Paavo Pohjolainen

Androidin tietoturva

Tietotekniikan kandidaatintutkielma

7. joulukuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Paavo Pohjolainen

Yhteystiedot: pohjolpa@student.jyu.fi

Ohjaaja: Tuomo Rossi

Työn nimi: Androidin tietoturva

Title in English: Security of Android

Työ: Kandidaatintutkielma

Sivumäärä: 20+0

Tiivistelmä: Tämä kandidaatintutkielma käsittelee Androidin tietoturvaa ja siihen liittyviä haasteita. Tutkielmassa tarkastellaan Androidin tietoturvamallia, erilaisia tietoturvauhkia, kuten haavoittuvuuksia ja haittaohjelmia, sekä tietoturvan kehittämiseen liittyviä menetelmiä. Erityistä huomiota kiinnitetään sovellusten tietoturvan kehittämiseen ja siihen, kuinka erilaiset analysointitekniikat ja menetelmät voivat auttaa parantamaan tietoturvan tasoa.

Avainsanat: Android-tietoturva, haavoittuvuudet, haittaohjelmat, sovellusten tietoturva

Abstract: This bachelor's thesis addresses the security of Android and the challenges associated with it. The study examines Android's security model, various security threats such as vulnerabilities and malware, and methods related to the development of security. Special attention is given to the development of security for applications and how different analysis techniques and methods can help improve the level of security.

Keywords: Android security, vulnerabilities, malware, application security

Kuviot

Sisällys

1 Johdanto

Tämä tutkielma käsittelee Android-käyttöjärjestelmän tietoturvan nykytilaa, sekä sen tulevaisuuden näkymiä. Android on avoimen lähdekoodin käyttöjärjestelmä, joka pohjautuu Linux-ytimeen. Android tunnetaan parhaiten mobiilikäyttöjärjestelmänä, mutta Androidia voidaan havaita myös televisiossa, autoissa, älykelloissa sekä muissa älylaitteissa. Avoimen lähdekoodin vuoksi valmistajien on mahdollista räätälöidä Android käyttöjärjestelmä laitteisiinsa. Tämä on tehnyt Androidista suositun valinnan monille laitteiden valmistajille ja sovelluskehittäjille.

Statista 2023 mukaan tällä hetkellä arviolta 6,9 miljardia henkilöä käyttää älypuhelinta. Näistä käyttäjistä noin 67,7% käyttää Android-käyttöjärjestelmällä varustettua älypuhelinta (Statcounter 2023). Käyttäjät säilyttävät älypuhelimissaan paljon arkaluontoista tietoa, kuten pankkitietoja, liikesalaisuuksia, salasanoja, kuvia ja videoita. Tämän vuoksi on tärkeää, että Androidin tietoturvan taso vastaa käyttäjien tarpeita ja tietoturvaa kehitetään jatkuvasti.

Androidin suosio ja sen käyttäjämäärän kasvu ovat kuitenkin johtaneet tietoturvariskien kasvuun. Avoimen lähdekoodin vuoksi Android on käyttöjärjestelmänä haavoittuvaisempi, kuin kilpailevat käyttöjärjestelmät, jotka ovat suljettuun lähdekoodiin pohjautuvia käyttöjärjestelmiä. Vuonna 2022 Android käyttöjärjestelmällä oli 899 haavoittuvuutta (CVE-Details 2022a), kun taas kilpailevalla iOS käyttöjärjestelmällä haavoittuvuuksia oli vain 242 (CVE-Details 2022b). Android-laitteisiin kohdistuu jatkuvasti haittaohjelmia, tietomurtoja ja tietojenkalasteluyrityksiä, jotka voivat aiheuttaa vakavia seurauksia käyttäjille. Lisäksi monet Android- laitteiden käyttäjät eivät ole tietoisia laitteidensa tietoturvasta ja riskitekijöistä.

Tutkielmassa käydään läpi Androidin useita eri tietoturvauhkia, sekä kootaan yhteen Androidin tietoturvaa parantavia tekniikoita. Tutkielmassa käsitellään tietoturvamallia, jonka päälle käyttöjärjestelmä on rakennettu. Tietoturvamalli sisällyttää ehdot, joiden perusteella käyttöjärjestelmälle annetaan oikeudet toimia. Tietoturvamalli osiossa käsitellään myös Androidin ekosysteemiä sekä uhkamallia. Näiden avulla käyttöjärjestelmän käyttökokemus, sekä tietoturva pysyvät käsikädessä. Tutkielmassa avataan myös Androidin tietoturvahaavoittuvuuksia, sekä käsitellään yleisimpiä haittaohjelmia. Lopuksi tutkielmassa käsitellään Androidin

tietoturvaa parantavia tekniikoita, kuten staattista analysointia, dynaamista analysointia sekä hybridianalysointia. Näiden avulla voidaan ennaltaehkäistä haavoittuvuuksien syntymistä jo kehitysvaiheessa. Dynaamisen- ja hybridianalysoinnin avulla voidaan tunnistaa haavoittuvuuksia myös jo julkaistuista sovelluksista.

2 Androidin tietoturvamalli

2.1 Ekosysteemi ja uhkamalli

Toimivassa ekosysteemissä jokainen osapuoli hyötyy ekosysteemin kasvusta, mutta osapuolten käyttämä yhteinen luottamus on minimaalista. Tämän vuoksi alustan täytyy luoda ympäristöjä jotka ovat safe-by-default. Mayrhofer ym. 2021 mukaan ympäristöissä keskeisimmät osapuolet (loppukäyttäjä, sovelluskehittäjä, käyttöjärjestelmä) voivat määrittää sitoutumisehdot, jotka edesauttavat toisiaan. Turvallisin tapa toimia tilanteessa, jossa osapuolet eivät pääse sopimukseen, on estää toiminta kokonaan (default-deny). Android käyttöjärjestelmän tietoturvamalli on rakennettu tämän ekosysteemipohjan päälle.

Loppukäyttäjäkeskeisenä käyttöjärjestelmänä, Androidin täytyy olla käytännöllinen käyttäjille, mutta myös houkutteleva kehittäjille. Mayrhofer ym. 2021 kertoo kuinka käyttöliittymän, sekä workflow:n tulee olla oletuksena turvallisia ja niiden täytyy vaatia selkeää tarkoitusta kaikille toimille, jotka voivat vaarantaa turvallisuutta, sekä yksityisyyttä. Loppukäyttäjäkeskeisenä käyttöjärjestelmänä vaaditaan myös, että teknisesti yksityiskohtaisia turvallisuus- tai yksityisyyspäätöksiä ei anneta käyttäjän vastuulle, jolla ei ole tarpeeksi ammattitaitoa tai kokemusta asiasta.

Mayrhofer ym. 2021 mukaan laiteohjelmistot täytyy sertifioida, vain jos ne sisältävät Google Palvelut. Alkuperäisten laitteiden valmistajat (OEM) voivat käyttää laitteiden käyttöjärjestelmänä AOSP (Android Open Source Project) pohjaista käyttöjärjestelmää, joka ei sisällä Google Palveluita. Laitteiden jotka markkinoivat yhteensopivuutta Android sovellusten kanssa täytyy läpäistä Compatibility Test Suite (CTS) testit.

Mayrhofer ym. 2021 kertoo myös, että mobiililaitteiden ja työpöytälaitteiden uhkamalli eroaa pääosin kahdella tavalla. Mobiililaitteet häviävät helpommin, sekä niitä varastetaan useammin. Ne myös yhdistävät tuntemattomiin verkkoihin, joka on oletettua käyttötoimintaa. Mobiililaitteet sisältävät usein myös enemmän henkilökohtaisia tietoja, kuin muut laitekategoriat.

2.2 Tietoturvamalli

Android käyttöjärjestelmän ympärille luotu ekosysteemi, sekä uhkamalli rakentaa pohjan tietoturvamallin toiminnalle.

Mayrhofer ym. 2021 määrittelee Androidin tietoturvamallin viiteen sääntöön.

Monen osapuolen suostumus

Mitään toimintoa ei suoriteta ellei kaikki osapuolet ole siihen suostuneet. Yleisimmin osapuolina toimii käyttäjä, alusta, sekä sovellus. Jos edes yksi osapuoli ei suostu toimintaan, niin toimintaa ei suoriteta.

Avoin pääsy ekosysteemiin

Käyttäjät, sekä sovelluskehittäjät ovat osa avointa ekosysteemiä, joka ei rajoitu yhteen sovelluskauppaan. Kehittäjien tarkkailu tai käyttäjien rekisteröinti ei ole vaadittua. Alustapohjaisen rajapinnan sijasta, kehittäjien on mahdollista luoda oma rajapinta, jonka avulla he kommunikoivat toisille sovelluksille.

Turvallisuus on yhteensopivuusvaatimus

Compatibility Definition Document (CDD) määrittää, että tietoturvamalli on osa Androidia. Laitteet, jotka eivät läpäise CTS, Vendor tai muita testejä, eivät ole luokiteltu Androidiksi. Laitteen roottaaminen, bootloader lukituksen avaaminen ja muutetun laiteohjelmiston asentaminen voivat olla CDD:n vastaisia toimia.

Tehdasasetusten palautus palauttaa laitteen turvalliseen tilaan

Tehdasasetusten palauttaminen uudelleenformatoi dataosiot joihin on mahdollista kirjoittaa, sekä palauttaa laitteen tilaan, jossa se on riippuvainen vain koskemattomista dataosioista. Laite saadaan takaisin turvalliseen tilaan ilman, että järjestelmäohjelmistoa tarvitsee uudelleen asentaa.

Sovellukset ovat turvallisuusperiaatteita

Androidissa käyttäjän toimesta suoritetut sovellukset eivät omaa täysiä käyttöoikeuksia, vaan ne ovat rajoitettuja.

3 Tietoturvauhat

3.1 Haavoittuvuudet

Garg ja Baliyan 2021 määrittelee haavoittuvuuden heikkoudeksi, jota hyökkääjä voi hyödyntää luvattomien toimintojen toteuttamiseksi verkossa tai järjestelmässä. Haavoittuvuudet mobiililaitteissa voivat johtua käyttäjien tietoturvakäytännöistä sekä heikosta teknisestä valvonnasta.

Androidin sekä iOS käyttöliittymien haavoittuvuudet ovat laskussa sitten vuoden 2017. Garg ja Baliyan 2021 mukaan haavoittuvuuksista noin 61% on Androidin haavoittuvuuksia, kun taas 39% iOS:n haavoittuvuuksia. Androidin keskiarvopisteytys haavoittuvuuksien vakavuudelle on 6,9, iOS:n puolestaan 6,2. Voidaan havaita, että Androidin haavoittuvuudet ovat myös vakavampia, kuin iOS:n. Tiedon kerääminen, koodin suorittaminen, palvelunesto, ylivuoto ja oikeuksien saaminen ovat Androidille tyypillisimpiä haavoittuvuuksia. iOS:n haavoittuvuudet sisältävät pääosin luvattomia pääsyjä, sekä muistin ylivuodon haavoittuvuuksia.

Garg ja Baliyan 2021 listaa yleisimmät haavoittuvuudet Androidissa.

Palvelunesto (Denial-of-Service) haavoittuvuudessa, korkealla järjestelmäresurssien käytöllä aiheutetaan palvelunestotila.

Koodin suorittaminen (Code Execution Vulnerability) haavoittuvuus pääosin ilmeni Androidin mediakehyksessä. Haavoittuvuus salli koodin suorittamisen etänä. Tästä johtuen haitallisien ohjelmien suorittaminen oli mahdollista.

Ylivuoto (Overflow) haavoittuvuus Androidissa yleisimmin kohdistuu kekomuistiin. Hyökkääjä lähettää etänä paketin, joka johtaa koodin suorittamiseen ja puskurin ylivuotoon. Hyökkääjä saa näinollen pääsyn järjestelmään.

Muistin korruptointi (Memory Corruption) Aiheutuu, kun käyttäjä allokoi muistia yli varojen. Muistia voi korruptoida myös tarkoituksellisesti.

SQL injektio (SQL Injection) haavoittuvuus pohjautuu Android sovellusten käyttämään SQ-

Lite tietokantaan. SQL injektiot voivat antaa hyökkääjälle luvattomat pääsyn tietokantaan, sekä kirjautumistietoihin.

Cross site scripting (XSS) haavoittuvuus antaa hyökkääjälle mahdollisuuden suorittaa haitallista koodia WebViewiin. Tämän seurauksena henkilökohtaisia tietoja ja selainevästeitä saatetaan anastaa.

Hakemiston läpikäynti (Directory Traversal) haavoittuvuus mahdollistaa hyökkääjälle pääsyn järjestelmän tiedostoihin ja kansioihin, jotka ovat juurikansion ulkopuolella.

HTTP vastauksen jakaminen (HTTP Response Splitting) haavoittuvuus mahdollistaa HTTP viestin katkaisun käyttämällä erinäisiä merkkejä, kuten CR tai LF.

3.2 Haittaohjelmat

Haittaohjelma on ohjelma, joka on suunniteltu aiheuttamaan vahinkoa järjestelmälle. Haittaohjelmat voivat Garg ja Baliyan 2021 mukaan kiertää tietoturvamekanismeja, kerätä henkilökohtaisia tietoja, näyttää tarpeettomia mainoksia tai häiritä järjestelmän tavallisia toimintoja.

Troijalainen (Trojan) on haittaohjelma, joka yleisimmin naamioidaan hyödylliseksi ohjelmaksi. Troijalainen voi saada pääsyn järjestelmään usean eri tavan kautta, kuten navigoimalla suojaamattomilla sivustoilla tai lataamalla liitetiedostoja kalastelusähköposteista. Troijalainen voi varastaa henkilökohtaisia tietoja tai se voi muokata dataa.

Kiristyshaittaohjelma (Ransomware) rajaa käyttäjän pääsyn järjestelmän resursseihin, jonka jälkeen hyökkääjä kiristää uhria taloudellisesti, jotta käyttäjä saa pääsyn järjestelmään takaisin. Yleisimmät kiristyshaittaohjelmat mobiililla ovat kryptovaluuttoja hyödyntäviä haittaohjelmia.

Takaovi (Backdoor) on haittaohjelma, joka ohittaa järjestelmän autentikontimekanismit. Seurauksena hyökkääjä saa pääsyn järjestelmän tietokantaan, sekä tiedostoihin. Takaovihaittaohjelma vaatii järjestelmävalvojan oikeudet, jotka Androidilla saadaan roottaamalla laite.

Vakoiluohjelma (Spydoor) yleisimmin tunkeutuu järjestelmään tavoin, jolla käyttäjä ei huo-

maa haittaohjelman olemassaoloa. Vakoiluohjelma kerää henkilökohtaisia tietoja ja jakaa tiedot mainostajille, data yrityksille tai ulkopuolisille käyttäjille. Androidilla vakoiluohjelma usein saadaan käyttäjän laitteelle sosiaalisten metodien avulla, kuten huijaamalla käyttäjä lataamaan sovelluksen kauppapaikalta.

Mainosohjelma (Adware) Näyttää käyttäjälle toivomattomia mainoksia, jotka voivat säädellä selaimen asetuksia, varastaa henkilökohtaisia tietoja tai ajaa koodia laitteessa.

4 Tietoturvan kehittäminen

Senanayake ym. 2023 mukaan tiheän Android sovellusten kehittämissyklin vuoksi suurimpaan osaan sovelluksista ei integroida kunnolla turvamekanismeja. Tämän seurauksena sovellusten haavoittuvuudet lisääntyvät kehitysvaiheessa. Google Play Store ei myöskään tarkista sovelluksissa olevia haavoittuvuuksia julkaisuvaiheessa, jonka vuoksi käyttäjät voivat kokea ongelmia tietoturvatarkastuksen puutteellisuudesta johtuen. Senanayake ym. 2023 kertoo, että tämän vuoksi on tärkeää integroida Android lähdekoodin haavoittuvuuksien tunnistamiseen tarkoitettuja mekanismeja ja työkaluja kehitysvaiheessa.

Sovelluksien analysointi on prosessi, jossa Android sovelluksen lähdekoodia ja käyttäytymistä tarkastellaan. Senanayake ym. 2023 identifioi kolme analysointitekniikkaa, joita käytetään sovellusten analysoinnissa. Staattinen analysointi, dynaaminen analysointi sekä hybridianalysointi.

Staattisessa analysoinnissa Android sovellusta ei ajeta, vaan sen lähdekoodia ja XML tiedostoja tarkastellaan. Kouliaridis ja Kambourakis 2021 kertoo staattisen analysoinnin olevan tehokas menetelmä vanhempien haittaohjelmien tunnistamiseen, mutta tehoton tilanteissa, jossa koodia on peitelty tai suojattu.

Dynaamisessa analysoinnissa sovellus suoritetaan hiekkalaatikkoympäristössä. Jotta dynaamista analysointia voidaan käyttää, tarvitaan valmis sovellus, esimerkiksi APK tiedosto. Tämän vuoksi dynaaminen analysointi on Senanayake ym. 2023 mukaan laajasti käytetty tapa tunnistaa haavoittuvuuksia ja haittaohjelmia jo kehitetyistä sovelluksista.

Hybridi-analysoinnissa hyödynnetään sekä staattista että dynaamista analysointia.

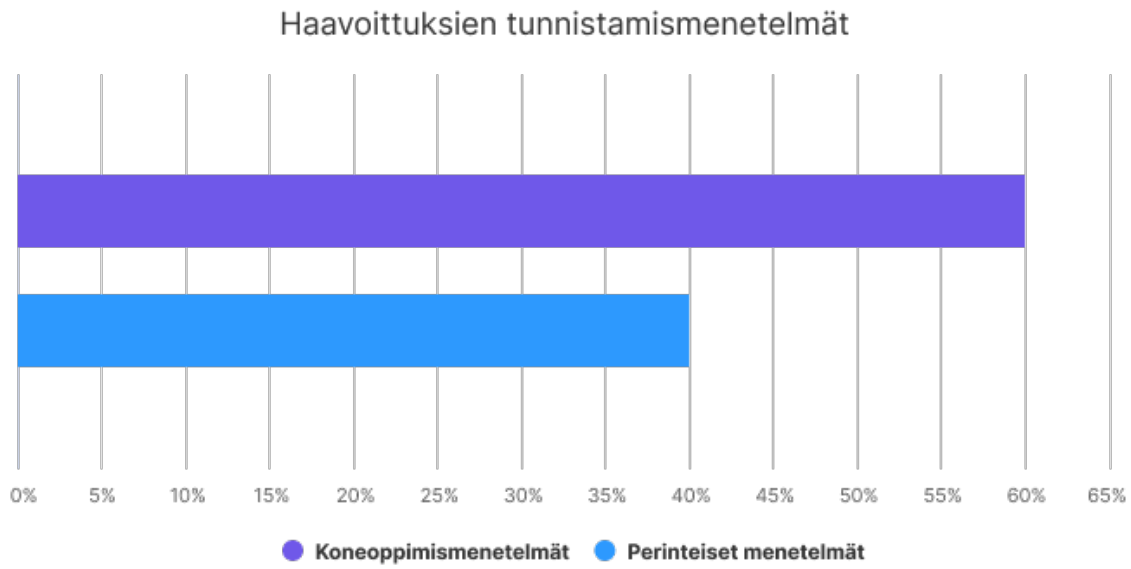
Sovelluksien avulla voidaan läpäistä tietoturvamekanismeja, hyödyntäen lähdekoodista löytyviä haavoittuvuuksia. On mahdotonta kehittää sovelluksia täysin tietoturvalliseksi, mutta tietoturva-aukkojen minimoimiseksi haavoittuvuudet lähdekoodissa täytyy tunnistaa. Senanayake ym. 2023 mukaan erinäisillä menetelmillä kuten koneoppimisella, syväoppimisella, heuristisilla menetelmillä ja formaaleilla menetelmillä yhdessä sovellusanalysoinnin kanssa voidaan tunnistaa haavoittuvuuksia lähdekoodissa.

Kuvio 1. Sovellusanalyysitekniikoiden jakauma Senanayake ym. 2023 laatimassa tutkimuksessa.

Gajrani ym. 2020 ehdotti Android haavoittuvuuksien havaitsemiseen ja korjaamiseen frameworkkia nimeltä Vulvet. Framework perustui staattisen analysoinnin tekniikkaan. Vulvet käytti haavoittuvuuksien ratkaisua, kontrollivirran instrumentointia, metodien/parametrien rekonstruointia, turvallisten metodikutsujen lisäämistä, manifestin muuttamista ja koodin poistamista. Malli analysoi 3700 sovellusta erinäisistä sovelluskaupoista ja saavutti 95,23% tarkkuuden haavoittuvuuksien havaitsemisessa. On identifioitu, että 10,46% sovelluksista sisälsi haavoittuvuuden.

Hybridi-analyysin käyttäminen koneoppimisen kanssa on laajasti käytetty tapa tunnistaa haavoittuvuuksia. Garg ja Baliyan 2019 esitti tavan tunnistaa haavoittuvuuksia käyttäen eriäviä rinnakkaisia luokittelijoita (parralel classifiers). Malli tunnisti zero-day haittaohjelmia, sekä erittäin vaikeasti havaittavia haavoittuvuuksia Androidissa 98,27% tarkkuudella.

Sovelluksien analysointitekniikoita voidaan hyödyntää myös perinteisissä menetelmissä tunnistaa haavoittuvuuksia. Perinteisiin menetelmiin kuuluvat esimerkiksi heuristiset- ja formaalitmenetelmät, sekä menetelmät joissa ei hyödynnetä koneoppimista.



Kuvio 2. Haavoittuvuuksien tunnistamismenetelmien jakauma Senanayake ym. 2023 laatimassa tutkimuksessa.

Haavoittuvuuksien ehkäiseminen kehitysvaiheessa on Senanayake ym. 2023 mukaan tehokkaampaa kuin haavoittuvuuksien ehkäiseminen valmiissa sovelluksessa. Tekniikoita voidaan integroida sovelluskehityksessä kehyksiin (framework), työkaluihin ja lisäosiin, joiden avulla sovelluskehittäjä kykenee tunnistamaan haavoittuvuuksia automaattisesti.

Senanayake ym. 2023 analysoi 118 tarkasti valittua tieteellistä artikkelia, jotka implementoivat edellämainittuja menetelmiä tunnistaa haavoittuvuuksia Androidissa. Tuloksista havaittiin, että sovellusanalysoinnissa 51% analysointitekniikoista oli staattista analysointia. 35% Hybridianalysointia ja loput 14% dynaamista analysointia. Tulokset voidaan havaita kuvaajasta 1. Staattisen analyysin suosio todennäköisimmin johtuu siitä, että analyysitekniikka on hyödyllinen suoraan ohjelmointitasolla. Staattinen analysointi on myös kevyempi, kuin dynaaminen analysointi tai hybridianalysointi. Dynaamisessa- sekä hybridianalysoinnissa tarvitaan lisäresursseja, kuten emulaattoreita, jotta lähdekoodia voidaan ajaa.

Senanayake ym. 2023 laatima tutkimus osoittaa, että koneoppimismenetelmät ovat tutkielman kirjoitusajankohtana suositumpia verrattuna perinteisiin menetelmiin. Kuvaaja 2 havainnollistaa koneoppimismenetelmien ja perinteisten menetelmien suhteen tutkimuksissa. Senanayake ym. 2023 mukaan perinteiset menetelmät olivat kuitenkin suositumpia 2016

edeltävinä vuosina. Koneoppimistekniikoiden kehityksen vuoksi, tutkijat ovat alkaneet hyödyntämään koneoppimista enemmän ja tutkimustulokset ovat olleet positiivisessa kasvussa tarkkuuden, helppokäyttöisyyden sekä skaalautuvuuden kanssa.

5 Yhteenveto

Mobiililaitteiden käytön nopea kasvu on nostanut Android-alustan tietoturvan keskeiseen asemaan, sillä se on maailman käytetyin mobiilialusta. Tietoturvan merkitys korostuu käyttäjien yksityisyyden ja tietojen suojaamisessa sekä laitteiden ja järjestelmien turvallisuuden varmistamisessa. Tietoturvatietoisuuden lisääminen on avainasemassa käyttäjien tietojen suojaamisessa.

Androidin tietoturvaan liittyvät uhat jaetaan pääasiassa kahteen kategoriaan: haavoittuvuudet ja haittaohjelmat. Haavoittuvuudet ovat heikkouksia, joita hyökkääjät voivat käyttää hyväkseen luvattomiin toimiin, kun taas haittaohjelmat ovat ohjelmia, jotka on suunniteltu vahingoittamaan järjestelmiä ja käyttäjiä. Androidin haavoittuvuudet ovat yleisempiä ja vakavampia kuin iOS:n, ja yleisimpiä haavoittuvuustyyppejä ovat palvelunestot, koodin suorittaminen, ylivuoto, oikeuksien saaminen, SQL-injektio, XSS ja hakemiston läpikäynti.

Tietoturvan kehittämiseen keskittyvä tutkimus korostaa tarvetta integroida tietoturvamekanismeja ja -työkaluja sovellusten kehitysvaiheessa. Sovellusten analysointi on prosessi, jossa sovelluksen lähdekoodia ja käyttäytymistä tarkastellaan kolmella eri tekniikalla: staattisella, dynaamisella ja hybridi-analysoinnilla. Staattisessa analysoinnissa tarkastellaan sovelluksen lähdekoodia ja XML-tiedostoja, kun taas dynaaminen analysointi vaatii suoritettavan sovelluksen ja sitä tehdään hiekkalaatikkoympäristössä. Hybridi-analysoinnissa yhdistetään näiden kahden tekniikan vahvuudet.

Koneoppimista, syväoppimista, heuristisia menetelmiä ja formaaleja menetelmiä voidaan hyödyntää yhdessä sovellusanalysoinnin kanssa tunnistamaan haavoittuvuuksia lähdekoodissa. Hybridi-analyysin ja koneoppimisen yhdistäminen on laajasti käytetty tapa tunnistaa haavoittuvuuksia. Sovelluskehityksessä voidaan integroida tekniikoita, kuten frameworkkeja, työkaluja ja lisäosia, joiden avulla sovelluskehittäjä kykenee tunnistamaan haavoittuvuuksia automaattisesti. Tämä on tehokkaampaa kuin haavoittuvuuksien ehkäiseminen valmiissa sovelluksessa.

Erään esimerkin tarjoaa Gajrani ym. 2020 kehittämä Vulvet-framework, joka perustuu staattiseen analysointiin ja pyrkii havaitsemaan ja korjaamaan Android-haavoittuvuuksia. Vulvet

saavutti 95,23% tarkkuuden haavoittuvuuksien havaitsemisessa, kun se analysoi 3700 sovellusta eri sovelluskaupoista. Tämä osoittaa, että lähes 10,46% sovelluksista sisälsi haavoittuvuuksia.

Garg ja Baliyan 2019 esittivät menetelmän, jossa hyödynnettiin hybridi-analyysia ja koneoppimista erilaisten rinnakkaisten luokittelijoiden avulla haavoittuvuuksien tunnistamiseksi. Tämä malli pystyi tunnistamaan niin kutsuttuja "zero-day"haittaohjelmia ja erittäin vaikeasti havaittavia haavoittuvuuksia Androidissa 98,27% tarkkuudella.

Perinteisiä menetelmiä, kuten heuristisia ja formaaleja menetelmiä, voidaan käyttää yhdessä sovellusanalyysin kanssa tunnistamaan haavoittuvuuksia ilman koneoppimista. Nämä menetelmät voivat tarjota lisätietoa ja vahvistaa tuloksia, joita saadaan koneoppimisen ja hybridianalyysin avulla.

Yhteenvetona voidaan todeta, että Android-sovellusten tietoturvan kehittäminen vaatii jatkuvaa huomiota ja kehitystyötä. Analysointitekniikoiden ja menetelmien yhdistäminen sekä niiden integroiminen sovelluskehityksen eri vaiheisiin voi parantaa tietoturvan tasoa ja vähentää haavoittuvuuksia. Käyttäjien tietoisuuden lisääminen tietoturvaan liittyvistä asioista on myös tärkeää, jotta he voivat suojata tietojaan ja laitteitaan tehokkaammin.

Lähteet

CVE-Details. 2022a. *Android : Vulnerability Statistics.* https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224.

———. 2022b. *Iphone Os : Vulnerability Statistics.* https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendor_id=49.

Gajrani, Jyoti, Meenakshi Tripathi, Vijay Laxmi, Gaurav Somani, Akka Zemmari ja Manoj Singh Gaur. 2020. "Vulvet: Vetting of Vulnerabilities in Android Apps to Thwart Exploitation". *Digital Threats* (New York, NY, USA) 1, numero 2 (toukokuu). ISSN: 2692-1626. https://doi.org/10.1145/3376121. https://doi-org.ezproxy.jyu.fi/10.1145/3376121.

Garg, Shivi ja Niyati Baliyan. 2019. "A novel parallel classifier scheme for vulnerability detection in Android". *Computers & Electrical Engineering* 77:12–26. ISSN: 0045-7906. https://doi.org/https://doi.org/10.1016/j.compeleceng.2019.04.019. https://www.sciencedirect.com/science/article/pii/S0045790618320123.

———. 2021. "Comparative analysis of Android and iOS from security viewpoint". *Computer Science Review* 40:100372. ISSN: 1574-0137. https://doi.org/https://doi.org/10.1016/j.cosrev.2021.100372. https://www.sciencedirect.com/science/article/pii/S1574013721000125.

Kouliaridis, Vasileios ja Georgios Kambourakis. 2021. "A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection". *Information* 12 (5). ISSN: 2078-2489. https://doi.org/10.3390/info12050185. https://www.mdpi.com/2078-2489/12/5/185.

Mayrhofer, René, Jeffrey Vander Stoep, Chad Brubaker ja Nick Kralevich. 2021. "The Android Platform Security Model". *ACM Trans. Priv. Secur.* (New York, NY, USA) 24, numero 3 (huhtikuu). ISSN: 2471-2566. https://doi.org/10.1145/3448609. https://doi-org.ezproxy.jyu.fi/10.1145/3448609.

Senanayake, Janaka, Harsha Kalutarage, Mhd Omar Al-Kadri, Andrei Petrovski ja Luca Piras. 2023. “Android Source Code Vulnerability Detection: A Systematic Literature Review”. *ACM Comput. Surv.* (New York, NY, USA) 55, numero 9 (tammikuu). ISSN: 0360-0300. https://doi.org/10.1145/3556974. https://doi-org.ezproxy.jyu.fi/10.1145/3556974.

Statcounter. 2023. *Mobile Operating System Market Share Worldwide,* toukokuu. https://gs.statcounter.com/os-market-share/mobile/worldwide/.

Statista. 2023. *Number of smartphone users worldwide.* https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.