökkäyksen etenemistä vaiheiden välillä ei ole otettu yhtä hyvin huomioon [14].

3.2.3 STRIDE

STRIDE on Microsoftin kehittämä malli, jota käytetään tietoturvauhkien tunnistamiseen. STRIDE -mallissa tarkastellaan systeemin eri komponentteja ja pyritään löytämään niistä heikkouksia, jotka voivat johtaa systeemin vaarantumiseen. Tarkastelussa otetaan huomioon kuusi yleistä tietoturvauhkaa, jotka on listattu alla. [15].

STRIDE -mallin tietoturvauhat [15]:

1. Laillisena käyttäjänä tai systeemin prosessina esiintyminen (Spoofing).
2. Tietojen luvaton muokkaaminen (Tampering).
3. Hyökkäyksen havaitsemisen estäminen ja jälkien peittäminen (Repudiation).
4. Luottamuksellisen tiedon luvaton jakaminen (Information disclosure).
5. Palvelunestohyökkäys (Denial of service).
6. Luvaton käyttöoikeuksien korottaminen (Elevation of privilege).

STRIDE on muihin tässä kappaleessa esitettyihin malleihin verrattuna vanhin, Mircrosoft otti sen käyttöön vuonna 2002 ja sitä on käytetty onnistuneesti erilaisten systeemien analysoimiseen. Mallin käyttö on kuitenkin työlästä ja löydettävien uhkien määrä kasvaa nopeasti sen mukaan, mitä monimutkaisempi systeemi on kyseessä [16].

3.3 Mobiililaitteisiin kohdistuvat uhat

Yksi tämän tutkielman tavoitteista on selvittää, millaisiin uhkakuviin tulisi kiinnittää eniten huomiota, kun tarkastelussa on organisaatioiden käytössä olevat Android-laitteet [TK2]. Tähän kysymykseen vastaamiseen käytetään lähteenä julkaisuja, joissa on analysoitu viime vuosina tapahtuneita hyökkäyksiä ja listattu niistä yleisimpiä.

Uhat pyritään valitsemaan sen mukaan, mitkä niistä ovat olleet viime aikoina eniten esillä ja koskevat erityisesti Android-laitteita.

ENISA on Euroopan Unionin alaisuudessa toimiva virasto, jonka tarkoituksena on edistää yleistä tietoturvan tasoa Euroopan alueella. Virasto julkaisee vuosittain tietoturvakatsauksen, joka sisältää tilastoja viimeisen vuoden aikana tapahtuneista tieturvaloukkauksista. Kyseinen raportti antaa yleiskuvan tämänhetkisistä tietoturvauhista Euroopan alueella [17].

Tietoturvaohjelmistoihin erikoistunut yritys Gen Digital julkaisee myös raportteja viime aikaisista uhkakuvista. Sen rapoteissa on myös mainittu useita mobiililaitteisiin kohdistuvia uhkia [18]. Kolmas vuosittaisia tietoturvakatsauksia julkaiseva taho on yritys nimeltä Zimperium, joka kehitää mobiililaitteisiin liittyviä tietoturvatuotteita. Zimperiumin julkaisuissa tarkastellaan erityisesti viimeisimpiä mobiililaitteisiin kohdistuvia uhkia, mihin tämän tutkielman osalta halutaan keskittyä [19] [20]. Vuoden 2024 raportissa listattiin yleisimpiä uhkia eri käyttöjärjestelmien (Android/iOS) sekä toimialojen mukaan. Näiden lisäksi nimettiin myös viisi uhkaa, joihin ogranisaatioiden tulisi erityisesti kiinnittää huomiota [20]. Zimperiumin raportin pohjalta mobiililaitteisiin sekä organisaatiohin kohdistuvia yleisimpiä uhkia on listattu alla:

Yleisiä Android-laitteisiin kohdistuvia uhkia: [20]:

1. Sovellusten asetaminen virallisen sovelluskaupan ulkopuolelta (sideloading apps).
2. Haavoittuvuuksia sisältävien Android-versioiden käyttäminen, joita ei voi päivittää.
3. Laitteelle ei ole asetettu pääsykoodia.
4. Haittaohjelmat.
5. Haitalliset verkkosivut.

6. Turvattomat verkot.

7. Mobiililaitteisiin kohdistuva tietojen kalastelu.

Uhkien suuren määrän vuoksi niitä voidaan käsitellä tämän tutkielman osalta vain rajallinen määrä. Tässä tutkielmassa keskitytään siis lähinnä tietyn tyyppisiin haittaohjelmiin. Edellä mainituissa raporteissa on nostettu esiin erityisesti pankkipalveluihin kohdistuvat haittaohjelmat, jotka ovat yleistyneet mobiililaitteissa vuosien 2022 ja 2023 aikana [17] [18] [19]. Nämä haittaojelmat ovat usein troijalaisia, jotka on esittävät olevansa pelejä tai muita haitattomia sovelluksia, mutta toimivat todellisuudessa haittaohjelman tavoin. Pankkipalvelujen tapauksessa haittaohjelma pyrkii yleensä saamaan haltuunsa käyttäjän pankkitunnukset tai siirtämään rahavaroja suoraan pankkisovelluksen kautta sen jälkeen, kun käyttäjä on ensin kirjautunut sisään [19]. Saavuttaakseen tavoitteensa, nämä haittaohjelmat käyttävät hyväksi puhelimen esteettomyys-ominaisuuksia, lukevat käyttäjän vastaanottamia SMS-viestejä sekä nauhoittavat puhelimen ruudun tapahtumia [18].

Raporteissa on mainittu nimeltä useita pankkihaittaohjelmia, mutta tämän tutkielman osalta keskitymme vain niihin, joiden käyttäytyminen on jo kartoitettu MITRE ATT&CK MOBILE -matriisiin. Näitä ovat mm. Cerberus [18] [19], Exobot (tunnetaan myös nimellä Octo) [18] [19], SharkBot [18] [19], TrickMo [18] sekä Flubot [19]. Exobot-pankkihaittaohjelma on lisätty MITRE -tietokantaan jo vuonna 2020 (ID S0522) [21]. Sen pohjalta on kuitenkin kehitetty uusia haittaohjelmia viime vuosien aikana, kuten Octo sekä Octo2, joita on kohdistettu eurooppalaisiin pankkipalveluihin [18]. SharkBot puolestaan havaittiin ensimmäisen kerran lokakuussa 2021 (ID S1055). Haittaohjelma pyrkii tekemään tilisiirtoja hyväksikäyttämällä Android-laitteen saavutettavuus-toimintoja [22]. Tämän tutkielman osalta käsittelemme ainoastaan Exobot- ja SharkBot-pankkihaittaohjelmia, koska muut mainitut pankkihaittaohjelmat käyttävät pitkälti samoja MITRE-tekniikoita näiden kahden kanssa.

MITRE ATT&CK -tietokanta listaa kunkin haittaojelman osalta niiden käyt-

tämät tekniikat. Kunkin tekniikan osalta puolestaan listataan myös keinot niiden havaitsemiseen, siltä osin kuin havaitseminen on mahdollista. Tarkastelemalla haittaohjelmien käyttämiä tekniikoita, saamme tietää, mitkä niistä on mahdollista havaita Android-laitteella ja mitä laitteen resursseja kohtaan ohjelmat yrittävät hyökätä. Tämän tutkielman osalta keskitymme vain niihin Android-laitteen resursseihin, jotka määriteltiin aiemmin kappaleessa 3.1.2. Tekniikoiden havaitseminen Android-sovellusrajapinnan kautta puolestaan käsitellään kappaleessa 4.

3.4 Pankkihaittaohjelmien käyttämät MITRE-tekniikat

Yksi pankkihaittaohjelmien käyttämistä MITRE-tekniikoista on T1626.001, jossa sovellus pyytää käyttäjältä järjestelmänvalvojan oikeuksia. Tämän jälkeen sovellus voi vaihtaa käyttäjän salasanan ja lukita tämän pois laitteelta, poistaa tietoja palauttamalla laitteen tehdasasetuksiin tai estää kameran käytön [12]. Tämä tekniikka kohdistuu siis useisiin laitteen resursseihin, riippuen mitä järjestelmänvalvojasovellus kussakin Android-versiossa kykenee tekemään. Edellä mainittujen toimintojen pohjalta voimme kuitenkin mainita vähintään resurssit R4, R11, R12 sekä R15.

T1437.001 puolestaan koskee web-protokollien käyttöä. Haittaohjelmaa voidaan käskyttää etänä HTTP-protokollan tai erityisten mobiiliviestipalveluiden, kuten Firebase Cloud Messaging -palvelun kautta. Tunnettujen protokollien käyttö tekee haitallisen verkkoliikenteen havaitsemisesta vaikeampaa, koska vastaavanlaista liikennettä näkyy laitteessa muutenkin [12]. Tämä tekniikka ei kuitenkaan suoraan kohdistu tässä tutkielmassa määriteltyihin laitteen resursseihin, eikä MITRE-tietokanta tarjoa keinoja sen havaitsemiseen.

Loppukäyttäjän palvelunesto T1642 rajoittaa tai estää käyttäjää tekemästä tiettyjä toimintoja laitteella. Tämä voidaan tehdä esimerkiksi hyödyntämällä järjestelmänvalvojan oikeuksia [12]. Resurssien osalta tähän voidaan laskea laitteella oleva data (R11, R12, R15), johon pääsy on estetty mikäli käyttäjä lukitaan pois laitteelta.

Tekniikkaa T1624.001 käytetään Android-sovellusten ja systeemikomponenttien välisten viestien (Intent) kuunteluun. Sovellukset voivat esimerkiksi rekisteröidä itsensä vastaanottamaan viestejä systeemiltä silloin, kun verkkoyhteys on saatavilla ja aloitta verkkoyhteyttä vaativat toiminnot vasta silloin. Kyseessä on tavallinen Androidin ominaisuus, mutta myös monet haittaohjelmat kuuntelevat näitä viestejä, jotta ne voivat aloittaa tietyt toiminnot esimerkiksi laitteen käynnistyttyä [12]. Tekniikka ei kuitenkaan suoraan kohdistu laitteen resursseihin, vaikka se onkin joltain osin havaittavissa.

T1417.001 puolestaan kuvaa käyttäjän näppäilytietojen tallentamista ja sen avulla voidaan päästä käsiksi esimerkiksi käyttäjätunnuksiin. Haittaohjelma voi esiintyä kolmannen osapuolen näppäimistösovelluksena, jonka kautta käyttäjä syöttää tekstiä laitteeseen, mutta tallentaa taustalla käyttäjän syöttämät tiedot muuta käyttöä varten. Saavutettavuus-toimintojen hyväksikäytöllä voidaan myös tarkkailla käyttäjän näytölle kirjoittamia tietoja. Toinen samantapainen tekniikka on T1417.002, jolla pyritään myös kaappaamaan käyttäjän tietoja. Tämä tekniikka kuitenkin käyttää hyväksi erillisiä UI-komponentteja. Haittaohjelma voi näyttää toisen sovelluksen käyttöliittymän päällä omia tekstikenttiä, joihin käyttäjä erehdyksessä syöttää tietonsa. Haittaohjelma voi myös näyttää notifikaatioita esittäen niiden liittyvän johonkin toiseen sovellukseen [12]. Molemmat tekniikat T1417.001 sekä T1417.002 kohdistuvat resurssiin R16 ja ovat havaittavissa.

Tekniikan T1655.001 avulla haittalliset sovellukset ja tiedostot pyritään nimeämään luotettavien sovellusten tai tiedostojen mukaan. Tavoitteena on ohittaa suojamekanismeja sekä naamioida sovelluksen toiminta. Haitalliset tiedostot voivat sijaita luotettavassa hakemistossa ja haittaohjelmien ikonit voivat olla kopioitu muista sovelluksista [12]. Tekniikka on joltain osin havaittavissa, ja liittyy osittain resurssiin R15, mikäli hyökkääjä käyttää jaettua kiintolevyä tiedostojen tallentamiseen.

Tekniikka T1636.003 mahdollistaa käyttäjän kontaktien keräämisen. Tämä voi-

daan tehdä Android-sovellusrajapinnan kautta ja rootatun laitteen tapauksessa se voidaan tehdä jopa käyttäjän huomaamatta. Vastaavasti tekniikka T1636.004 kohdistuu käyttäjän SMS-viestien lukemiseen. [12]. MITRE-tietokanta sisältää keinoja molempien tekniikoiden havaitsemiseen. Tekniikat kohdistuvat resursseihin R5 ja R14.

T1604 viittaa välityspalvelimen käyttöön. Hyökkääjä voi käyttää Android-laitetta välityspalvelimena, jolloin hyökkääjän verkkoliikenne näyttää tulevan uhriksi joutuneen käyttäjän Android-laitteesta. Näin hyökkäjä voi ohittaa palvelujen käyttämiä verkkoliikenteen rajoituksia [12]. Tämä tekniikka ei kuitenkaan kohdistu aiemmin määriteltyihin resursseihin.

SMS-viesteihin liittyvä tekniikka T1582 viittaa viestien poistamiseen tai lähettämiseen ilman käyttäjän lupaa. Tekniikkaa voidaan käyttää esimerkiksi haittaohjelmien levittämiseen [12]. Tekniikka on havaittavissa ja kohdistuu resurssiin R14.

Tekniikan T1418.001 avulla puolestaan voidaan havaita laitteeseen asennetut tietoturvaohjelmistot ja jäättää tiettyjä toimintoja tekemättä tarpeen mukaan [12]. Tekniikka ei kohdistu määrittelemiimme resursseihin.

T1426 mahdollistaa laitekohtaisten tietojen keräämisen. Näitä tietoja ovat esimerkiksi käyttöjärjestelmän versio, laitteen valmistaja sekä malli, operaattorin nimi ja laitteen maa-asetukset. Hyökkääjä voi käyttää tietoja muiden toimintojen suunnitteluun, yrittäen esimerkiksi hyväksikäyttää vanhan käyttöjärjestelmän haavoittuvuuksia [12]. Osa näistä tiedoista voidaan sisällyttää resurssiin R10, mutta MITRE-tietokannan mukaan tekniikan havaitseminen on haastavaa ja suosittelee sen sijaan keskittymään muiden tekniikoiden jäljittämiseen.

T1422-tekniikkaa käytetään laitteen verkkoasetusten tutkimiseen. Tähän sisältyy mm. IP- ja MAC -osoitteet, Wi-Fi verkon tiedot, SIM-kortin tiedot sekä laitten IMEI-numero. Alitekniikka T1422.001 puolestaan liitty verkkoyhteyden saatavuuden tarkistamiseen. Hyökkääjä voi käyttää näitä tekniikoita tiedusteluun ja seuraa-

vien toimintojen suunnittelemiseen, riippuen siitä millainen verkkoympäristö on kyseessä [12]. SIM-kortin tiedot voidaan sisällyttää R10 resurssin piiriin, joten T1422-tekniikka kohdistuu siihen. Tekniikan käyttö on MITRE-tietokannan mukaan havaittavissa.

T1517-tekniikan avulla voidaan lukea laitteen notifikaatioita. Notifikaatiot voivat sisältää arkaluontoista tietoa, kuten SMS-viestillä lähetettyjä kertakäyttöisiä salasanoja. Hyökkääjä voi myös piilottaa notifikaatioita käyttäjältä tai käyttää notifikaatioiden painikkeita [12]. Tekniikka kohdistuu resurssiin R9 ja sen käyttö on mahdollista havaita.

Tekniikka T1661 käyttää hyväksi sovelluksien päivittämistä. Alun perin haitaton sovellus on voitu julkaista Google Play -kaupassa, mutta sovelluksen uusi versio sisältää haitallista ohjelmakoodia ja sovelluksen toiminta muuttuu haitalliseksi vasta, kun sovellus päivitetään. Tekniikkaa voidaan käyttää myös varastettujen Google Play -tunnusten kanssa, jolloin hyökkääjä saa haittaohjelmalle valmiin joukon käyttäjiä [12]. Tekniikka ei kuitenkaan suoraan kohdistu aiemmin määrittelemiimme resursseihin.

T1407 puolestaan liittyy haitallisen ohjelmakoodin lataamiseen ohjelman suoritusaikana. Tällöin haitallista koodia ei löydy lainkaan sovellusta asennettaessa, joten sitä on hankala havaita staattiseen analyysiin pohjautuvilla menetelmillä ennen kuin sovellus käynnistetään [12]. Tämäkään tekniikka ei kohdistu aiemmin määrittelemiimme resursseihin.

T1637.001-tekniikalla generoidaan domain-nimiä tietyn algoritmin avulla. Tällöin haittasovelluksella voi olla tuhansia mahdollisia domain-nimiä, joita se voi käyttää hakeakseen hyökkääjän palvelimelta uusia käskyjä seuraavien toimenpiteiden tekemiseksi. Tämä puolestaan vaikeuttaa etähallintayhteyksien estämistä ja jäljittämistä [12]. Tekniikka ei kuitenkaan suoraan kohdistu mihinkään resurssiin.

Tekniikat T1521.001 ja T1521.002 liittyvät etähallintayhteyden salaamiseen. Hyök-

kääjä voi käyttää symmetrisiä (T1521.001) tai asymmetrisiä (T1521.002) salausalgoritmeja tietoliikenteen salaamiseen. Nämä tekniikat eivät kuitenkaan kohdistu määrittelemiimme resursseihin.

Toinen haittaohjelman ja palvelimen etähallintayhteyteen liittyvä tekniikka on T1646, jossa laitteesta varastettu data lähetetään eteenpäin palvelimelle [12]. Tämä tekniikka ei kuitenkaan suoraan koske tässä tutkielmassa esitettyjä resursseja, ja MITRE -tietokannan mukaan tämän tekniikan havaitseminen on muutenkin hankalaa.

Tekniikka T1630.001 liittyy haittaohjelman jälkien peittämiseen. Haittaohjelma voi sisältää toiminnallisuutta sovelluksen poistamiseksi, jolloin sen jäljittäminen vaikeutuu ja sen tekemä vahinko voi jäädä täysin huomaamatta [12]. Tekniikka ei kuitenkaan kohdistu mihinkään määrittellyistä resursseista, vaikka sen käyttö on jossain määrin havaittavissa.

T1544 viittaa ulkoisten työkalujen tai tiedostojen lataamiseen laitteelle. Hyökkääjä voi käyttää näitä tiedostoja apuna seuraavien toimenpiteiden tekemiseen [12]. Mutta mikäli tiedostoja ladataan sovelluksen omaan hakemistoon, tekniikka ei kohdistu määriteltyihin resursseihin.

T1516-tekniikan avulla hyökkääjä voi syöttää tietoa laitteen käyttöliittymälle. Tekniikan avulla haittaohjelma voi esimerkiksi myöntää itselleen permissioita, varastaa rahaa käyttäjän PayPal-tililtä tai estää muita sovelluksia avautumasta [12]. Koska tekniikalla voidaan myöntää lisää permissioita, se voi kohdistua moneen eri resurssiin, mm. R2, R4, R5, R7, R10 sekä R14. Tekniikka voidaan osittain havaita tarkastelemalla myönnettyjä käytettävyys-toimintojen permissioita.

T1406 viittaa obfuskoinnin käyttöön. Hyökkääjä voi salata ja pakata käyttämiään tiedostoja välttääkseen paljastumasta [12]. Tekniikka ei kuitenkaan suoraan kohdistu tämän tutkielman listassa määriteltyihin resursseihin.

T1644-tekniikan avulla hyökkääjä hyödyntää SMS-, Bluetooth- tai NFC-yhteyksiä

kommunikoidakseen kohteena olevan laitteen kanssa. Tällöin laite on hyökkääjän tavoitettavissa myös silloin, kun laite ei ole yhdistettynä internettiin eikä kommunikaatiota voida havaita pelkästään verkkoliikennettä tarkastelemalla. Sovellukset voivat esimerkiksi lukea hyökkääjän lähettämiä SMS-viestejä laitteen notifikaatioista [12]. Tekniikka voi kohdistua resursseihin R8, R9 tai R14. MITRE -tietokannan mukaan tekniikka voidaan havaita käyttöliittymän kautta tarkastelemalla notifikaatioita.

Tekniikka T1424 liittyy tiedusteluun. Hyökkääjä pyrkii selvittämään laitteessa käynnissä olevat prosessit saadakseen lisätietoa suoritusympäristöstä [12]. Tekniikka ei suoraan kohdistu määriteltyihin resursseihin.

Oheisessa taulukossa 3.1 on listattu pankkihaittaohjelmien käyttämien MITRE-tekniikoiden tunnukset, tekniikan kuvaus, tekniikkaa käyttävät haittaohjelmat sekä tieto siitä, tarjoaako MITRE-tietokanta keinoja tekniikan käytön havaitsemiseen.

Taulukossa 3.2 on puolestaan listattu aiemmin kappaleessa 3.1.2 määritellyt resurssit, sekä niihin kohdistuvat MITRE-tekniikat. Listassa ovat vain ne pankkihaittaohjelmien käyttämät tekniikat, jotka kohdistuvat johonkin resurssiin ja ovat jollain tapaa havaittavissa. Tämä taulukko vastaa osittain tutkimuskysymys TK2:een, koska se määrittelee, minkälaisiin uhkiin tulisi kiinnitää huomiota.

Tekniikka	Kuvaus	Haittaohjelmat	Havaittavissa (X)
T1626.001	Järjestelmänvalvojan oikeudet	Exobot	X
T1437.001	Web protokollan käyttö	Exobot, SharkBot	
T1642	Palvelunesto	Exobot	X
T1624.001	Systeemin viestien kuuntelu	Exobot	X
T1417.001	Näppäilyjen tarkkailu	Exobot, SharkBot	X
T1417.002	GUI syötteiden keräys	Exobot, SharkBot	X
T1655.001	Luotettava nimeäminen	Exobot	X
T1636.003	Kontaktien keräys	Exobot	X
T1636.004	SMS (lukeminen)	Exobot, SharkBot	X
T1604	Välityspalvelimen käyttö	Exobot	X
T1582	SMS (hallinta)	Exobot, Sharkbot	X
T1418.001	Tietoturvaohjelmistot	Exobot	X
T1426	Laitetietojen keräys	Exobot	
T1422	Verkkotietojen keräys	Exobot	X
T1422.001	Verkkoyhteyden saatavuus	Exobot	X
T1517	Notifikaatiot	SharkBot	X
T1661	Sovelluksen päivitys	SharkBot	X
T1407	Ohjelmakoodin lataus	SharkBot	X
T1637.001	Domain generointi	SharkBot	X
T1521.001	Symmetriset salausalgoritmit	SharkBot	
T1521.002	Epäsymmetriset salausalgoritmit	SharkBot	
T1646	Datan lähetys palvelimelle	SharkBot	
T1630.001	Sovelluksen poistaminen	SharkBot	X
T1544	Tiedostojen lataaminen	SharkBot	X
T1516	Tietojen syöttäminen UI:lle	SharkBot	X
T1406	Obfuskointi	SharkBot	X
T1644	Kommunikaatiokanavat	SharkBot	X
T1424	Käynnissä olevat prosessit	SharkBot	X

Taulukko 3.1: Kaikki Exobot ja SharkBot -pankkihaittaohjelmien käyttämät tekniikat, tekniikoiden kuvaukset ja tieto siitä, onko tekniikan käyttö havaittavissa.

Resurssi	Tekniikat
R2	T1516
R4	T1626.001, T1516
R5	T1636.003, T1516
R7	T1516
R8	T1644
R9	T1644, T1517
R10	T1422, T1422.001, T1516
R11	T1626.001, T1642
R12	T1626.001, T1642
R14	T1636.004, T1582, T1516, T1644
R15	T1626.001, T1642, T1655.001
R16	T1626.001, T1417.001, T1417.002

Taulukko 3.2: Android-laitteen resurssit ja niihin kohdistuvat pankkihaittaohjelmien käyttämät MITRE-tekniikat. Taulukossa listattu ainoastaan havaittavissa olevat tekniikat, jotka kohdistuvat johonkin resurssiin.

4 Hyökkäyksiin viittaavan datan kerääminen

Aiemmissa luvuissa tarkasteltiin Android-laitteen resursseja sekä niihin kohdistuvia uhkia. Tässä luvussa puolestaan selvitetään, millaista dataa Android-laitteesta tulisi kerätä, jotta voidaan havaita resursseihin kohdistuvia hyökkäyksiä.

Datan keräämistä tarkastellaan kahdesta näkökulmasta. Ensimmäiseksi selvitetään, voiko resursseihin pääsyä monitoroida julkisen Android-rajapinnan kautta. Mikäli tämä on mahdollista, voidaan kunkin resurssin osalta havaita, milloin niitä yritetään käyttää hyväksi.

Resursseihin pääsyn valvomisen lisäksi datan keräämistä voidaan tarkastella myös aiemmin määriteltyjen uhkien näkökulmasta. Luvussa 3.2.1 mainittiin, että MITRE-tietokanta sisältää tietoa siitä, miten kunkin MITRE-tekniikan käyttö voidaan tunnistaa Android-laitteessa. Osa tekniikoista voidaan siis mahdollisesti tunnistaa jo ennen kuin hyökkääjä on päässyt käsiksi haluamaansa resurssiin. MITRE-tekniikoiden käyttöön liittyvä data voi tästä syystä myös olla hyödyksi hyökkäysten tunnistamisessa.

4.1 Laitteen resurssien käytön tunnistaminen

Yksi tapa haitallisen toiminnan tunnistamiseen on tarkkailla laitteen resurssien käyttöä. Android-laitteen resurssit määriteltiin aiemmin kappaleessa 3.1.

AppOpsManager-luokka sisältää toiminnallisuutta runtime-permissioiden jäljittämiseen [23]. Dokumentaation mukaan tämän luokan kautta on mahdollista tarkkailla runtime-permissioiden suojaamiin resusseihin pääsyä. Tarkka jäljittäminen on dokumentaation mukaan kuitenkin mahdollista vain systeemi-komponenteille [23].

Tämän tutkielman osalta testattiin AppOpsManager-luokan julkisia metodeja, mutta niiden avulla ei ollut mahdollista suoraan havaita resurssien käyttöä. Metodien startWatchingActive() ja setOnOpNotedCallback() yhteydessä määritelty callback ei toiminut, kun kameraa tai mikrofonia käytettiin muiden sovellusten kautta. Metodin startWatchingMode() yhteydessä määritelty callback sen sijaan ilmoitti ainoastaan kameran tai mikrofonin permissioiden muutoksista.

Android ei siis ilmeisesti tarjoa suoraan keinoja resurssien käytön tarkkailuun julkisen rajapinnan kautta. Sovellusten näkyvyyttä on muutenkin rajoitettu Androidissa ja sovellukset tarvitsevat erityisiä permissioita jo pelkästään saadakseen listan muista laitteeseen asennetuista sovelluksista [24].

Resurssien suoran käytön sijaan voidaan kuitenkin yrittää tarkastella resursseja suojaavien permissioiden käyttöä. Tämä menetelmä voisi sopia ainakin osittain Android-sovellusten kautta tapahtuvan väärinkäytön havaitsemiseen.

Sovellusten Android-permissiot on tallennettu laitteessa keskitettyyn paikkaan, josta muut laitteen palvelut voivat tarkistaa sovelluksen oikeudet. Sovelluksen pyytäessä esimerkiksi laitteen sijaintitietoja, sijaintipalvelu tarkistaa ensin kutsuvan sovelluksen Android-permissiot ennen tietojen antamista sovellukselle [1]. Poikkeuksellisen laajat Android-permissiot voivat siis viitata siihen, että sovellus pyrkii hyväksikäyttämään laitteen resursseja.

4.2 Sovellusten käyttämien permissioiden keräämi-nen

Kappaleessa 3.1.1 mainittiin, että sovellusten permissioita keräämällä voidaan tunnistaa mahdollisia haittaohjelmia. MITRE-tietokanta ehdottaa myös monen tekniikan havaitsemiseen permissioiden tarkastelua. Tarkemmat tiedot kunkin tekniikan tunnistamiseen tarvittavista permissioista on kuvattu seuraavissa kappaleissa. Tässä kappaleessa sen sijaan käsitellään yleisellä tasolla, miten sovellusten käyttämiä Android-permissiota voidaan kerätä laitteesta.

Android-sovellusrajapinta tarjoaa PackageManager-luokan laitteeseen asennettujen sovellusten tarkastelemiseen [25]. Tämän luokan kautta voidaan kerätä kaikki permissiot, joita laitteeseen asennetut sovellukset pyytävät. Näiden permissioiden joukosta voidaan myös tunnistaa sovelluksille myönnetyt permissiot. Kaikkien laitteeseen asennettujen sovellusten tietojen kerääminen kuitenkin vaatii android.permission.QUERY_ALL_PACKAGES -permission käyttämistä, jota Google ei suosittele kuin tietyissä erityistapauksissa. Google Play -sovelluskaupan ehdoissa lista laitteeseen asennetuista sovelluksista luokitellaan käyttäjän henkilökohtaiseksi tiedoksi, joten Google ei julkaise QUERY_ALL_PACKAGES -permissiota käyttäviä sovelluksia, elleivät ne täytä tiettyjä vaatimuksia. Tämä rajoitus on hyvä pitää mielessä, jos halutaan kerätä permissioita sellaisen sovelluksen kautta, joka on tarkoitus julkaista Google Play -sovelluskaupassa [24].

Ohjelmalistaus 1 näyttää, miten kerätään kaikki laitteeseen asennettujen sovellusten pyytämät permissiot. Ohjelmalistaus myös näyttää, onko permissio myönnetty sovellukselle.

Ohjelmalistaus 1 Sovellusten käyttämien permissioiden kerääminen (java)

```
import android.Manifest;

import android.content.pm.PackageInfo;

import android.content.pm.PackageManager;

import java.util.List;


// Hae kaikki laitteeseen asennetut sovellukset, niiden pyytämät

// permissiot ja tarkista, onko permissio myönnetty.

PackageManager pm = getApplicationContext().getPackageManager();

List<PackageInfo> packages = pm.getInstalledPackages(

PackageManager.GET_PERMISSIONS);

for(PackageInfo packageInfo : packages) {

    String[] permissions = packageInfo.requestedPermissions;

    for(String p: permissions) {

        int granted = pm.checkPermission(p, packageInfo.packageName);

    }

}
```

4.3 MITRE-tekniikoden käytön tunnistaminen

4.3.1 Tietojen syöttäminen UI:lle - T1516

Tekniikan T1516 avulla hyökkääjä voi komentaa Android-laitteen käyttöliittymää käyttäen hyväksi käytettävyys-palveluita [12]. Käytettävyyspalvelut on alun perin tarkoitettu auttamaan vammaisia käyttäjiä vuorovaikuttamaan laitteen käyttöliittymän kanssa ja niistä on apua myös käyttäjille, jotka tilapäisesti eivät pysty käyttämään laitetta normaalisti, esimerkiksi autoa ajaessa tai hoitaessaan pientä lasta [26]. Haittaohjelmat voivat kuitenkin hyväksikäyttää ominaisuutta vääriin tarkoituksiin.

MITRE-tietokannan mukaan tekniikan käyttö on mahdollista havaita laitteen

käyttöliittymän kautta. Laitteen asetuksissa on listattu sovellukset, jotka on rekisteröity käyttämään käytettävyyspalveluita. Jokaisen sovelluksista voidaan erikseen aktivoida tai ottaa pois käytöstä. MITRE-tietokanta käyttää tunnuksia tekniikoiden lisäksi myös tunnistusmenetelmille. Tunniste DS0042 MITRE-tietokannassa viittaa tekniikoiden tunnistamiseen laitteen asetuksien avulla [12].

Tässä tutkielmassa tavoitteena on kuitenkin tunnistaa menetelmien käyttö Android-sovellusrajapinnan kautta. Tämä onnistuu käyttämällä AccessibilityManager-luokkaa, jonka kautta saadaan lista kaikista laitteeseen asennetuista käytettävyyspalveluita tarjoavista sovelluksista. Tämän lisäksi on mahdollista kerätä tietoja vain niistä käytettävyyspalveluseovelluksista, joiden käytettävyys-ominaisuudet on aktivoitu. Ohjelmalistaus 2 näyttää, miten kerätään laitteeseen asennettujen käytettävyyssovellusten nimet. Ohjelmalistaus näyttää myös erikseen, miten kerätään niiden käytettävyyssovellusten nimet, jotka on aktivoitu.

Ohjelmalistaus 2 Käytettävyyspalvelusovellusten tietojen kerääminen (java)

```
// Hae kaikkien käytettävyyspalvelusovellusten tiedot.
AccessibilityManager am = getApplicationContext().getSystemService(
AccessibilityManager.class);
List<AccessibilityServiceInfo> servicesInfos =
am.getInstalledAccessibilityServiceList();

// Hae aktiivisten käytettävyyspalvelusovellusten tiedot.
servicesInfos = am.getEnabledAccessibilityServiceList(
AccessibilityServiceInfo.FEEDBACK_ALL_MASK);
```

4.3.2 Järjestelmänvalvojan oikeudet - T1626.001

MITRE-tietokanta tarjoaa kaksi menetelmää järjestelmänvalvojasovellusten tunnistamiseen. Yksi menetelmistä on DS0041, jossa tarkastellaan laitteeseen asennettujen sovellusten permissioita. Järjestelmänvalvojasovelluksilta edellytetään android.permission.BIND_DEVICE_ADMIN -permissiota. Toinen keino tunnistaa järjestelmänvalvojasovelluksia on tarkastella laitteen käyttöliittymää (DS0042). Nämä sovellukset näkyvät laitteen asetuksissa ja niiden pyytäessä järjestelmänvalvojan oikeuksia, käyttäjälle esitetään pyyntö laitteen käyttöliittymällä [12].

Käyttöliittymän tarkkailu ei kuitenkaan ole tämän tutkielman osalta käytännöllinen ratkaisu. Voimme silti kerätä tietoa BIND_DEVICE_ADMIN -permissiota käyttävistä sovelluksista, jolloin tiedämme, mitkä sovellukset laitteessa on mahdollista asettaa järjestelmänvalvojaksi. DevicePolicyManager-luokan kautta on myös mahdollista kerätä lista aktiivisista järjestelmänvalvojasovelluksista [27]. Tämän menetelmän käyttöä ei ole rajoitettu permissioiden osalta, mutta sillä ei voida tunnistaa aktivoimattomia järjestelmänvalvojasovelluksia.

Ohjelmalistaus 3 näyttää, miten saadaan selville laitteessa olevat aktiiviset järjestelmänvalvojasovellukset. Tämän lisäksi voidaan tarkkailla BIND_DEVICE_ADMIN -permission käyttöä ja sitä kautta selvittää kaikki ne sovellukset, jotka on mahdollista asettaa järjestelmänvalvojaksi, mutta eivät välttämättä ole aktiivisia. Ohjelmalistaus 1 näyttää, miten permissiot kerätään laitteesta.

Ohjelmalistaus 3 Järjestelmänvalvojasovellusten tietojen kerääminen (java)

```
// Hae aktiiviset järjestlmänvalvojasovellukset.
DevicePolicyManager dpm = getApplicationContext().getSystemService(
DevicePolicyManager.class);
List<ComponentName> components = dpm.getActiveAdmins();
```

4.3.3 Kontaktien keräys - T1636.003

MITRE-tietokanta ehdottaa kahta tapaa kontaktien keräämisen tunnistamiseen [12]. Ensimmäinen näistä on permissioiden tarkastelu (DS0041). Kontaktitietoihin pääsy edellyttää sovellukselta android.permission.READ_CONTACTS -permissiota, jota useimmat sovellukset eivät tarvitse ja siitä syystä permissiota käyttäviin sovelluksiin tulisi suhtautua varauksin. Toinen tapa havaita tekniikan käyttö on tarkastella sovellusten permissioita käyttöliittymän kautta laitteen asetuksissa (DS0042) [12].

Permissioiden tarkastelu on tämän tutkielman tavoitteiden osalta käytännöllisempi ratkaisu. Ohjelmalistaus 1 näyttää, miten permissiot kerätään laitteesta. Tarkastelemalla READ_CONTACTS -permission käyttöä voidaan tunnistaa sovellukset, jotka mahdollisesti pyrkivät lukemaan käyttäjän kontakteja.

4.3.4 Notifikaatiot - T1517

Android-laitteen notifikaatioiden lukeminen ja hallitseminen on MITRE-tietokannan mukaan mahdollista tunnistaa kahdella tavalla. Yksi tapa on tarkastella android.permission.BIND_NOTIFICATION_LISTENER_SERVICE -permission käyttöä (DS0041). Toinen tapa on tarkastella laitteen asetuksia käyttöliittymän kautta (DS0042) [12].

Ohjelmalistaus 4 näyttää, miten tunnistetaan ne sovellukset, joille voidaan myöntää lupa notifikaatioiden lukemiseen. Ohjelma hakee sovellukset, jotka käyttävät Androidin NotificationListenerService -luokkaa sekä BIND_NOTIFICATION_LISTENER_SERVICE -permissiota, jotka vaaditaan notifikaatioiden lukemiseksi. Näille sovelluksille on mahdollista myöntää lupa notifikaatioiden lukemiseen laitteen asetuksissa. Ohjelma ei kuitenkaan kykene tarkistamaan, onko sovelluksille myönnetty lupa notifikaatioiden lukemiseen vai ei. Sovellukset kuitenkin kykenevät käyttämään T1517-tekniikkaa hyväksi, mikäli käyttäjä myöntää niille luvan laitteen asetuksissa.

Ohjelmalistaus 4 Notifikaatioita lukevien sovellusten tunnistaminen (java)

```
// Hae niiden sovellusten package-nimet, joille voidaan myöntää
// lupa notifikaatioiden lukemiseen.
PackageManager pm = getApplicationContext().getPackageManager();
List<String> packageNames = new ArrayList<>();
Intent intent = new Intent(
"android.service.notification.NotificationListenerService");
List<ResolveInfo> infos = pm.queryIntentServices(intent,
PackageManager.GET_META_DATA);
for (ResolveInfo info : infos) {
    if ("android.permission.BIND_NOTIFICATION_LISTENER_SERVICE".equals(
    resolveInfo.serviceInfo.permission)) {
        packageNames.add(resolveInfo.serviceInfo.packageName);
    }
}
```

4.3.5 Kommunikaatiokanavat - T1644

Hyökkääjä voi käyttää internet-yhteyden sijasta muita keinoja kommunikoidessaan hyökkäyksen kohteena olevan Android-laitteen kanssa. Näihin kuuluvat mm. SMS, Bluetooth ja NFC. Sovellukset voivat lukea näiden kanavien kautta lähetettyä tietoa laitteen notifikaatioista [12].

MITRE-tietokannan mukaan tekniikka voidaan havaita tarkkailemalla laitteen käyttöliittymää ja siinä näkyviä notifikaatioita (DS0042). Jos laitteella havaitaan tavallisesta poikkeavia notifikaatioita, tulisi asennetut sovellukset tarkistaa mahdollisten haittaohjelmien varalta [12].

Tämän havainnointimenetelmän osalta ei kuitenkaan määritellä tarkalleen, millaiset notifikaatiot viittaavat haittaohjelman toimintaan. Tekniikassa käytetään kui-

tenkin hyväksi notifikaatioiden lukemista, joka käsiteltiin jo aiemmin T1517-tekniikan yhteydessä. T1644-tekniikan osalta voidaan siis käyttää samaa tunnistusmenetelmää kuin T1517:n kohdalla, eli tarkistaa BIND_NOTIFICATION_LISTENER_SERVICE -permissiota käyttävät sovellukset.

4.3.6 Verkkotietojen keräys - T1422 ja T1422.001

Tekniikat T1422 ja alitekniikka T1422.001 liittyvät laitteen verkkotietojen tarkasteluun. Android 10 -versiosta alkaen laitteen yksilöiviin tunnisteisiin pääsyä on rajoitettu ja esimerkiksi laitteen IMEI:n lukeminen on mahdollista vain tietyin edellytyksin. MITRE-tietokanta ehdottaakin tekniikoiden tunnistamiseen sovellusten permissioiden tarkastelua (DS0041). READ_PRIVILEGED_PHONE_STATE -permission käyttäminen saattaa viitata siihen, että sovellus yrittää kerätä laitekohtaisia tietoja [12].

Ohjelmalistaus 1 näyttää, miten saadaan selville sovellusten käyttämät permissiot. Hakemalla näiden joukosta READ_PRIVILEGED_PHONE_STATE -permissiota käyttävät sovellukset, voidaan tunnistaa T1422-tekniikkaa käyttävät sovellukset.

4.3.7 Palvelunesto - T1642

Palvelunesto tarkoittaa tässä yhteydessä laitteen käytön estämistä. Tämä voidaan tehdä järjestelmänvalvojasovelluksen kautta vaihtamalla laitteen pääsykoodi ja lukitsemalla laite. Android 7 -versiosta alkaen mikä tahansa järjestelmänvalvojasovellus ei kuitenkaan pysty vaihtamaan laitteen pääsykoodia. Tekniikan käyttö voidaan havaita tarkastelemalla laitteen permissioita (DS0041), tai tarkastelemalla aktiivisia järjestelmänvalvojasovelluksia laitteen asetuksissa käyttöliittymän kautta (DS0042) [12].

Järjestelmänvalvojasovellusten tunnistaminen käsiteltiin jo aiemmin T1626.001-tekniikan yhteydessä. Samaa tunnistusmenetelmää voidaan käyttää myös palvelu-

neston tunnistamiseen, eli hakemalla laitteen aktiiviset järjestelmänvalvojasovellukset sekä BIND_DEVICE_ADMIN -permissiota käyttävät sovellukset.

4.3.8 SMS (lukeminen) - T1636.004

MITRE-tietokanta ehdottaa SMS-viestejä lukevien sovellusten tunnistamiseen kahta menetelmää. SMS-viestien lukeminen edellyttää android.permission.READ_SMS -permissiota, joten tämän permission käyttöä tarkkailemalla voidaan havaita tekniikan käyttö. Toinen MITRE-tietokannan ehdottama tapa tekniikan käytön tunnistamiseen on laitteen asetuksien tarkkailu käyttöliittymän kautta (DS0042).

Permissioiden tarkkailu on tämän tutkielman tavoitteiden kannalta parempi ratkaisu. Ohjelmalistaus 1 näyttää, miten sovellusten käyttämät permissiot kerätään laitteesta. T1636.001-tekniikan käyttö voidaan havaita hakemalla ne sovellukset, jotka käyttävät READ_SMS -permissiota.

4.3.9 SMS (hallinta) - T1582

SMS-viestien poistamiseen, lähettämiseen ja muuhun hallintaan liittyvät toimenpiteet on MITRE-tietokannan mukaan myös mahdollista tunnistaa permissioiden avulla. RECEIVE_SMS -sekä SEND_SMS -permissiot mahdollistavat sovelluksille viestien vastaanottamisen ja lähettämisen. Näiden lisäksi MITRE-tietokanta ehdottaa tarkastamaan, mikä sovellus on asetettu oletuksena käsittelemään SMS-viestejä. Tämä on mahdollista tarkistaa laitteen asetuksissa (DS0042). Oletettu SMS-sovellus (default SMS-handler) voi mahdollisesti poistaa, editoida tai lisätä viestejä laitteen viestitietokantaan [12].

Oletettu SMS-viestisovellus on mahdollista tarkistaa myös Android-sovellusrajapinnan kautta. Ohjelmalistaus 5 näyttää, miten sovelluksen package-nimi saadaan selville. Tässäkin tapauksessa oletetaan, että tietoa keräävällä sovelluksella on QUERY_ALL_PACKAGES -permissio. Kun SMS-viestisovelluksen package-nimi on sel-

vitetty, voidaan kerätä PackageManager-rajapinnan kautta lisätietoa sovelluksesta tarpeen mukaan.

Ohjelmalistaus 5 Oletukseksi asetetun SMS-viestisovelluksen tunnistaminen (java)

```
import android.provider.Telephony;
String packageName = Telephony.Sms.getDefaultSmsPackage(
getApplicationContext());
```

4.3.10 Luotettava nimeäminen - T1655.001

Haittaohjelmat voivat käyttää luotettavien sovellusten nimiä ja ikoneita (Netflix, WhatsApp), tai nimetä tiedostoja niin, että ne eivät herätä epäilyksiä haittaohjelman toiminnasta. MITRE-tietokannassa listatut tunnistusmenetelmät eivät kuitenkaan ole täysin yksiselitteisiä. Sovelluksen koodia ja metadataa voidaan arvioida haitallisen toiminnan tunnistamiseksi (DS0041). Laitteen käyttöliittymällä näkyvä tavallisesta poikkeava sovelluksen käyttäytyminen voi myös viitata haittaohjelman toimintaan (DS0042) [12]. Näitä ei kuitenkaan voida sitoa suoraan Android-rajapinnan kautta kerättävään dataan. Tämän tekniikan tunnistaminen edellyttäisi tarkempaa tutkimista jokaisen haittaohjelman kohdalla.

4.3.11 Näppäilyjen tarkkailu - T1417.001

Hyökkääjä voi tallentaa käyttäjän näppäilyjä käytettävyyspalveluiden, tai näppäimistösovelluksen kautta. MITRE-tietokannan mukaan käytettävyyssovellukset voidaan tunnistaa tarkistamalla käytettävyyssovellukset laitteen asetuksista, sekä tarkkailemalla BIND_ACCESSIBILITY_SERVICE -permission käyttöä (DS0041). Näppämäimistösovellukset löytyvät myös käyttöliittymän kautta laitteen asetuksista (DS0042) [12].

Ohjelmalistaus 2 näyttää, miten käytettävyyspalvelusovellukset voidaan tunnistaa laitteesta. Näppäimistösovellukset voidaan myös tunnistaa Android-rajapinnan kautta. Tämä on esitetty ohjelmalistauksessa 6. Ohjelma hakee ensin kaikki laitteeseen asennetut näppäimistösovellukset, eli sovellukset, joiden avulla käyttäjä voi syöttää tekstiä laitteen käyttöliittymälle. Tämän jälkeen ohjelma hakee näistä sovelluksista vain ne, jotka on aktivoitu laitteen asetuksissa. Lopuksi ohjelma hakee tällä hetkellä käytössä olevan näppäimistösovelluksen. Tavanomaisesta poikkeavat käytettävyys- tai näppäimistösovellukset voivat viitata siihen, että laitteessa on haittaohjelma.

Ohjelmalistaus 6 Laitteeseen asennettujen näppäimistöjen tunnistaminen (java)

```
InputMethodManager imm = (InputMethodManager) getApplicationContext()
.getSystemService(Context.INPUT_METHOD_SERVICE);
List<InputMethodInfo> inputMethods = imm.getInputMethodList();
for (InputMethodInfo inputMethod : inputMethods) {
    // Näppäimistösovelluksen package-nimi
    String packageName = inputMethod.getPackageName();
}
inputMethods = imm.getEnabledInputMethodList();
for (InputMethodInfo inputMethod : inputMethods) {
    // Aktivoidun näppäimistösovelluksen package-nimi
    String packageName = inputMethod.getPackageName();
}
InputMethodInfo currentInput = imm.getCurrentInputMethodInfo();
// Käytössä oleva näppäimistösovellus
String packageName = currentInput.getPackageName();
```

4.3.12 GUI syötteiden keräys - T1417.002

MITRE-tietokannan mukaan sovellus voi näyttää sisältöä laitteen käyttöliittymällä muiden sovellusten päällä, mikäli sille on myönnetty SYSTEM_ALERT_WINDOW -permissio [12]. Tätä permissiota käyttävät sovellukset voivat siis mahdollisesti olla haittaohjelmia, jotka koittavat varastaa esimerkiksi käyttäjän kirjautumistietoja. Ohjelmalistaus 1 näyttää, miten sovelluksille myönnetyt permissiot kerätään laitteesta.

4.4 Yhteenveto kerättävästä datasta

Yksi tämän tutkielman tavoitteista oli selvittää, millaisen datan avulla voidaan tunnistaa Android-laitteisiin kohdistuvia uhkia (TK3). Kappaleessa 4.3 käsiteltiin, millaista dataa tulisi kerätä pankkihaittaohjelmien käyttämien MITRE-tekniikoiden tunnistamiseen. Tekniikoiden tunnistamiseen liittyvä data on listattu yhteen taulukossa 4.1. Taulukossa mainitut datapisteet tulisi kerätä jokaisen sovelluksen osalta, jolloin voidaan arvioida, onko jokin sovellus mahdollisesti pankkihaittaohjelma. On kuitenkin hyvä huomata, että tässä tutkielmassa käsiteltiin rajallinen määrä MITRE-tekniikoita ja keräämisen arvoista dataa olisi saatavilla enemmänkin niiden MITRE-tekniikoiden osalta, joita tässä tutkielmassa ei käsitelty.

Hyödyllisiä rajapintoja tiedon keräämiseen

MITRE-tekniikoiden tunnistamiseen liittyvän datan pohjalta voimme myös listata yleisesti Android-sovellusrajapinnan osia, jotka sisältävät hyödyllistä dataa. Rajapintoja on listattu taulukossa 4.2.

PackageManager- ja PackageInfo-luokkia voidaan käyttää laitteeseen asennettujen sovellusten tietojen tarkastelemiseen [25]. Näitä tietoja ovat esimerkiksi sovelluksen package-nimi, signatuuri ja permissiot. Package-nimi yksilöi sovelluksen

Tekniikka	Tunnistamiseen tarvittava data
T1516	Käytettävyyspalvelusovellukset
T1626.001	Järjestelmänvalvojasovellukset, permissiot
T1636.003	READ_CONTACTS -permissio
T1517	BIND_NOTIFICATION_LISTENER_SERVICE -permissio
T1644	BIND_NOTIFICATION_LISTENER_SERVICE -permissio
T1422	READ_PRIVILEGED_PHONE_STATE -permissio
T1422.001	READ_PRIVILEGED_PHONE_STATE -permissio
T1642	Järjestelmänvalvojasovellukset, permissiot
T1636.004	READ_SMS -permissio
T1582	Oletukseksi asetettu SMS-sovellus, permissiot
T1655.001	-
T1417.001	Käytettävyyspalvelusovellukset, näppäimistösovellukset
T1417.002	SYSTEM_ALERT_WINDOW -permissio

Taulukko 4.1: MITRE-tekniikat ja data, jonka avulla voidaan havaita tekniikan käyttö Android-laitteessa.

laitteessa, joten sitä voidaan käyttää yksittäisen laitteen sovellusten listaamiseen. Haittaohjelmat voivat kuitenkin käyttää tunnettujen sovellusten package-nimiä piiloutuakseen. Kahdesta eri laitteesta löydetty sovellus ei siis välttämättä ole sama, vaikka sillä olisi sama package-nimi. Kaikki laitteeseen asennetut sovellustiedostot on kuitenkin allekirjoitettu digitaalisesti sovelluskehittäjän avaimilla [2]. Sovelluksen signatuuri auttaa siis tunnistamaan sovelluksia, kun dataa on saatavilla useasta eri laitteesta. Mikäli eri laitteista löytyy sovelluksia, jolla on sama package-nimi mutta eri signatuurit, tulisi kyseiset sovellukset tarkistaa mahdollisen haittaohjelman varalta.

AccessibilityManager-luokka puolestaan sisältää tietoa laitteseen asennetuista käytettävyyssovelluksista. Useat haittaohjelmat käyttävät hyväksi käytettävyystoimintoja saadakseen pääsyn laitteen resursseihin, joten näiden sovellusten tietojen kerääminen on myös hyödyllistä.

InputMethodManager-luokan kautta voidaan voidaan selvittää tietoa laitteeseen asennetuista näppäimistösovelluksista. Näppäimistösovellukseksi tekeytynyt haittaohjelma voi yrittää tallentaa käyttäjän syöttämiä käyttäjätunnuksia ja salasanoja, joten epäilyttävien näppäimistösovellusten tarkastelu voi auttaa haittaohjelmien tunnistamisessa.

DevicePolicyManager-luokka sisältää järjestelmänvalvojasovelluksille suunnattuja toimintoja. Tämän rajapinnan kautta on mahdollista selvittää laitteessa aktiivisena olevat järjestelmänvalvojasovellukset. Haittaohjelmat voivat käyttää hyväksi tätä ominaisuutta, joten näiden sovellusten tarkastelu voi auttaa tunnistamaan mahdollisia haittaohjelmia.

Telephony.Sms -luokan kautta puolestaan saadaan selville mm. oletetuksi asetettu SMS-sovellus, jolla on erityisen paljon käyttöoikeuksia SMS-toimintoihin liittyen. Osa haittaohjelmista voi yrittää saada käyttäjän asettamaan haittaohjelma oletetuksi SMS-sovellukseksi, joten tämän rajapinnan tarkkailu auttaa myös haittaohjelmien tunnistamisessa.

Android-rajapinta
android.content.pm.PackageManager
android.content.pm.PackageInfo
android.view.accessibility.AccessibilityManager
android.view.inputmethod.InputMethodManager
android.app.admin.DevicePolicyManager
android.provider.Telephony.Sms

Taulukko 4.2: Android-sovellusrajapintoja, joiden kautta voidaan kerätä tietoturvaan liittyvää dataa.

5 Hyökkäysten visualisointi ja tunnistaminen

Tässä luvussa käsitellään Android-laitteista kerätyn datan analysoimista. Tavoitteena on vastata tutkimuskysymykseen TK4, eli selvittää, miten Android-laitteisiin kohdistuvia hyökkäyksiä voidaan havaita niistä kerätyn datan avulla. Datan keräämistä käsiteltiin aikaisemmin luvussa 4. Tässä luvussa puolestaan vahvistetaan, että kerätty data todella auttaa tunnistamaan haitallista toimintaa. Validointia varten suoritetaan yksinkertainen koe, jossa kerätään dataa muutamasta Android-laitteesta ja piirretään kuvaajia datan pohjalta.

5.1 Koe datan keräämiseen ja visualisointiin

5.1.1 Koeasetelma

Kokeen tavoitteena on saada selville, riittääkö MITRE-tekniikoiden pohjalta kerätty data haittaohjelmien tunnistamiseen. Esimerkkinä tarkastellaan pankkihaittaohjelmien MITRE-tekniikoita käyttävää sovellusta. Tarkoituksena on testata, erottuuko testisovellus yksittäisestä laitteesta kerätystä datasta, tai kun useamman laitteen dataa vertaillaan toisiinsa.

Datan keräämiistä testataan kolmella Android-emulaattorilla, joissa kaikissa on Android 15 -käyttöjärjestelmä. Kutsumme laitteita nimillä A, B ja C. Kaikkiin kol-

meen laitteeseen asennetaan Android-sovellus, joka kerää laitteista dataa testiä varten. Tämän lisäksi laitteeseen A asennetaan testiä varten kehitetty Android-sovellus, joka käyttäytyy haittaohjelman tavoin, eli se käyttää tiettyjä MITRE-tekniikoita päästäkseen hyödyntämään laitteen resursseja.

Laitteeseen B asennetaan puolestaan neljä suosituinta Google Play -kaupassa saatavilla olevaa sovellusta. Vuonna 2024, eniten ladattuja sovelluksia olivat Facebook, Instagram, WhatsApp ja TikTok [28]. Näin näemme, erottuuko haittaohjelma tyypillisten sovellusten joukosta datan analysointivaiheessa. Kokeessa käytettävien sovellusten versionumerot on listattu taulukossa 5.1.

Sovelluksen nimi	Versionumero
WhatsApp	2.25.9.78
Instagram	375.0.0.38.66
Facebook	508.0.0.74.47
TikTok	39.5.3

Taulukko 5.1: Laitteessa B käytettävien sovellusten versionumerot.

Laite C toimii vertailupohjana eikä sisällä dataa keräävän sovelluksen lisäksi muita sovelluksia, kuin ne, mitä laitteeseen on valmiiksi asennettu.

Kun data on kerätty kaikista kolmesta laitteesta, se visualisoidaan Excel-ohjelmalla yksinkertaisina pylväsdiagrammeina. Pylväsdiagrammeista tarkistetaan, erottuuko haittaohjelma muiden sovellusten joukosta. Kokeessa ei kuitenkaan oteta kantaa siihen, miten data siirretään Android-laitteesta ulos analysoimista varten.

5.1.2 Haittaohjelmana toimiva Android-sovellus

Testiä varten kehitetty haittaohjelma on yksinkertainen Android-sovellus, joka käyttää MITRE-tekniikoita saadakseen pääsyn laitteen resursseihin. Tähän sisältyy myös

vaarallisten permissioiden käyttäminen. Testissä ei oteta kantaa siihen, miten haittaohjelmalle myönnetään sen pyytämät permissiot. Sovellus ei siis testissä varsinaisesti suorita ohjelmakoodia pyytääkseen käyttäjää myöntämään permissioita tai esimerkiksi lukeakseen käyttäjän SMS-viestejä. Tämä rajoitus on tehty, koska dataa keräävä sovellus ei kykene tunnistamaan muuta kuin permissioiden muutokset ja tiettyjen MITRE-tekniikoiden käytön. Sovelluksen tarkempi toiminta ei siis vaikuta testin tuloksiin. Testissä käyttäjä myöntää tarvittavat permissiot sovellukselle suoraan laitteen asetuksissa.

Haittaohjelman käyttämät MITRE-tekniikat on valittu pankkihaittaohjelmien pohjalta. Sovellus hyödyntää käytettävyyspalveluita ja se on mahdollista asettaa oletetuksi SMS-sovellukseksi. Tämän lisäksi sovellus pyytää kalenteri- ja SMS-permissioita. Sovelluksen havaittavissa olevat toiminnot ja niihin liitetyt MITRE-tekniikat on listattu taulukossa 5.2.

Haittaohjelman ohjelmakoodia on kuvattu liittessä A.

Toiminnon kuvaus	MITRE-tekniikat
Käytettävyyspalvelusovellus	T1516, T1417.001
READ_CONTACTS -permissio	T1636.003
READ_SMS -permissio	T1636.004
Notifikaatioita lukeva sovellus	T1517
Oletettu SMS-sovellus	T1582
RECEIVE_SMS -permissio	T1582
SEND_SMS -permissio	T1582

Taulukko 5.2: Kokeessa käytettävän haittaohjelman toimintojen kuvaukset ja toimintoihin liitetyt MITRE-tekniikat.

5.1.3 Android-sovellus datan keräämiseen

Koetta varten tarvitaan haittaohjelman lisäksi toinen sovellus, joka kerää laitteista dataa. Koska kokeen tavoitteena on yksittäisen haittaohjelman tunnistaminen, testissä kerättävä data rajoitetaan laitteeseen asennettujen sovellusten tietoihin. Kokeessa kerätään myös vain esimerkkinä käytetyn haittaohjelman tunnistamiseen tarvittavaa dataa. Kunkin sovelluksen osalta kerätään sovelluksen yksilöivä packagenimi, sovelluksen pyytämät permissiot, käytettävyyspalvelusovellukset, oletukseksi asetettu SMS-sovellus sekä notifikaatioita kuuntelevat sovellukset. Testissä ei oteta kantaa siihen, onko pyydetty permissio myönnetty vai ei. Datan keräämisessä ei vielä myöskään kiinnitetä huomiota siihen, liittyvätkö permissiot mihinkään MITRE-tekniikkaan. Permissioiden suodattaminen tehdään sen sijaan vasta silloin, kun dataa käsitellään Excel-ohjelmalla.

Sovelluksen yksinkertaistamiseksi data kerätään JSON-tiedostoihin ja tallennetaan sovelluksen omaan tiedostohakemistoon. Emulaattorilla olevia tiedostoja voidaan ladata Windows 11 -työasemalle suoraan Android Studio -kehitysympäristön kautta. Kun JSON-tiedostot on ladattu emulaattorista, ne tuodaan Excel-ohjelmaan ja visualisoidaan pylväsdiagrammeina.

Dataa keräävän sovelluksen ohjelmakoodi löytyy liitteestä B.

5.2 Kokeen tulokset

Dataa kerättiin laitteesta C heti dataa keräävän sovelluksen asentamisen jälkeen. Laitteeseen B asennettiin ensin vertailupohjana käytetyt sovellukset Google Play -sovelluskaupasta, jonka jälkeen dataa keräävä sovellus asennettiin ja data kerättiin talteen. Laitteeseen A sen sijaan ensin asennettiin kokeessa käytetty haittaohjelma. Asentamisen jälkeen haittaohjelma asetettiin oletetuksi SMS-sovellukseksi laitteen asetuksissa ennen dataa keräävän sovelluksen asentamista ja datan keräämistä. Muut

haittaohjelman käyttämät menetelmät eivät edellyttäneet lisätoimenpiteitä.

Seuraavaksi laitteista kerätty data tuotiin Excel-ohjelmaan visualisointia varten. Data sisälsi tietoa laitteisiin asennetuista sovelluksista ja sovellusten käyttämistä MITRE-tekniikoista, joita listattiin aiemmin taulukossa 5.2. Kunkin sovelluksen osalta laskettiin, kuinka monta talukossa 5.2 näkyvää datapistettä (7 kpl) datasta löytyy. Mikäli datasta löytyy suuri määrä tiettyyn sovellukseen liittyvää datapistettä, sovellus on suuremmalla todennäköisyydellä pankkihaittaohjelma.

5.2.1 Tulosten visualisointi

Kokeen tuloksia on visualisoitu Excel-kaavioina kuvissa 5.1, 5.2 ja 5.3. Kuvaajissa näkyy laitteeseen asennettujen sovellusten package-nimiä. Kunkin sovelluksen osalta näkyy myös, kuinka monta pankkihaittaohjelmaan viittaava datapistettä löytyi. Kuvista on jätetty selkeyden vuosi pois ne sovellukset, jotka sisälsivät vähiten datapisteitä.

5.2.2 Tulosten arviointi

Kuvassa 5.1 näkyy pankkihaittaohjelman sisältävästa laitteesta A kerättyä dataa. Haittaohjelmaan *com.securitydata.banker* viittavia datapisteitä löytyi kaikkein eniten, joten se on siis tunnistettavissa kerätyn datan perusteella.

On kuitenkin hyvä huomata, että joistain muista sovelluksista löytyi myös huomattava määrä datapisteita. Laitteeseen valmiiksi asennettun Google Play services -sovelluksen (com.google.android.gms) osalta datapisteitä löytyi 5 kappaletta. Laitteissa B ja C, oletetuksi SMS-sovellukseksi oli asetettu haittaohjelman sijaan Google Messages (com.google.android.apps.messaging), jolloin tämän sovelluksen osalta datapisteitä kertyi myös 5 kappaletta.

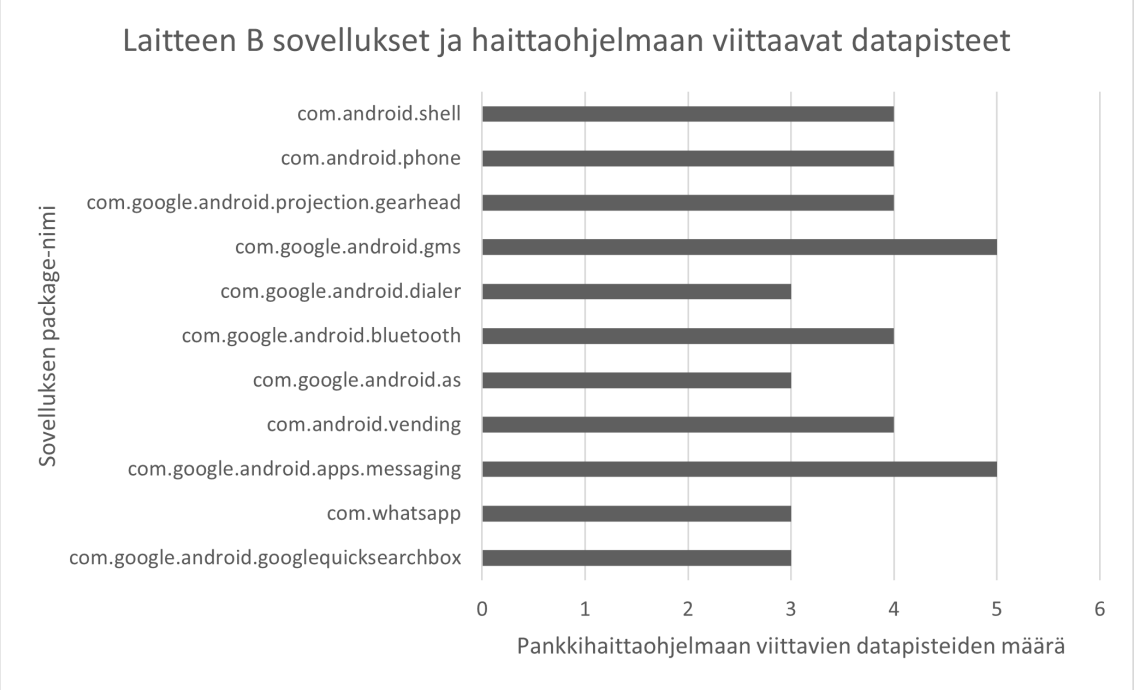
Laitteeseen B asennetuista lisäsovelluksista (Facebook, Instagram, WhatsApp, TikTok) sen sijaan ei löytynyt merkittäviä määriä datapisteitä. Laitteen B tuloksia

Kuva 5.1: (Laite A) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä.

esittävässä kuvassa 5.2 näkyy, että WhatApp -sovellukseen (com.whatsapp) viittaavia datapisteitä löytyi 3 kappaletta. Muissa lisäsovelluksissa datapisteitä oli vielä vähemmän, minkä takia ne eivät näy kuvassa ollenkaan.

Syy useiden sovellusten suuriin datapisteiden määrään näyttää liittyvän niiden pyytämiin Android-permissioihin. Tämä voidaan päätellä tarkastelemalla laitteeseen A asennettuja käytettävyyssovelluksia taulukossa 5.3, sekä notifikaatioita lukevia sovelluksia taulukossa 5.4. Taulukoista ei löydy haittaohjelman lisäksi yhtäkään samaa sovellusta, joten korkea datapisteiden määrä ei johdu näistä datapisteistä.

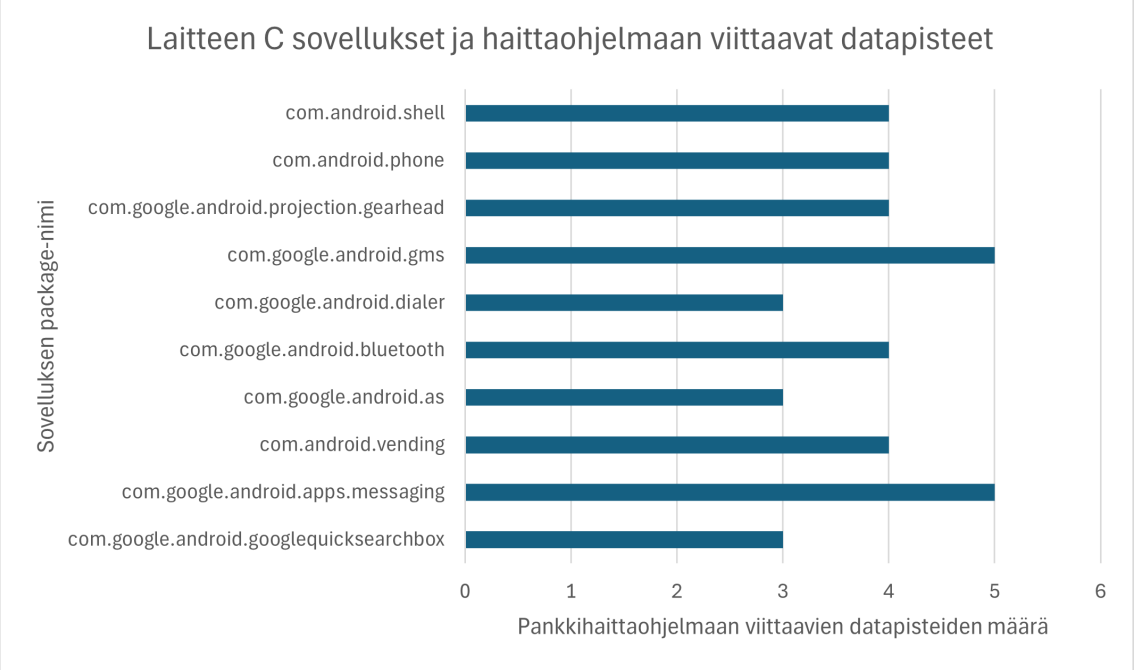
Pelkistä permissioista voi kertyä kokeessa yhteensä 4 datapistettä, mikä selvästi näkyy väärinä hälytyksinä kokeen tuloksissa. Vääriä hälytyksiä voisi vähentää tarkastelemalla permissioita laajemmin. Kokeessa tarkasteltiin pääosin SMS-



Kuva 5.2: (Laite B) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä.

toimintoihin liittyviä permissioita, mutta näitä toimintoja käyttävät tavalliset sovellukset nousevat silloin datasta esiin varsin helposti. Toinen tapa väärien hälytysten minimoimiseen olisi painottaa datapisteitä eri tavalla. Kokeessa permissioiden osuus kaikista datapisteistä oli yli puolet. Permissioiden arvoa vähentämällä haittaohjelma voisi erottua joukosta helpommin. Turvallisiksi tunnettuja sovelluksia voi toki myös karsia pois datasta, jolloin ne eivät vääristä tuloksia.

Kokeen perusteella sovellusten pyytämiä Android-permissioita tulisi siis tutkia tarkemmin, sillä ne voivat aiheuttaa vääriä hälytyksiä haittaohjelmien tunnistamisessa. Sen sijaan käytettävyyssovellusten ja notifikaatioita lukevien sovellusten tarkastelu vaikuttaisi olevan tehokas keino kokeessa käytetyn haittaohjelman tunnistamiseen. Osa laitteisiin valmiiksi asennetuista sovelluksista käytti yhtä näistä omi-



Kuva 5.3: (Laite C) Pankkihaittaohjelmaan viittaavien datapisteiden määrä kutakin laitteeseen asennettua sovellusta kohden. Kuvaajasta on jätetty pois sovellukset, joista löytyi vähiten datapisteitä.

naisuuksista, mutta ainoastaan haittaohjelma käytti molempia.

Kokeessa kuitenkin tarkasteltiin vain yhtä, koetta varten kehitettyä haittaohjelmaa. Vakuuttavampia tuloksia voitaisiin saada, jos suoritettaisiin laajempi koe oikealla haittaohjelmalla. Dataa tulisi myös kerätä laajemmin niin, että tarkasteltaisiin useampia datapisteitä sekä sovellusten pyytämiä permissioita.

Käytettävyyssovellukset
com.android.systemui.accessibility.accessibilitymenu
com.google.android.marvin.talkback
com.securitydata.banker

Taulukko 5.3: Laitteeseen A asennettujen käytettävyyssovellusten package-nimet.

Notifikaatioita lukevat sovellukset
com.google.android.apps.nexuslauncher
com.google.android.as
com.google.android.contacts
com.google.android.gms
com.google.android.projection.gearhead
com.securitydata.banker

Taulukko 5.4: Laitteeseen A asennettujen, notifikaatioita lukevien sovellusten package-nimet.

6 Yhteenveto

Tutkielmassa käsiteltiin Android-laitteiden resursseja, niihin kohdistuvia hyökkäyksiä sekä laitteista kerätyn datan analysoimista. Resurssit määriteltiin niiden Android-permissioiden pohjalta, jotka on suunniteltu suojaamaan käyttäjän yksityisyyttä koskevia tietoja sekä laitteen toimintoja. Tuloksena saatiin lista suojattavista resursseista, joka vastaa ensimmäiseen tutkimuskysymykseen (TK1), eli millaisia resursseja tulisi suojata työntekijöiden Android-laitteissa.

Toisen tutkimuskysymyksen (TK2) tavoitteena oli selvittää ne uhat, joihin tulisi kiinnittää eniten huomiota silloin, kun kyseessä on joukko organisaation työntekijöiden käyttämiä Android-laitteita. Tähän vastattiin tarkastelemalla viime vuosien aikana julkaistuja mobiililaitteiden tietoturvaan liittyviä katsauksia. Pankkihaittaohjelmat nousivat esiin yhtenä yleistyvänä uhkana, joten se valittiin tämän tutkielman osalta tarkasteluun. Tutkielmassa ei kuitenkaan käsitelty muita artikkeleissa mainittuja uhkia, joten tutkielmassa käsitelty data perustuu ainoastaan pankkihaittaohjelmien tunnistamiseen. Yksi jatkokehitysaihe olisi siis muiden uhkien, kuten vakoiluohjelmien Android-käyttöjärjestelmän haavoittuvuuksien tutkiminen ja niiden pohjalta kerrättävän datan määrittäminen.

Kolmas tutkimuskysymys (TK3) liittyi puolestaan siihen, millaista dataa Android-laitteesta voitaisiin kerätä sovellusrajapinnan kautta. Datan tunnistamisessa käytettiin apuna MITRE ATT&CK -tietokantaa, jonka avulla pankkihaittaohjelmien toiminnot kartoitettiin datapisteiksi, joita laitteeseen asennettu sovellus voisi kerätä

laitteesta haittaohjelman tunnistamiseksi. Tuloksena saatiin useita ohjelmakoodilistauksia datan keräämiseen, sekä taulukko kerättävistä tiedoista. Tulosten osalta tulee kuitenkin ottaa huomioon tietyt rajoitukset. MITRE ATT&CK tarjoaa rajallisen muotin haittaohjelmien toimintojen kartoittamiseen. Android-laitteista voisi mahdollisesti löytyä muutakin haittaohjelmiin viittaavaa dataa, jota MITRE-tekniikoita tutkimalla ei noussut esiin. Tässä tutkielmassa käsiteltiin myös vain rajallinen määrä MITRE-tekniikoita. Muiden tekniikoiden kartoittaminen tuottaisi lisää datapisteitä, joita laitteista voitaisiin kerätä ja ne auttaisivat kattamaan muitakin uhkia kuin pelkkiä pankkihaittaohjelmia.

Neljäntenä tutkimuskysymyksenä (TK4) oli selvittää, miten hyökkäyksiä voitaisiin havaita kerätyn datan perusteella. Tähän vastaamista varten suoritettiin koe, jossa testattiin tutkielmassa kerätyn datan keräämistä. Kokeessa käytettiin pankkihaittaohjelman tavoin käyttäytyvää sovellusta sekä dataa keräävää sovellusta, jotka asennettin Android-laitteeseen. Tuotoksena syntyi ohjelmakoodi datan keräämiseen, ohjelmakoodi pankkihaittaohjelman toimimintojen simuloimiseen sekä graafit kokeen tuloksista. Vaikka pankkihaittaohjelma erottui muiden sovellusten joukosta, monet laitteeseen valmiiksi asennetut sovellukset kuitenkin nousivat pankkihaittaohjelman ohella esiin testin tuloksissa. Datan analysoimisessa tulisi ottaa erityisesti huomioon Android-permissioiden vaikutus, sillä monet tavalliset sovellukset voivat käyttää samoja permissioita haittaohjelmien kanssa. Yksi jatkokehitysaihe olisi myös tekoälyn käyttäminen datan analysoimisessa.

Lähdeluettelo

[1] R. MAYRHOFER, J. V. STOEP, C. BRUBAKER et al., "The Android Platform Security Model (2023)", 2023, https://arxiv.org/pdf/1904.05572.pdf, (haettu 22.02.2024).

[2] "Android Security Paper 2023", https://services.google.com/fh/files/misc/android-enterprise-security-paper-2023.pdf, (haettu 22.02.2024).

[3] , https://source.android.com/docs/security/features/trusty, (haettu 22.02.2024).

[4] , https://source.android.com/docs/security/features/verifiedboot/, (haettu 22.02.2024).

[5] , https://source.android.com/docs/security/features/keystore, (haettu 01.03.2024).

[6] , https://source.android.com/docs/security/app-sandbox, (haettu 22.02.2024).

[7] , https://source.android.com/docs/security/features/selinux, (haettu 24.03.2024).

[8] I. M. ALMOMANI ja A. A. KHAYER, "A Comprehensive Analysis of the Android Permissions System", 2020, https://ieeexplore.ieee.org/document/9272963, (haettu 14.05.2024).

[9] P. BHAT ja K. DUTTA, ”A Survey on Various Threats and Current State of Security in Android Platform”, 2019, https://dl.acm.org/doi/abs/10.1145/3301285, (haettu 22.04.2024).

[10] , https://developer.android.com/reference/android/Manifest.permission_group, (haettu 25.08.2024).

[11] S. ALSOGHYER ja I. ALMOMANI, ”On the Effectiveness of Application Permissions for Android Ransomware Detection”, 2020, https://ieeexplore.ieee.org/document/9044263, (haettu 14.05.2024).

[12] , https://attack.mitre.org/techniques/, (haettu 07.02.2025).

[13] B. AL-SADA, A. SADIGHIAN ja G. OGLIERI, ”MITRE ATT&CK: State of the Art and Way Forward”, 2024, https://dl.acm.org/doi/10.1145/3687300, (haettu 04.10.2024).

[14] N. NAIK, P. JENKINS, P. GRACE ja J. SONG, ”Comparing Attack Models for IT Systems: Lockheed Martin’s Cyber Kill Chain, MITRE ATT&CK Framework and Diamond Model”, 2022, https://ieeexplore.ieee.org/document/10005490, (haettu 31.08.2024).

[15] R. KHAN, K. MCLAUGHLIN, D. LAVERTY ja S. SEZER, ”STRIDE-based Threat Modeling for Cyber-Physical Systems”, 2017, https://ieeexplore.ieee.org/abstract/document/8260283, (haettu 06.12.2024).

[16] N. SHEVChENKO, T. A. CHICK, P. O’RIORDAN, T. P. SCANLON ja C. WOODY, ”THREAT MODELING: A SUMMARY OF AVAILABLE METHODS”, 2018, https://insights.sei.cmu.edu/documents/569/2018_019_001_524597.pdf, (haettu 08.12.2024).

[17] ”ENISA THREAT LANDSCAPE 2024”, https://www.enisa.europa.eu/sites/default/files/11/ENISA Threat Landscape 2024_0.pdf, (haettu 01.01.2025).

[18] ”Gen Digital Q3 2024 Threat Report”, https://cmsb.nortonlifelock.com/sites/default/files/ 11/Q3 Threat Labs Whitepaper final_0.pdf, (haettu 31.1.2025).

[19] ”Zimperium 2023 Global Mobile Threat Report”, https://lp.zimperium.com/hubfs/MAPS_MTD/REPORT/GEN/Global_Mobile_Threat_Report_2023.pdf, (haettu 15.12.2024).

[20] ”Zimperium 2024 Global Mobile Threat Report”, https://lp.zimperium.com/hubfs/MAPS Mobile Threat Report 2024 FINAL (1).pdf, (haettu 15.12.2024).

[21] , https://attack.mitre.org/software/S0522/, (haettu 07.02.2025).

[22] , https://attack.mitre.org/software/S1055/, (haettu 07.02.2025).

[23] , https://developer.android.com/reference/android/app/AppOpsManager, (haettu 07.03.2025).

[24] , https://developer.android.com/training/package-visibility, (haettu 22.02.2025).

[25] , https://developer.android.com/reference/android/content/pm/PackageManager, (haettu 28.02.2025).

[26] , https://developer.android.com/guide/topics/ui/accessibility/service, (haettu 21.02.2025).

[27] , https://developer.android.com/reference/android/app/admin/DevicePolicyManager, (haettu 22.02.2025).

[28] , https://www.statista.com/statistics/1448018/global-leading-android-downloaded-mobile-apps/, (haettu 29.03.2025).

Liite A Pankkihaittaohjelma (Android-sovellus)

Tutkielmassa kehitettiin Android-sovellus, joka käyttäytyy pankkihaittaohjelman tavoin. Ohjelmalistaus 7 näyttää sovelluksen pyytämät permissiot SMS -toimintojen käyttämiseen ja kontaktien lukemiseen. Ohjelmalistauksissa 8 ja 9 puolestaan näkyy esimerkki notifikaatioita kuuntelevasta Service-luokasta, joka vaaditaan, jotta sovellukselle voi antaa luvan notifikaatioiden lukemiseen.

Ohjelmalistaukset 10 ja 11 sisältävät esimerkin Service-luokasta, joka vaaditaan käytettävyyspalveluiden hyödyntämiseen. Service vaaditaan, jotta käytettävyystoiminnot voidaan aktivoida laitteen asetuksissa.

Jotta sovellus voidaan asettaa oletetuksi SMS-sovellukseksi, sen pitää sisältää asianmukaiset Service-, Activity- sekä kaksi Receiver -luokkaa. Esimerkit näistä on kuvattu ohjelmalistauksissa 12, 13, 14, 15, 16 sekä 17.

Ohjelmalistaus 7 Sovelluksen käyttämät permissiot (AndroidManifest.xml)

```
{XML}

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

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

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

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

<uses-feature android:name="android.hardware.telephony"

    android:required="false"/>
```

Ohjelmalistaus 8 Service notifikaatioiden lukemiseen (AndroidManifest.xml)

```
{XML}
<service android:name=".BankerNotificationListenerService"
    android:label="Banker"
    android:exported="false"
    android:permission=
        "android.permission.BIND_NOTIFICATION_LISTENER_SERVICE">
    <intent-filter>
        <action android:name=
            "android.service.notification.NotificationListenerService" />
    </intent-filter>
    <meta-data
        android:name="android.service.notification.default_filter_types"
        android:value="conversations|alerting">
    </meta-data>
    <meta-data
        android:name="android.service.notification.disabled_filter_types"
        android:value="ongoing|silent">
    </meta-data>
</service>
```

Ohjelmalistaus 9 Service notifikaatioiden lukemiseen (Java-luokka)

```
{Java}

package com.securitydata.banker;

import android.service.notification.NotificationListenerService;

public class BankerNotificationListenerService

    extends NotificationListenerService {

    @Override

    public void onCreate() {

        super.onCreate();

    }

    @Override

    public void onListenerConnected() {

        super.onListenerConnected();

    }

}
```

Ohjelmalistaus 10 Service käytettävyyspalvelujen hyödyntämiseen (AndroidManifest.xml)

```
{XML}
<service android:name=".BankerAccessibilityService"
    android:exported="true"
    android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
    android:label="Banker">
    <intent-filter>
        <action android:name=
            "android.accessibilityservice.AccessibilityService" />
    </intent-filter>
    <meta-data
        android:name="android.accessibilityservice"
        android:resource="@xml/accessibility_service_config" />
    </service>
```

Ohjelmalistaus 11 Service käytettävyyspalvelujen hyödyntämiseen (Java-luokka)

```
{Java}
package com.securitydata.banker;
import android.accessibilityservice.AccessibilityService;
import android.view.accessibility.AccessibilityEvent;
public class BankerAccessibilityService extends AccessibilityService {
    @Override
    public void onAccessibilityEvent(
        AccessibilityEvent accessibilityEvent) {}
    @Override
    public void onInterrupt() {}
}
```

Ohjelmalistaus 12 SMS-toimitoihin vaadittu Activity ja Service (AndroidManifest.xml)

```
{XML}
<activity
    android:name=".SmsActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.SENDTO" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="sms" />
        <data android:scheme="smsto" />
        <data android:scheme="mms" />
        <data android:scheme="mmsto" />
    </intent-filter>
</activity>
<service
    android:name=".SmsService"
    android:exported="true"
    android:permission="android.permission.SEND_RESPOND_VIA_MESSAGE">
    <intent-filter>
        <action android:name="android.intent.action.RESPOND_VIA_MESSAGE" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="sms" />
        <data android:scheme="smsto" />
        <data android:scheme="mms" />
        <data android:scheme="mmsto" />
    </intent-filter>
</service>
```

Ohjelmalistaus 13 SMS-toimitoihin vaadittu Receiver, 2kpl (AndroidManifest.xml)

```
{XML}
<receiver
    android:name=".SmsReceiver"
    android:exported="true"
    android:permission="android.permission.BROADCAST_SMS">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_DELIVER" />
    </intent-filter>
</receiver>
<receiver
    android:name=".MmsReceiver"
    android:exported="true"
    android:permission="android.permission.BROADCAST_WAP_PUSH">
    <intent-filter>
        <action android:name="android.provider.Telephony.WAP_PUSH_DELIVER" />
        <data android:mimeType="application/vnd.wap.mms-message" />
    </intent-filter>
</receiver>
```

Ohjelmalistaus 14 Activity SMS-toimintoihin (Java-luokka)

```
{Java}
package com.securitydata.banker;
import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
public class SmsActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

Ohjelmalistaus 15 Service SMS-toimintoihin (Java-luokka)

```
{Java}
package com.securitydata.banker;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import androidx.annotation.Nullable;
public class SmsService extends Service {
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Ohjelmalistaus 16 SMS Receiver (Java-luokka)

```
{Java}

package com.securitydata.banker;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

public class SmsReceiver extends BroadcastReceiver {

    @Override

    public void onReceive(Context context, Intent intent) {}

}
```

Ohjelmalistaus 17 MMS Receiver (Java-luokka)

```
{Java}

package com.securitydata.banker;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

public class MmsReceiver extends BroadcastReceiver {

    @Override

    public void onReceive(Context context, Intent intent) {}

}
```

Liite B Android-sovellus datan keräämiseen

Tutkielmassa kehitettiin Android-sovellus, joka kerää laitteesta tietoturvaan liittyvää dataa. Ohjelmalistaus 18 näyttää sovelluksen pyytämän permission, jonka se tarvitsee kerätäkseen tietoa muista laitteeseen asennetuista sovelluksista.

Ohjelmalistauksessa 19 puolestaan näkyy, miten kerätty data tallennetaan JSON-tiedostoihin sovelluksen omaan tiedostohakemistoon. Muut esitellyt metodit käyttävät tätä ohjelmalistausta datan tallentamiseen.

Kaikkien laitteeseen asennettujen sovellusten package-nimet sekä niiden käyttämät permissiot kerätään ohjelmalistauksissa 20 ja 21.

Ohjelmalistauksessa 22 puolestaan kerätään notifikaatioita lukevien sovellusten package-nimet. Samaan toiminnallisuuteen liittyvä ohjelmalistaus 23 tallentaa package-nimet JSON-tiedostoon.

Vastaavasti ohjelmalistauksessa 24 kerätään kaikki laitteeseen asennettujen käytettävyyssovellusten package-nimet. Ohjelmalistaus 25 taas näyttää, miten käytettävyyssovellusten package-nimet tallennetaan JSON-tiedostoon.

Ohjelmalistauksessa 26 kerätään laitteen oletetun SMS-sovelluksen package-nimi, joka tallennetaan JSON-tiedostoon.

Ohjelmalistaus 18 Sovelluksen käyttämä permissio (AndroidManifest.xml)

```
{XML}

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

Ohjelmalistaus 19 Datan tallentaminen JSON-tiedostoon.

```
{Java}

private void saveToFile(String fileName, String content) {

    try {

        getApplicationContext().deleteFile(fileName);

        FileOutputStream fos = getApplicationContext().openFileOutput(

            fileName, Context.MODE_PRIVATE);

        fos.write(content.getBytes());

        fos.close();

    } catch (IOException e) {}

}
```

Ohjelmalistaus 20 Laitteeseen asennettujen sovellusten käyttämien permissioiden kerääminen (1/2).

```
{Java}
private void collectPermissions(boolean onlyGranted) {
    try {
        JSONObject jsonObject = new JSONObject();
        JSONArray appsArray = new JSONArray();
        PackageManager pm = getApplicationContext().getPackageManager();
        List<PackageInfo> packages = pm.getInstalledPackages(
            PackageManager.GET_PERMISSIONS);
        for(PackageInfo packageInfo : packages) {
            JSONObject appObject = new JSONObject();
            JSONArray permissionsArray = new JSONArray();
            appObject.put("packageName", packageInfo.packageName);
            String[] permissions = packageInfo.requestedPermissions;
            if(permissions != null) {
                for(String p: permissions) {
                    if(onlyGranted) {
                        if(PackageManager.PERMISSION_GRANTED ==
                        pm.checkPermission(p, packageInfo.packageName)) {
                            permissionsArray.put(p);
                        }
                    } else {
                        permissionsArray.put(p);
                    }
                }
            }
// jatkuu seuraavalla sivulla...
```

Ohjelmalistaus 21 Laitteeseen asennettujen sovellusten käyttämien permissioiden kerääminen (2/2).

```
{Java}
// ...jatkoa edelliseltä sivulta
            if(onlyGranted) {
                appObject.put("grantedPermissions", permissionsArray);
            } else {
                appObject.put("requestedPermissions", permissionsArray);
            }
            appsArray.put(appObject);
        } //for()
        jsonObject.put("packages", appsArray);
        if(onlyGranted) {
            saveToFile("granted-permissions.json", jsonObject.toString());
        } else {
            saveToFile("requested-permissions.json", jsonObject.toString());
        }
    } catch (JSONException e) {}
} //collectPermissions()
```

Ohjelmalistaus 22 Notifikaatioita lukevien sovellusten package-nimien keräämi-nen.

```
{Java}
private List<String> getNotificationListenerApps() {
    ArrayList<String> apps = new ArrayList<>();
    PackageManager pm = getApplicationContext().getPackageManager();
    Intent intent = new Intent(
        "android.service.notification.NotificationListenerService");
    List<ResolveInfo> resolveInfos = pm.queryIntentServices(intent,
            PackageManager.GET_META_DATA);
    for (ResolveInfo resolveInfo : resolveInfos) {
        if(resolveInfo.serviceInfo != null) {
            String permission = resolveInfo.serviceInfo.permission;
            if (permission.equals(
            "android.permission.BIND_NOTIFICATION_LISTENER_SERVICE")) {
                apps.add(resolveInfo.serviceInfo.packageName);
            }
        }
    }
    return apps;
}
```

Ohjelmalistaus 23 Notifikaatioita lukevien sovellusten package-nimien tallentaminen JSON-tiedostoon.

```
{Java}

private void collectNotificationListenerApps() {

    List<String> accessibilityApps = getNotificationListenerApps();

    try {

        JSONObject appsObject = new JSONObject();

        JSONArray appsArray = new JSONArray();

        for (String app : accessibilityApps) {

            appsArray.put(app);

        }

        appsObject.put("notificationListenerApps", appsArray);

        saveToFile("notification-listeners.json", appsObject.toString());

    } catch (JSONException e) {}

}
```

Ohjelmalistaus 24 Käytettävyyssovellusten package-nimien kerääminen.

```
{Java}
private List<String> getAccessibilityApps(boolean enabledOnly) {
    ArrayList<String> accessibilityApps = new ArrayList<>();
    AccessibilityManager am = getApplicationContext()
        .getSystemService(AccessibilityManager.class);
    List<AccessibilityServiceInfo> infos;
    if(enabledOnly) {
        infos = am.getEnabledAccessibilityServiceList(
            AccessibilityServiceInfo.FEEDBACK_ALL_MASK);
    } else {
        infos = am.getInstalledAccessibilityServiceList();
    }
    for(AccessibilityServiceInfo info : infos) {
        // Same app may use many accessibility features,
        // but we only want to list it once.
        if(!accessibilityApps.contains(
        info.getResolveInfo().serviceInfo.packageName)) {
            accessibilityApps.add(
                info.getResolveInfo().serviceInfo.packageName);
        }
    }
    return accessibilityApps;
}
```

Ohjelmalistaus 25 Käytettävyyssovellusten package-nimien tallentaminen JSON-tiedostoon.

```
{Java}

private void collectAccessibilityApps() {

    List<String> accessibilityApps = getAccessibilityApps(false);

    try {

        JSONObject appsObject = new JSONObject();

        JSONArray appsArray = new JSONArray();

        for (String app : accessibilityApps) {

            appsArray.put(app);

        }

        appsObject.put("allAccessibilityApps", appsArray);

        saveToFile("all-accessibility-apps.json", appsObject.toString());

    } catch (JSONException e) {}

}
```

Ohjelmalistaus 26 Oletetun SMS-sovelluksen package-nimen kerääminen.

```
{Java}

private void collectDefaultSmsApp() {

    String defaultSmsApp = Telephony.Sms.getDefaultSmsPackage(

        getApplicationContext());

    if(defaultSmsApp == null) {

        defaultSmsApp = "";

    }

    try {

        JSONObject jsonObject = new JSONObject();

        jsonObject.put("defaultSmsApp", defaultSmsApp);

        saveToFile("default-sms-app.json", jsonObject.toString());

    } catch (JSONException e) {}

}
```