TURUN YLIOPISTO

Yhdistelmäsyväoppimismenetelmien tehokkuus haittaohjelmien tunnistamisessa Windows-ympäristössä

Tieto- ja viestintätekniikan tutkinto-ohjelma
Tietotekniikan laitos, Teknillinen tiedekunta
TkK -tutkielma

Taru Kylä-Kaila
Kesäkuu 2025

TkK-tutkielma

Turun yliopisto

Tietotekniikan laitos, Teknillinen tiedekunta

Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Tekijä: Taru Kylä-Kaila
Otsikko: Yhdistelmäsyväoppimismenetelmien tehokkuus haittaohjelmien tunnistamisessa Windows-ympäristössä.
Sivumäärä: 28 sivua
Päivämäärä: Kesäkuu 2025

Haittaohjelmat ovat luvattomiin toimintoihin suunniteltuja ohjelmistoja, joiden tarkoituksena on vaarantaa järjestelmien turvallisuus. Niistä on muodostunut yksi merkittävimmistä nykypäivän kyberturvallisuusuhista, minkä vuoksi tehokkaiden ja luotettavien tunnistusmenetelmien kehittäminen on entistä kriittisempää. Syväoppimismenetelmät tarjoavat lupaavan, automaattiseen analyysiin perustuvan lähestymistavan haittaohjelmien havaitsemiseen.

Tässä tutkielmassa tarkastellaan Windows-ympäristöön kohdistuvien haittaohjelmien tunnistamista erilaisten yhdistelmäsyväoppimismallien avulla. Tutkimus toteutettiin kirjallisuuskatsauksena analysoimalla seitsemää tieteellistä artikkelia, joissa kehitettiin syväoppimisalgoritmeja yhdistävä malli haittaohjelmien tunnistukseen. Nämä yhdistelmämallit hyödyntävät konvoluutioneuroverkkojen lisäksi yhtä tai useampaa eri syväoppimisarkkitehtuuria. Mallien tehokkuutta haittaohjelmien tunnistuksessa arvioidaan tarkkuuden, täsmällisyyden, sensitiivisyyden ja F1-arvon avulla.

Yhdistelmäsyväoppimismallit saavuttivat haittaohjelmien tunnistuksessa parhaimmillaan yli 99 %:n tarkkuuden. Tarkkuuden lisäksi myös täsmällisyys, sensitiivisyys ja F1-arvot pysyivät korkeina, mikä osoittaa mallien tasapainoisen ja luotettavan suorituskyvyn. Vaikka tulosten välillä havaittiin vaihtelua, voidaan syväoppimismalleja pitää suorituskykymittareiden perusteella arvioituna tehokkaana ratkaisuna haittaohjelmien tunnistuksessa. Laajojen ja monipuolisten aineistojen avulla koulutetut syväoppimismallit ylsivät pääsääntöisesti parempiin tuloksiin. Myös yksittäisiin syvä- ja koneoppimismenetelmiin verrattuna yhdistelmämallit suoriutuivat tunnistustarkkuudeltaan paremmin.

Yhdistelmäsyväoppimismalleja voidaan pitää suorituskykymittareiden perusteella tehokkaana ja lupaavana ratkaisuna haittaohjelmien tunnistuksessa. Haasteena on kuitenkin edelleen erityisesti uusien ja muuntautuvien haittaohjelmien tunnistus, sekä mallien sovellettavuus todelliseen ympäristöön. Lisäksi mallien suorituskyvyn arviointia vaikeuttaa osaltaan epäyhtenäinen raportointitapa käytettyjen mittareiden osalta. Jatkotutkimuksessa tulisi keskittyä erityisesti mallien testaamiseen realistisissa ja epätasapainoisissa ympäristöissä, yleistettävyyden arviointiin sekä läpinäkyvään suorituskyvyn vertailuun.

Asiasanat: haittaohjelma, syväoppiminen, Windows, tarkkuus, täsmällisyys, sensitiivisyys, F1-arvo.

Sisällysluettelo

1 Johdanto

Haittaohjelmat tai haitalliset ohjelmistot ovat tietoisesti kehitettyjä ohjelmia, joiden tavoitteena on suorittaa luvattomia toimintoja tai vaarantaa järjestelmien turvallisuus. Digitalisaation myötä haittaohjelmat leviävät ja muuntuvat yhä nopeammin, mikä tekee niistä yhden merkittävimmistä kyberturvallisuusuhista nykypäivänä.[1] Erilaisia haittaohjelmia tunnetaan useita, kuten troijalaiset, madot, virukset ja kiristysohjelmat. Näiden avulla kyberrikolliset pyrkivät usein hyötymään tilanteesta taloudellisesti, vahingoittamaan kriittisiä infrastruktuureja tai varastamaan arkaluontoisia tietoja.[2] Haittaohjelmien määrä ja niiden aiheuttamat vahingot ovat kasvaneet merkittävästi viimeisen vuosikymmenen aikana. Pelkästään vuonna 2024 havaittiin yli 80 miljoonaa uuttaa haittaohjelmaa, joista suurin osa kohdistui Windows-käyttöjärjestelmään. Lisäksi päivittäin havaitaan keskimäärin 450 000 uutta haittaohjelmaa tai ei-toivottua sovellusta, mikä kuvastaa haittaohjelmien nopeaa kehitystä ja leviämistä.[3]

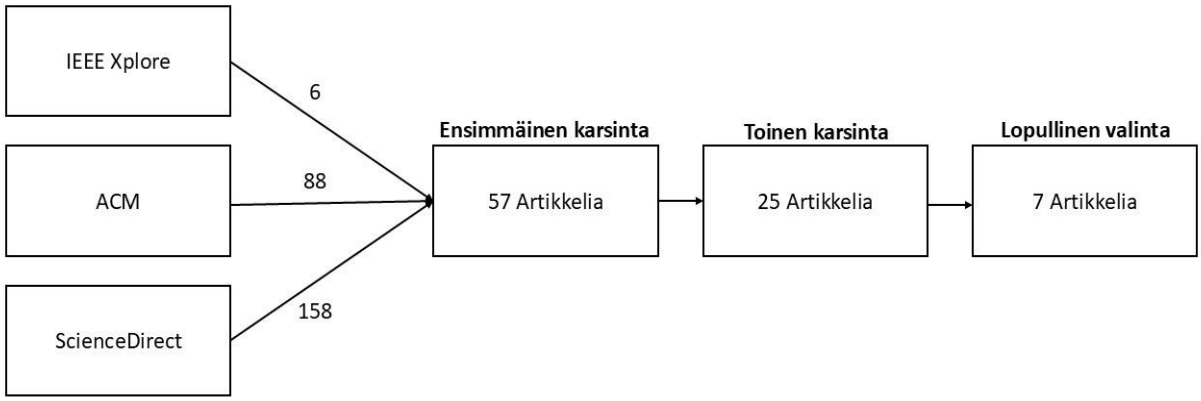
Uusien ja tehokkaiden tunnistusmenetelmien kehittäminen on entistä tärkeämpää, jotta haittaohjelmat havaitaan ennen kuin ne ehtivät aiheuttaa vahinkoa käyttäjille tai järjestelmille. Viime vuosina tekoälyn kasvava suosio on laajentanut sen sovellusmahdollisuuksia myös haittaohjelmien tunnistukseen. Erityisesti koneoppimispohjaiset menetelmät ovat yleistyneet, mutta ne edellyttävät usein manuaalista piirteiden valintaa, mikä hidastaa tunnistusprosessia ja lisää virhealttiutta. Tähän haasteeseen on esitetty ratkaisuksi syväoppimismalleja, jotka oppivat erottamaan automaattisesti haitalliset näytteet vaarattomista. Varsinkin syväoppimismallit, jotka yhdistävät eri syväoppimisalgoritmeja, ovat saavuttaneet erinomaisia tuloksia haittaohjelmien tunnistuksessa. Parhaimmillaan nämä mallit yltävät jopa 99,7 % :n tunnistustarkkuuteen.[4],[5]

Tutkielma toteutettiin kirjallisuuskatsauksena, jonka tavoitteena on arvioida eri syväoppimisalgoritmeja yhdistävien syväoppimismallien tehokkuutta haittaohjelmien tunnistuksessa erilaisten suorituskykymittareiden avulla arvioituna. Tarkasteltavat yhdistelmämallit hyödyntävät konvoluutioneuroverkkojen lisäksi yhtä tai useampaa eri syväoppimisalgoritmia. Tutkielmassa käsitellyt syväoppimiskehykset ovat suunniteltu erityisesti Windows-ympäristöön. Tutkimuskysymyksiksi muodostuivat seuraavat:

TK1. Miten syväoppimisalgoritmien yhdistelmiä voidaan hyödyntää haittaohjelmien tunnistuksessa Windows-ympäristössä?

TK2. Miten tehokkaasti konvoluutioneuroverkkopohjaiset yhdistelmäsyväoppimismallit tunnistavat haittaohjelmat erilaisten mittareiden avulla arvioituna?

Tiedonhaku toteutettiin käyttämällä kolmea alan keskeistä tietokantaa: IEEE Xplore, ScienceDirect ja ACM (*eng.* Association for Computing Machinery). Hakulauseeksi muodostui "*Malware*" AND ("Hybrid" AND ("*Deep Learning*" OR "*DL*")) AND "*detection*" AND "*Windows*" NOT "*Android*" NOT "*IoT*". Tarkastelun ajanjaksoksi rajattiin 2022–2024, jotta tutkielmaan valikoituisi mahdollisimman ajankohtaista tietoa. Tiedonhakuprosessin vaiheet esitetään kuvassa 1. Ensimmäisessä vaiheessa otsikkoseulonnan perusteella tarkasteluun valittiin 57 artikkelia. Tässä vaiheessa rajattiin pois kaikki artikkelit, jotka eivät käsitelleet haittaohjelmien tunnistamista ja syväoppimismenetelmien yhdistelmiä otsikossa. Toisessa vaiheessa artikkelien määrä rajattiin tiivistelmän perusteella 25:een. Syvempään analyysiin valittiin seitsemän tutkimusartikkelia, joissa esiteltiin yhdistelmäsyväoppimismalli haittaohjelmien tunnistukseen ja raportoitiin mallin suorituskyvystä.



Kuva 1. Tiedonhakuprosessin eteneminen vaiheittain.

Tutkielman luvussa kaksi määritellään haittaohjelmat, niiden tyypit sekä keinot, joilla haittaohjelmat yrittävät välttää tunnistuksen. Lisäksi luvussa määritellään työssä tarkastellut syväoppimisalgoritmit. Luvussa kolme perehdytään haittaohjelmien tunnistuksessa käytettäviin analyysimenetelmiin, sekä esitellään tutkielmassa tarkasteltavat yhdistelmäsyväoppimismallit ja niiden hyödyntäminen. Luvussa neljä määritellään arvioinnissa käytetyt suorituskykymittarit, arvioidaan syväoppimismallien tehokkuutta niiden avulla, sekä tehdään johtopäätökset menetelmien tehokkuudesta. Lopulta luvussa viisi kootaan tutkielma yhteen.

2 Haittaohjelmien ja syväoppimismenetelmien määrittely

Tässä luvussa keskitytään haittaohjelmien ja syväoppimisen määrittelyyn. Käsitellään ensin eri haittaohjelmatyyppien toiminnallisuudet sekä keinot, joilla haittaohjelmat yrittävät välttää tunnistuksen. Tämän jälkeen määritellään tutkielmassa käytetyt syväoppimisalgoritmit.

2.1 Haittaohjelmat

Haittaohjelmalla tai haitallisella koodilla tarkoitetaan pahantahtoista ohjelmisto- tai laiteohjelmistopohjaista koodia, joka on suunniteltu suorittamaan luvattomia toimintoja ja vahingoittamaan järjestelmän luottamuksellisuutta, eheyttä tai saatavuutta. Yleisesti haittaohjelmien määritelmät sisältävät kaksi keskeistä piirrettä: haitallisuus ja vahingollisuus sekä kyky toimia ilman käyttäjän ja järjestelmänvalvojan suostumusta tai tietoisuutta. Haittaohjelmat voivat kaapata järjestelmiä ja verkkoja, varastaa luottamuksellisia tietoja, muodostaa etäyhteyksiä tartunnan saaneisiin verkkoihin tai asentaa haitallista ohjelmistoa uhrin laitteille ilman käyttäjän tai järjestelmänvalvojan lupaa. Nämä haitalliset toiminnot voivat kohdistua niin yksityishenkilöihin kuin suuriin organisaatioihin.[1], [6]

Haittaohjelmat vaarantavat tietoturvan perusperiaatteet – luottamuksellisuuden, eheyden ja saatavuuden – hyödyntämällä olemassa olevia haavoittuvuuksia tai luomalla uusia. Käyttäjä olettaa tietokonejärjestelmän olevan turvallinen näiden kolmen periaatteen osalta. Toisin sanoen käyttäjä luottaa, että järjestelmä on aina käytettävissä, sen sisältämien tietojen eheys on säilytetty, eikä järjestelmään tallennettu tieto päädy asiattomien osapuolten haltuun. Haitalliset toiminnot, kuten arkaluontoisten tietojen varastaminen näppäintallentajien avulla, IP-osoitteiden muokkaaminen tai järjestelmäresurssien ylikuormitus vaarantavat nämä tietoturvaperiaatteet.[7] Tarkastellaan seuraavaksi haittaohjelmia tarkemmin luokittelemalla ne eri kategorioihin niiden toimintatavan mukaan. Haittaohjelmatyypit ja niiden tiivistetyt kuvaukset esitetään taulukossa 1 (sivu 5).

Tietokonevirus on haittaohjelmia tai haitallinen koodi, joka liitetään toiseen ohjelmaan tai tiedostoon. Sille ominaista on, että se kykenee monistumaan ja leviämään tietokoneesta toiseen isännän eli suoritettavan tiedoston mukana. Käyttäjä voi avata vaarallisen sähköpostiliitteen, jolloin virus aktivoituu ja tartuttaa tietokoneen. Virusten aiheuttamia vahinkoja ovat muun muassa tietojen tuhoaminen ja vahingoittaminen, järjestelmäresurssien kuluttaminen sekä näppäinpainallusten tallentaminen.[7], [8]

Troijalainen on haittaohjelmatyyppi, joka naamioituu vaarattomaksi tiedostoksi tai ohjelmaksi. Se leviää tyypillisesti sähköpostiliitteiden tai ilmaisohjelmien mukana, jolloin käyttäjä lataa ja asentaa sen tietämättään. Tämä haittaohjelmatyyppi antaa hyökkääjälle pääsyn käyttöjärjestelmään ja mahdollistaa arkaluontoisten tietojen, kuten salasanojen tai pankkitietojen varastamisen tai käyttäjän verkkoliikenteen seuraamisen.[7],[8]

Mato on puolestaan troijalaisen alalaji, joka pystyy monistumaan ja leviämään itsestään tietoverkkojen avulla. Madot voivat levitä erittäin nopeasti ja aiheuttaa merkittävää haittaa, sillä ne eivät tarvitse käyttäjän toimenpiteitä tarttuakseen uusiin järjestelmiin. Ne voivat tukkia verkkoa ja hidastaa järjestelmän toimintaa monistumalla hallitsemattomasti. Madot leviävät tyypillisesti paikallisverkon tai internet-yhteyden kautta ja ne pystytään yleensä pysäyttämään päivittämällä järjestelmät tai ohjelmistot. [7],[8]

Vakoiluohjelma on haittaohjelma, joka tunkeutuu tietokonejärjestelmään, kerää tietoa käyttäjästä ja lähettää sen luvatta kolmannelle osapuolelle. Tiedot voivat olla henkilökohtaisia tietoja, verkkoselailuhistoriaa ja kirjautumistietoja. Vakoiluohjelma voi olla myös laillinen ohjelmisto, joka kerää tietoja käyttäjästä mainontaa tai kaupallisia tarkoituksia varten. Vakoiluohjelmat eivät välttämättä ole haitallisia mutta voivat vaarantaa käyttäjän yksityisyyden. Haitalliset vakoiluohjelmat puolestaan ovat suunniteltu hyödyntämään varastettuja tietoja taloudellisen hyödyn saamiseksi. Tämä voi johtaa tietomurtoihin tai henkilökohtaisten tietojen väärinkäyttöön. Lisäksi vakoiluohjelmat heikentävät tietokoneiden ja verkkojen suorituskykyä, koska ne hidastavat järjestelmiä ja kuluttavat resursseja.[8]

Botti (*eng*. Bot) on haittaohjelmatyyppi, joka antaa hyökkääjälle mahdollisuuden hallita käyttäjän järjestelmää tai suorittaa toimintoja ilman käyttäjän tietämystä. Bottiohjelmat voivat vierailla verkkosivuilla, levittää haittaohjelmia muihin järjestelmiin tai lähettää pyyntöjä verkkopalvelimille. Bottien ohjaaminen tapahtuu komentopalvelimelta (*eng.* Command and Control, C&C) ja yhdessä useiden kaapattujen laitteiden kanssa ne voivat muodostaa bottiverkon (*eng.* Botnet). Bottiverkkoja voidaan käyttää erityisesti hajautettuihin palvelunestohyökkäyksiin tai kryptovaluutan louhintaan. Palvelunestohyökkäyksissä tuhannet tai miljoonat tartunnan saaneet laitteet kuormittavat kohdepalvelinta lähettämällä valtavan määrän pyyntöjä samanaikaisesti. Tavoitteena on kaataa verkkosivusto tai estää laillisten käyttäjien pääsy sivustoille.[7], [8]

Kryptohaittaohjelmat mahdollistavat kryptovaluutan louhimisen hyökkääjälle ilman uhrin suostumusta. Uhrin laitteiden laskentatehoa hyödynnetään luvatta kryptovaluutan

louhimiseen. Tällaisen hyökkäyksen seurauksena uhrille koituu haittoja, kuten lisääntynyt sähkönkulutus ja laskentatehon heikkeneminen, kun taas hyökkääjä hyötyy kaikista louhituista kryptovaluutoista.[8]

Kiristyshaittaohjelma *(eng.* Ransomware) on kyberrikollisten käyttämä hyökkäystapa, jossa kohteen tietoja lukitaan, salataan, poistetaan tai varastetaan ja vaaditaan lunnaita näiden tietojen palauttamiseksi. Lunnaat vaaditaan yleensä virtuaalivaluuttoina kuten Bitcoineina, jolloin rikollisten jäljittäminen on usein vaikeampaa.[1], [9], [10]

APT-hyökkäykset (*eng.* advanced persistent threat) ovat jatkuvia, huomaamattomia ja monimutkaisia hyökkäysmenetelmiä, joiden avulla pyritään murtautumaan järjestelmiin ja pysymään niissä pitkään. Nämä hyökkäykset hyödyntävät järjestelmien vakavia haavoittuvuuksia ja niiden kohdistaminen yrityksiin, instituutioihin ja arvokkaisiin tietojärjestelmiin tekee niiden torjunnasta erityisen haastavaa.[2]

Tiedostoton haittaohjelma on muistissa toimiva haittaohjelma, joka ei tallennu kiintolevylle, vaan toimii suoraan tietokoneen järjestelmämuistissa (*eng.* Random-access memory, RAM). Tämä tekee sen havaitsemisesta vaikeampaa perinteisillä virustorjuntamenetelmillä. Lisäksi haittaohjelmatyypin havaitsemista vaikeuttaa tartunnan katoaminen, kun tietokone käynnistetään uudelleen.[8]

Taulukko 1. Haittaohjelmatyypit ja niiden kuvaukset.

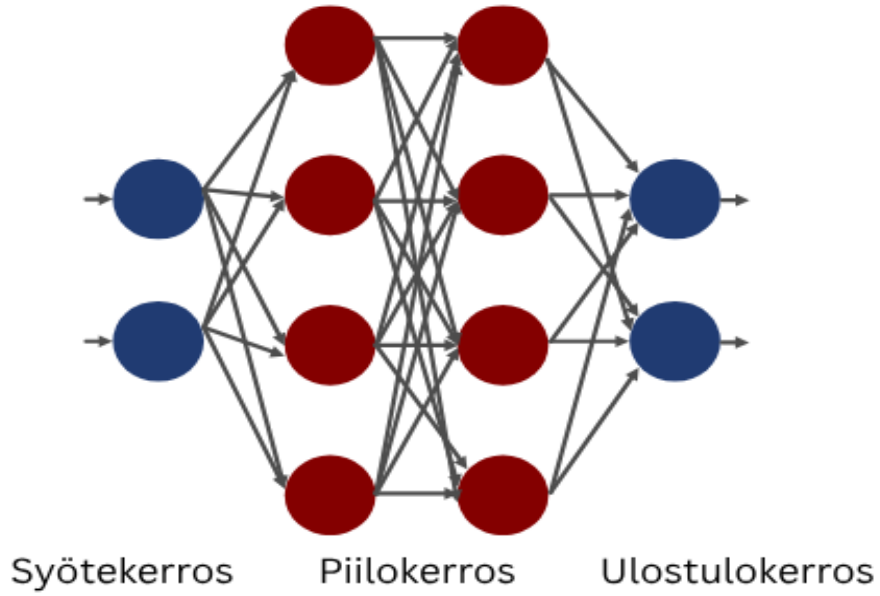
Haittaohjelmatyyppi	**Kuvaus**
Virus	Monistuva haittaohjelma, joka leviää isäntätiedoston mukana.
Troijalainen	Naamioituu vaarattomaksi tiedostoksi tai ohjelmaksi ja huijaa käyttäjää lataamaan tai asentamaan sen.
Mato	Monistuu ja leviää itsenäisesti verkon kautta ilman käyttäjän toimia
Vakoiluohjelma	Kerää tietoa käyttäjästä ja lähettää sen luvatta eteenpäin.
Piilohallintaohjelma	Mahdollistaa järjestelmän hallinnan etänä käyttäjältä salaa.
Botti	Antaa hyökkääjälle pääsyn järjestelmään, yleensä osana bottiverkkoa.
Kryptohaittaohjelma	Hyödyntää laitteen resursseja kryptovaluutan louhintaan ilman lupaa.
Kiristyshaittaohjelma	Varastaa tai salaa tietoja ja vaatii lunnaita niiden palauttamiseksi.
APT-hyökkäykset	Pitkäkestoinen, kohdistettu hyökkäys arvokkaisiin kohteisiin.
Tiedostoton haittaohjelma	Tietokoneen järjestelmämuistissa toimiva haittaohjelma.

Pahantahtoiset toimijat käyttävät erilaisia keinoja piilottaakseen haittaohjelman todellisen tarkoituksen. Tätä kutsutaan koodin hämäännyttämiseksi tai hämärtämiseksi (*eng.* Code Obfuscation), jossa haittaohjelmakoodin kirjoittaja muokkaa koodia niin, että sen haitallinen tarkoitus peittyy ja koodin analysointi vaikeutuu [11]. Perinteiset virustorjuntaohjelmat etsivät haittaohjelmia vertaamalla tiedostoja tunnettuihin haittakoodin malleihin. Koodin hämärtämisen avulla haittaohjelmien tekijät hankaloittavat näiden tunnisteiden poimimista binäärikoodista.[7], [12]

Uuden sukupolven haittaohjelmat käyttävät erilaisia hämärtämistekniikoita, kuten polymorfismia tai metamorfismia. Perinteisesti haittaohjelmat tunnistetaan niiden binäärikoodin tai merkkijonojen avulla. Polymorfiset haittaohjelmat pystyvät piilottamaan nämä ominaisuudet muuttamalla haittaohjelman binääridatan satunnaiseksi. Polymorfinen haittaohjelma salaa itsensä erilaisten salausavaimien avulla jokaisessa tartunnassa uudelleen, mutta itse koodin haitallinen osa pysyy samanlaisena. Metamorfiset haittaohjelmat puolestaan pystyvät muokkaamaan binäärikoodiaan aina, kun ne luovat uuden kopion itsestään. Jokainen haittaohjelman kopio näyttää erilaiselta kuin edellinen mutta suorittaa silti saman haitallisen toiminnon. Monimuotoiset haittaohjelmat voivat myös käyttää useita hämärtämistekniikoita samanaikaisesti, mikä tekee niiden havaitsemisesta entistä haastavampaa.[7], [12]

2.2 Syväoppimismenetelmät

Syväoppiminen on koneoppimisen osa-alue, joka hyödyntää syviä keinotekoisia neuroverkkoarkkitehtuureja monimutkaisten ilmiöiden mallintamiseen ilman manuaalista piirrevalintaa. Viime vuosina syväoppimisesta on tullut keskeinen työkalu esimerkiksi kuvantunnistuksessa, poikkeavuuksien havaitsemisessa tai haittaohjelmien tunnistuksessa. Keinotekoiset neuroverkot koostuvat tyypillisesti kolmesta pääkerroksesta: syötekerroksesta, yhdestä tai useammasta piilokerroksesta sekä ulostulokerroksesta (Kuva 2.1). Syötekerros vastaanottaa raakadataa, kuten kuvan pikseliarvoja, tekstidataa tai verkkoliikenteen ominaisuuksia. Varsinainen tiedonkäsittely ja oppiminen tapahtuvat piilokerroksissa, joiden määrä ja rakenne voivat vaihdella sovelluksen mukaan. Mitä syvempi verkko on, sitä hienojakoisempia ja monimutkaisempia piirteitä se kykenee havaitsemaan. Ulostulokerros tuottaa lopullisen ennusteen tai luokituksen mallin oppiman tiedon perusteella.[13], [14], [15], [16]



Kuva 2. Keinotekoisten neuroverkkojen kolme kerrosta: syötekerros, piilokerros ja ulostulokerros. Mukaillen lähteestä [14].

Verkko oppii automaattisesti kokeilemalla ja korjaamalla itseään säätämällä sisäisiä painokertoimia harjoitusaineistosta lasketun virheen perusteella. Syväoppimismallien hierarkkinen rakenne mahdollistaa eritasoisten piirteiden oppimisen. Alemmat kerrokset oppivat tunnistamaan yksinkertaisia matalan tason ominaisuuksia, kuten viivoja ja tekstuureja, kun taas ylemmät kerrokset yhdistelevät ja tulkitsevat näitä piirteitä muodostaen korkeamman tason käsitteitä, kuten muotoja, rakenteita tai semanttisia kokonaisuuksia. Erilaisia neuroverkkoihin perustuvia arkkitehtuureja ovat syvä neuroverkko (*eng.* Deep Neural Network, DNN), konvoluutioneuroverkot (*eng.* Convolutional Neural Network, CNN), pitkäkestoinen-lyhytkestomuisti (*eng.* Long Short-Term Memory, LSTM) sekä erilaiset transformer-pohjaiset mallit. Käsitellään näitä malleja seuraavaksi tarkemmin.[13], [14], [15], [16]

DNN on keinotekoinen syvä neuroverkko, joka sisältää useita piilokerroksia syötekerroksen ja ulostulokerroksen välillä. Piilokerrokset mahdollistavat hierarkkisen piirreoppimisen, jossa matalan tason piirteistä muokataan korkeatasoisia esityksiä. Syvyyden lisääminen antaa mallille kyvyn oppia yhä monimutkaisempia rakenteita, mutta samalla se vaatii tehokkaita optimointi- ja säännöstelymenetelmiä estämään ylisovittamista ja oppimisen hajoamista.[16]

CNN on syviin neuroverkkoihin pohjautuva syväoppimisalgoritmi, joka kehitettiin alun perin kuvantunnistukseen. Sen käyttöä on myöhemmin laajennettu esimerkiksi tekstin luokitteluun ja haittaohjelmien tunnistukseen. CNN-mallit rakentuvat useista erikoistuneista kerroksista,

jotka muokkaavat syötettä kerros kerrokselta. Niiden toiminta perustuu neljään keskeiseen periaatteeseen: paikallisiin yhteyksiin, jaettuihin painoihin, poolaukseen ja monikerroksisuuteen. CNN-mallit koostuvat konvoluutiokerroksista, joissa pienet suodattimet liikkuvat syötteen yli etsien toistuvia kuvioita, kuten esimerkiksi reunoja tai tietyn kutsujoukon havaitsemista haittaohjelma-analyysissä. Nämä suodattimet jakavat samat painot eri kohdissa syötettä. Poolauskerrokset (*eng.* Pooling layer) pienentävät tietomäärää säilyttäen tärkeimmät piirteet, mikä tekee menetelmästä vähemmän herkän sijainnin muutoksille. Syvemmät verkot kykenevät tunnistamaan yhä monimutkaisempia rakenteita. CNN-verkkojen tehokkuus perustuu siihen, että ne hyödyntävät datan paikallista rakennetta ja toistuvia kuvioita. Verkon lopussa täysin kytketyt kerrokset (*eng.* Fully connected layers) yhdistävät opitut piirteet ja suorittavat lopullisen luokittelun tai ennusteen. Erilaisia CNN-arkkitehtuureja ovat muun muassa ResNet ja DenseNet.[14],[15], [17]

LSTM on parannettu toistuvien neuroverkkojen arkkitehtuuri, joka on suunniteltu säilyttämään ja hyödyntämään pitkän aikavälin riippuvuuksia sekventiaalisessa datassa, kuten tekstissä tai aikasarjoissa. LSTM-käyttää erityisiä muistiyksiköitä, joita kutsutaan solutiloiksi (*eng*. cell state), sekä kolmea porttia: syöttöportti, unohdusportti ja ulostuloportti. Syöttöportin avulla määritetään mitä uutta tietoa tallennetaan solutilaan. Unohdusportti päättää, mitä tietoa aiemmasta muistista unohdetaan ja ulostuloportti kontrolloi mitä tietoa siirretään seuraavaan aikaväliin. Nämä portit mahdollistavat sen, että LSTM kykenee oppimaan mitkä tiedot ovat olennaisia tallentaa tai unohtaa eri vaiheissa sekvenssiä. LSTM on osoittautunut erittäin tehokkaaksi esimerkiksi puheentunnistuksessa, konekäännöksessä ja haittaohjelmien tunnistuksessa, koska se kykenee käsittelemään pitkiä riippuvuuksia ilman gradientin häviämistä.[14], [17]

GRU (*eng.* Gated Recurrent Unit) on myös toistuvien neuroverkkojen arkkitehtuuri, joka käsittelee sekventiaalista dataa käyttämällä erikoistuneita porttimenetelmiä. Mallit ovat luotu ymmärtämään ja hyödyntämään aikajärjestyksen rakenteita. GRU-mallissa on vähemmän portteja LSTM-malliin verrattuna, mikä vähentää laskennallisia vaatimuksia sekä mallin koulutusaikaa. BiGRU (*eng.* Bidirectional Gated Recurrent Unit) on kaksisuuntainen versio, jossa sekvenssin tietoa käsitellään sekä eteen-, että taaksepäin.[14]

Luonnollisen kielen käsittely (*eng.* Natural Language Processing, NLP) on tekoälyn osa-alue, joka keskittyy ihmisten käyttämän kielen ymmärtämiseen, tulkintaan ja tuottamiseen koneellisesti. NLP:n sovelluksia ovat esimerkiksi konekäännös, tekstin luokittelu, ja

puheentunnistus. Tavoitteena on mahdollistaa vuorovaikutus ihmisen ja koneen välillä luonnollisella kielellä. Transformer-pohjaiset mallit mullistivat luonnollisen kielen käsittelyn ja ovat sittemmin muodostuneet perustaksi BERT (*eng*. Bidirectional Encoder Representations from Transformers) - ja GPT-malleille. Toisin kuin LSTM tai konvoluutioneuroverkot transformer-mallit hyödyntävät pelkästään itsetarkkailumekanismeja tarjotakseen enemmän rinnakkaisuutta ja pidemmän aikavälin riippuvuuksien oppimisen.[14], [18], [19]

BERT (*eng*. Bidirectional Encoder Representations from Transformers) on transformer-pohjainen syväoppimisalgoritmi, joka on esikoulutettu suurella tekstimassalla ymmärtämään syvällisempiä yhteyksiä syötteissä. BERT-malli osaa säilyttää sanayhteydet ja kontekstin, minkä vuoksi sitä hyödynnetään erityisesti tekstin semanttisessa tulkinnassa. BERT-mallin hyödyntäminen haittaohjelmien tunnistuksessa perustuu sen kykyyn käsitellä esimerkiksi API-kutsujonoja (*eng.* Application Programming Interface Calls) samalla tavalla kuin luonnollista kieltä**.**[20], [19]

Haitallinen toiminta voi ilmetä peräkkäisinä API-kutsuina, kuten OpenProcess, ReadFile ja WriteFile. Yksittäin kutsut ovat tavallisia ja laillisia, mutta tietyssä kontekstissa ja järjestyksessä ne voivat viitata haitalliseen toimintaan. Jotta BERT-malli pystyy käsittelemään API-kutsuja luonnollisen kielen tavoin, kutsut muunnetaan vektoriesityksiksi kolmen upotuksen avulla. Tunnisteen upotus (*eng.* token embedding) kuvaa kutsun merkityksen, segmentin upotus (*eng*. segment embedding) erottaa mahdolliset osajaksot ja paikan upotus (*eng.* position embedding) ilmoittaa kutsun sijainnin. Transformer-arkkitehtuuri analysoi näitä piirteitä tunnistaakseen kutsujen välisiä suhteita ja rakenteita. Lopuksi BERT tiivistää koko sekvenssin yhdeksi edustukseksi ja luokittelee, onko kyseessä haitallinen vai hyväntahtoinen toiminta. Mallin kaksisuuntainen rakenne mahdollistaa sekä edeltävien että seuraavien kutsujen huomioimisen, mikä parantaa haitallisten toimintamallien tunnistamista.[20] [19]

3 Haitallisen toiminnallisuuden tunnistaminen

Haitallisen toiminnallisuuden tunnistaminen tarkoittaa ohjelman käyttäytymisen ja sisällön tutkimista, jotta voidaan arvioida, onko ohjelma hyväntahtoinen vai haitallinen. Tunnistusprosessi koostuu kolmesta vaiheesta. Ensimmäisenä analysoidaan haittaohjelman toimintaa ja sen aiheuttamia vahinkoja. Seuraavaksi haittaohjelmasta kerätään haitalliseen toimintaan viittaavat käyttäytymismallit ja tekniset ominaisuudet. Lopulta ohjelma luokitellaan kerättyjen tietojen perusteella joko haitalliseksi tai hyväntahtoiseksi.[2] Seuraavissa alaluvuissa käsitellään yleisimmät haittaohjelmien analyysimenetelmät ja tarkastellaan, miten syväoppimismenetelmiä on hyödynnetty haittaohjelmien tunnistuksessa.

3.1 Haittaohjelmien analyysimenetelmät

Haittaohjelmien analyysi tarkoittaa haitallisen ohjelmiston toimintatapojen, alkuperän ja mahdollisten vaikutusten tutkimista. Haittaohjelmien kehittyvän ja muuttuvan luonteen vuoksi haittaohjelmien analyysi on tärkeässä roolissa parempien tunnistusmenetelmien kehittämiseksi. Analyysimenetelmät jaetaan kolmeen pääkategoriaan: staattinen, dynaaminen ja muistianalyysi. Jokainen menetelmä tarkastelee haittaohjelmia eri näkökulmasta ja hyödyntää erilaisia ominaisuuksia analyysissä. Hybridianalyysi yhdistää näitä menetelmiä, minkä ansiosta sitä pidetään usein tehokkaampana kuin yksittäisiä lähestymistapoja.[12], [21], [22]

Staattisessa analyysissä haitallisen tiedoston ominaisuuksia tutkitaan ilman sen suorittamista. Windows-ympäristössä staattinen analyysi keskittyy usein PE-tiedostojen (*eng.* Portable Executable) rakenteen ja sisällön tarkasteluun. PE-tiedosto on Windows-käyttöjärjestelmän suoritettavan ohjelmakoodin tiedostomuoto. Se koostuu otsikoista ja osioista, jotka määrittelevät muun muassa ohjelmakoodin, tietorakenteet ja kirjastoviittaukset. Monet haittaohjelmat ovat PE-muotoisia suoritettavia tiedostoja (EXE) tai dynaamisia kirjastoja (DLL), joiden rakenteellisia ominaisuuksia voidaan hyödyntää tunnistuksessa.[23]

Staattisen analyysin aikana haittaohjelmatiedostosta poimitaan erilaisia piirteitä, kuten operaatiokoodeja, API-kutsuja, hajautusarvoja tai tiedoston metatietoja. Näitä tietoja hyödynnetään tiedoston luokittelussa joko haitalliseksi tai vaarattomaksi. Staattisen analyysin tavoitteena on löytää haittaohjelmakoodista ominaisuuksia, jotka erottavat haitalliset ohjelmat hyväntahtoisista. Menetelmää pidetään nopeana ja tehokkaana, sillä tiedostoa ei tarvitse

suorittaa. Sen avulla tunnistetaan kuitenkin huonosti haittaohjelmia, jotka käyttävät erilaisia hämärrystekniikoita.[2], [9], [14], [21]

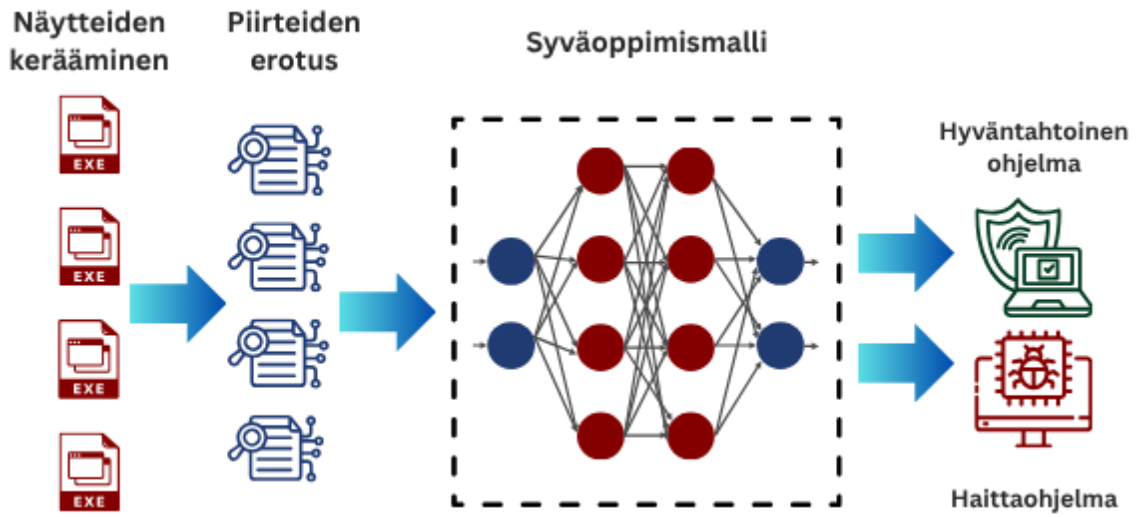
Dynaaminen analyysi puolestaan perustuu haittaohjelman tarkkailuun ohjelman suorittamisen aikana. Tätä varten käytetään hiekkalaatikkoympäristöä, jossa haittaohjelman käyttäytymistä voidaan seurata turvallisesti suorituksen aikana. API-kutsut ovat keskeisessä roolissa haittaohjelma-analyysissä, sillä ne mahdollistavat monia perustoimintoja, kuten tiedostojen luomisen, avaamisen ja sulkemisen, rekisterimuutokset sekä viestien näkymisen näytöllä. Jokainen Windows-pohjainen sovellus toimii kutsumalla API-funktioita. Tämä ohjelmointirajapinta mahdollistaa kuitenkin API-kutsujen hyödyntämisen myös haitalliseen toimintaan, mikä tekee kutsujen seurannasta tehokkaan keinon tarkkailla, onko suoritettava ohjelma haitallinen. Dynaamisen analyysin etuna on, että se pystyy tunnistamaan myös hämärrettyjä haittaohjelmia. Uudet monimuotoiset ja nopeasti kehittyvät haittaohjelmat saattavat kuitenkin tunnistaa hiekkalaatikkoympäristön ja piilottaa toimintansa.[2], [14] , [4], [24]

Staattinen ja dynaaminen analyysi eivät aina kykene tunnistamaan haittaohjelmia, jotka suorittavat haitalliset toiminnot suoraan järjestelmämuistissa. Esimerkiksi vuonna 2023 Mockingjay-hyökkäys injektoi haitallisen hyötykuorman suoraan RAM-muistiin, minkä avulla se onnistui välttämään monet antivirus ja EDR-työkalut (*eng.* Endpoint Detection and Response). Sen sijaan muistianalyysi keskittyy tietokoneen RAM-muistin käsittelyyn. Se soveltuu erityisen hyvin monimutkaisten tai hämärrettyjen haittaohjelmien tunnistamiseen. RAM-muistiin jää tietoja aktiivisista ja päättyneistä prosesseista, rekisterioperaatioista sekä API-kutsuista. Muistianalyysissä muistin sisältö tallennetaan muistivedoksiin, joita analysoimalla voidaan tunnistaa haittaohjelmien toiminta. Tämän menetelmän etuna on, että se tunnistaa reaaliaikaisesti haittaohjelmien toimintoja muistissa.[21], [25]

3.2 Syväoppimismenetelmät haittaohjelmien tunnistuksessa

Syväoppimismenetelmien hyödyntämistä haittaohjelmien tunnistuksessa on tutkittu viime vuosina yhä enemmän. Syväoppimismallit voidaan kouluttaa tunnistamaan haittaohjelmille tyypilliset piirteet ja luokittelemaan tiedosto automaattisesti hyväntahtoiseksi tai haitalliseksi. Näiden mallien suurimpana etuna pidetään niiden kykyä oppia automaattisesti olennaiset piirteet laajoista tietoaineistoista. Syväoppimisen avulla voidaan tunnistaa monimutkaiset ja muuttuvat hyökkäyskuviot tarkemmin ja tehokkaammin perinteisiin sääntöpohjaisiin menetelmiin tai koneoppimismenetelmiin verrattuna.[26]

Syväoppimispohjainen haittaohjelmantunnistusjärjestelmä (Kuva 2) toimii seuraavasti. Haitallisia ja hyväntahtoisia näytteitä, kuten tiedostoja, ohjelmia, muistivedoksia tai käyttäytymislokeja, kerätään koulutus ja testausaineistoksi. Näytteet esikäsitellään ja syötetään syväoppimismalliin raakadatana. Malli oppii automaattisesti useiden syväoppimisverkkojen avulla tunnistamaan haittaohjelmien ja hyväntahtoisten ohjelmien olennaisia piirteitä, kuten haitallisia API-kutsuja. Lopulta malli luokittelee opittujen piirteiden perusteella ohjelman automaattisesti joko haitalliseksi tai hyväntahtoiseksi.[2], [27]



Kuva 2. Syväoppimispohjainen haittaohjelmien tunnistusjärjestelmä. Mukaillen lähteestä [2].

Tutkielman yhdistelmämalleissa konvoluutioneuroverkot ovat erikoistuneet toistuvien tai rakenteellisten paikallisten piirteiden havaitsemiseen. Eri malleissa konvoluutioneuroverkkoja on hyödynnetty eri tavoin. Konvoluutioneuroverkot voivat poimia API-kutsusekvensseistä haitallisia piirteitä tai toimintaketjuja. Ne voivat tunnistaa kuvamuotoon muunnetuista binäärirakenteista toistuvia malleja tai tunnistaa PE-tiedostojen otsikoista haittaohjelmien säännönmukaisuuksia.[4], [5], [20], [24], [25], [28]

LSTM ja BiGRU-rakenteita hyödynnetään malleissa sekventiaalisen datan analysointiin. Toisin sanoen algoritmit ovat suunniteltu käsittelemään tietoa, jossa järjestyksellä on merkitystä. Niiden tehtävänä on säilyttää pitkän aikavälin riippuvuuksia, eli havaita miten tietyt kutsut tai toiminnot liittyvät toisiinsa. Algoritmit oppivat muistamaan aiemmat tapahtumat ja järjestykset tulevien tapahtumien tulkinnassa. LSTM- ja BiGRU-tyyppiset verkot ovat hyödyllisiä tilanteissa, joissa haitallinen toiminta ilmenee toimintojen järjestyksen tai ajoituksen kautta.[4], [5], [20], [24]

Transformer-pohjaiset mallit keskittyvät kontekstin ja semanttisten suhteiden ymmärtämiseen. Ne eivät tarkastele pelkästään yhtä API-kutsua, vaan muodostavat päätöksen haitallisuudesta tai hyväntahtoisuudesta sen perusteella mitä muita asioita tapahtuu ennen tai jälkeen API-kutsun. Mallit tarkastelevat koko syötteen samanaikaisesti ja arvioivat, mitkä kutsut liittyvät toisiinsa, säilyttäen samalla pitkän aikavälin yhteydet. BERT-mallit huomioivat yksittäisten API-kutsujen lisäksi myös niiden kontekstin, jotta haitallinen toiminta voidaan tunnistaa. Toisin sanoen BERT-mallit muodostavat käsityksen siitä, milloin tietyt kutsut esiintyvät ja mihin muihin kutsuihin ne liittyvät. Käsitellään seuraavaksi tarkemmin tutkielmaan valikoitujen yhdistelmäsyväoppimismallien toimintaperiaatteita.[20], [25]

Maniriho ym. [4] sekä Karat ym. [24] kehittämät syväoppimiskehykset hyödyntävät API-kutsuja ja dynaamista analyysiä haittaohjelmien tunnistuksessa. Molemmissa lähestymistavoissa API-kutsut kerätään ja muunnetaan numeeriseen muotoon syväoppimismallia varten. Maniriho ym. [4] soveltavat CNN-BiGRU-arkkitehtuuria, kun taas Karat ym. [24] hyödyntävät CNN-LSTM- yhdistelmämallia. Molemmissa malleissa konvoluutioneuroverkot tunnistavat haittaohjelmien tyypilliset piirteet, kun taas LSTM- ja BiGRU-komponentit tunnistavat tiettyjä toistuvia toimintoja ja järjestyksiä. CNN-LSTM-yhdistelmämalli on suunniteltu erityisesti polymorfisten ja metamorfisten haittaohjelmien sekä nollapäivähyökkäysten tunnistamiseen. CNN-BiGRU-yhdistelmämalli puolestaan koulutettiin tunnistamaan useita eri haittaohjelmakategorioita, kuten kiristyshaittaohjelmia, viruksia, matoja, troijalaisia, mainos- ja vakoiluohjelmia sekä näppäinpainallusten tallentajia. [4], [24]

Monet API-kutsuihin perustuvat menetelmät keskittyvät yksittäisten kutsujen esiintymistiheyksiin tai erillisiin tunnisteisiin, eivätkä ne huomioi API-kutsujen välisiä semanttisia suhteita tai samankaltaisuuksia. Haitallinen toiminta voi muodostua useiden API-funktioiden yhteisvaikutuksesta, eikä yksittäinen kutsu välttämättä vielä viittaa haitallisuuteen.[20], [25]. Ji Liu ym. [25] sekä Ju Liu ym. [20] syväoppimiskehykset korostavat semanttista ymmärrystä API-funktioiden välisistä suhteista paremman tunnistustarkkuuden saavuttamiseksi.

Ju Liu ym. [20] kehittämä syväoppimismalli ei ainoastaan tunnista haittaohjelmia, vaan myös luokittelee ne eri haittaohjelmaperheisiin. Menetelmä perustuu staattiseen analyysiin ja API-kutsusekvensseihin, jotka muunnetaan BERT-Transformerin avulla semanttisiksi vektoreiksi. Näitä analysoidaan CNN-LSTM-yhdistelmämallilla, jossa konvoluutioneuroverkot

tunnistavat API-kutsusekvenssien paikalliset piirteet, kuten usein esiintyvät toimintaketjut, ja LSTM säilyttää pitkän aikavälin riippuvuudet, jotta haitalliset toistuvat käyttäytymismallit voidaan havaita. Lopuksi softmax-luokitin määrittää, onko kyseessä haittaohjelma ja mihin haittaohjelmakategoriaan se kuuluu. Tämä yhdistelmämalli on suunniteltu tunnistamaan erityisesti hämärrettyjä haittaohjelmia sekä nollapäivähyökkäyksiä.

Ji Liu ym. [25] kehittämässä mallissa erilaisia syväoppimismenetelmiä hyödynnetään muistissa toimivien haittaohjelmien tunnistukseen yhdessä muistianalyysin kanssa. Malli sisältää kaksi haaraa, joista ensimmäinen seuraa ja tallentaa ajonaikaisia API-kutsuja muistivedoksista. Hienosäädetty BERT-Transformeri muuntaa nämä kutsujonot vektoreiksi ja tunnistaa viittaavatko ne haitalliseen toimintaan. Mallin toinen osa muuntaa suoritettavat koodisivut pieniksi RGB-kuviksi, joista ResNet-34 poimii haittaohjelmille tyypilliset binäärirakenteet. Lopulta molempien haarojen piirteet yhdistetään ja syötetään DNN-pohjaiseen luokittelijaan, joka ilmoittaa onko ohjelma haitallinen vai hyväntahtoinen.

Bao ym. [28] yhdistelmämalli parantaa hämärrettyjen Windows-haittaohjelmien tunnistusta yhdistämällä staattisen ja dynaamisen analyysin syväoppimismenetelmiin. Malli sisältää kaksi erillistä haaraa, joista toisen avulla PE-tiedostojen otsikoista ja tilastollisista arvoista poimitaan haittaohjelmille tyypillisiä staattisia piirteitä. Nämä piirteet syötetään CNN-syväoppimismallille analysoitavaksi. Toinen haara hyödyntää dynaamista analyysiä ohjausvirtauskaavion (*eng.* Control Flow Graph, CFG) tuottamiseen. CFG- on ohjelmakoodin suoritusrakennetta kuvaava kaavio, jossa solmut kuvaavat koodilohkoja ja kaaret niiden välisiä mahdollisia suoritussiirtymiä. Ohjausvirtauskaaviota analysoidaan GIN (*eng*. Graph Isomorphism Network) -mallin avulla. GIN on syväoppimiseen perustuva graafineuroverkko, joka kykenee erottamaan hienovaraisia rakenteellisia eroja graafeissa ja tunnistamaan haitallisia ohjauspolkuja. Lopulta syväoppimismallien tulokset yhdistetään täysin kytkettyyn kerrokseen, joka luokittelee tiedoston haitalliseksi tai hyväntahtoiseksi.

Alrehaili ym. [5] syväoppimismalli keskittyy erityisesti APT-hyökkäysten reaaliaikaiseen tunnistamiseen yhdistämällä konvoluutioneuroverkot ja LSTM-arkkitehtuurin. Mallissa konvoluutioneuroverkot poimivat ensin datasta paikallisia rakenteellisia piirteitä, joita LSTM-kerrokset analysoivat huomioiden niiden ajalliset muutokset. Kehitetty yhdistelmämalli oppii itsenäisesti datan sisältämät piilorakenteet ja hyödyntää tehokkaasti ajallista dynamiikkaa, mikä parantaa mallin suorituskykyä monimutkaisten hyökkäystilanteiden tunnistamisessa.

Hien ym. [29] kehittämä yhdistelmämalli hyödyntää staattista- ja dynaamista analyysiä Windows-käyttöjärjestelmään kohdistuvien haittaohjelmien tunnistamiseen. Staattinen analyysi käyttää PE-tiedostoista poimittuja merkkijonoja ja dynaaminen analyysi tarkastelee hiekkalaatikkoympäristöstä kerättyjä API-kutsusekvenssejä. Molemmat piirrejoukot esikäsitellään NLP-menetelmillä ja muunnetaan numeerisiksi sekvensseiksi. Syväoppimismallina käytetään yhdistelmää yksisuuntaisesta konvoluutioneuroverkosta ja Bi-LSTM- arkkitehtuurista, joka kykenee oppimaan sekä paikallisia että ajallisia riippuvuuksia.

Edellä esiteltyjen tutkimusten tavoitteena oli kehittää syväoppimismalli, jonka avulla tehostetaan haittaohjelmien tunnistamista perinteisiin menetelmiin tai muihin syväoppimismalleihin verrattuna. Taulukossa 2 esitetään tarkasteltujen tutkimusten hyödyntämät yhdistelmäsyväoppimismallit, analyysimenetelmät, tunnistuksessa käytetyt piirrejoukot sekä haittaohjelmatyypit, joiden tunnistamiseen mallit ovat koulutettu. Seuraavaksi tarkastellaan näiden syväoppimismallien tehokkuutta haittaohjelmien tunnistuksessa erilaisten tunnuslukujen avulla.

Taulukko 2. Yhteenveto haittaohjelmatunnistukseen kehitetyistä syväoppimismalleista.

Artikkeli	**Syväoppimismalli**	**Analyysimenetelmä**	**Piirteet**	**Haittaohjelmatyypit**
[4]	CNN-BiGRU	Dynaaminen analyysi	API-kutsujonot	Virukset, troijalaiset, madot vakoilu- ja mainosohjelmat, kiristysohjelmat,
[24]	CNN-LSTM	Dynaaminen analyysi	API-kutsut, käyttäytymislokit	Nollapäivähyökkäykset, polymorfiset- ja metamorfiset haittaohjelmat
[20]	BERT, CNN-LSTM	Staattinen analyysi	API-kutsujonot	Polymorfiset ja metamorfiset haittaohjelmat, nollapäivähyökkäykset
[25]	BERT, ResNet, DNN	Muistianalyysi	API-kutsut, muistidata, suoritettavat tiedostot	muistissa toimivat haittaohjelmat
[28]	GIN, CNN	Staattinen- ja dynaaminen analyysi	PE-otsikot, ohjausvirrat	Hämärretyt haittaohjelmat
[29]	CNN, Bi-LSTM	Staattinen- ja dynaaminen analyysi	Merkkijonot, API-kutsut	Haitalliset PE-tiedostot

[5]	CNN-LSTM	-	verkkoliikenne	APT-hyökkäykset

4 Syväoppimismenetelmien tehokkuuden arviointi

Haittaohjelmien tunnistus perustuu haittaohjelmien erottamiseen hyväntahtoisista ohjelmista. Haittaohjelmatutkimuksessa keskeistä on myös haittaohjelmien luokittelu eri haittaohjelmaperheisiin. Tunnistaminen ja luokittelu liittyvät läheisesti toisiinsa ja molemmat ovat olennaisia tehokkaiden tunnistusmenetelmien kehittämiseksi. Tässä tutkielmassa tarkastellaan kuitenkin ainoastaan tuloksia, jotka kertovat syväoppimismallien kyvystä tunnistaa haitalliset ja hyväntahtoiset ohjelmat. Määritellään seuraavaksi arvioinnissa käytetyt suorituskykymittarit, jonka jälkeen esitellään tutkimustulokset ja lopulta tehdään johtopäätökset syväoppimismallien tehokkuudesta.

4.1 Arvioinnissa käytetyt suorituskykymittarit

Syväoppimismenetelmien tehokkuuden arvioinnissa käytetään erilaisia suorituskykymittareita, joita ovat tarkkuus (*eng.* accuracy), täsmällisyys (*eng*. precision), sensitiivisyys (*eng.* recall) ja F1-arvo (*eng.* F1-score). Lisäksi arvioinnissa voidaan käyttää myös muita mittareita, kuten väärien positiivisten osuus (*eng.* False Positive Rate, FPR), oikeiden positiivisten osuus (*eng.* True Positive Rate), tai väärien negatiivisten osuus (*eng.* False negative Rate, FNR) sekä AUROC- arvoa (*eng.* Area Under the receiver operating characteristic).[6], [26]

Suorituskykymittarien tulokset perustuvat seuraaviin tekijöihin: oikeat positiiviset, oikeat negatiiviset, väärät positiiviset sekä väärät negatiiviset. Oikeat positiiviset ovat tapauksia, jossa haittaohjelma on tunnistettu oikein haittaohjelmaksi. Väärät positiiviset ovat tapauksia, jossa hyväntahtoiset ohjelmat on tunnistettu väärin haittaohjelmiksi. Oikeat negatiiviset ovat oikein tunnistettuja hyväntahtoisia ohjelmia. väärät negatiiviset ovat tapauksia, jossa haittaohjelma on virheellisesti luokiteltu hyväntahtoiseksi ohjelmaksi.[6], [26]

Tarkkuus mittaa, kuinka monta havaintoa malli luokittelee oikein kaikkien havaintojen joukosta. Se lasketaan kaavalla 1, missä oikeiden positiivisten ja negatiivisten summa jaetaan kaikkien havaintojen summalla.[29]

$$Tarkkuus = \frac{oikeat\ positiiviset + oikeat\ negatiiviset}{oikeat\ positiiviset + väärät\ positiiviset + oikeat\ negatiiviset + väärät\ negatiiviset} \quad (1)$$

Täsmällisyys kuvaa, miten moni haitalliseksi luokiteltu ohjelma on todella haitallinen. Se lasketaan jakamalla oikeiden positiivisten määrä, oikeiden ja väärien positiivisten summalla. (Kaava 2).[29]

$$Täsmällisyys = \frac{oikeat\ positiiviset}{oikeat\ positiiviset + väärät\ positiiviset} \quad (2)$$

Sensitiivisyys mittaa mallin kykyä tunnistaa positiivisia tapauksia eli oikein tunnistettuja haittaohjelmia. Sensitiivisyys lasketaan jakamalla oikeiden positiivisten määrä oikeiden positiivisten ja väärien negatiivisten summalla (Kaava 3).[29]

$$Sensitiivisyys = \frac{oikeat\ positiiviset}{oikeat\ positiiviset + väärät\ negatiiviset} \quad (3)$$

F1-arvo on täsmällisyyden ja sensitiivisyyden tasapainotettu keskiarvo, joka lasketaan kaavalla 4. Korkea F1-arvo tarkoittaa, että järjestelmä tunnistaa uhat tehokkaasti ilman liiallisia virhehälytyksiä.[29]

$$F1\ arvo = 2 \times \frac{täsmällisyys \times sensitiivisyys}{täsmällisyys + sensitiivisyys} \quad (4)$$

Väärien positiivisten osuus kertoo, kuinka suuri osa hyväntahtoisista haittaohjelmista luokitellaan virheellisesti haittaohjelmiksi. FPR-arvon tulisi olla mahdollisimman matala, jotta mallia voidaan pitää luotettavana haittaohjelmien tunnistuksessa. Sen laskeminen on osoitettu kaavassa 5.[4]

$$FPR = \frac{väärät\ positiiviset}{väärät\ positiiviset + oikeat\ negatiiviset} \quad (5)$$

Väärien negatiivisten osuus kertoo, kuinka suuri osa todellisista haittaohjelmista jää tunnistamatta ja luokitellaan väärin hyväntahtoisiksi. FNR-arvon tulisi olla mahdollisimman matala, jolloin suurin osa haittaohjelmista tunnistetaan. Sen laskeminen on osoitettu kaavassa 6.[4]

$$FNR = \frac{väärät\ negatiiviset}{väärät\ negatiiviset + oikeat\ positiiviset} \quad (6)$$

ROC-käyrä (*eng.* Receiver Operating Charasteristic Curve) on yleinen diagnostiikkatyökalu, jota käytetään arvioimaan binääriluokittelijan suorituskykyä. Se mittaa oikein tunnistettujen haittaohjelmien ja väärin tunnistettujen hyväntahtoisten ohjelmien suhdetta eri kynnysarvoilla

eli kuvaa mallin kykyä erottaa kaksi luokkaa toisistaan. ROC-käyrää käytetään yleensä tilanteessa, jossa luokkien näytteet ovat tasapainossa. AUC (*eng*. Area Under the Curve) on yleisesti käytetty mittari ROC-käyrien vertailuun. AUC-arvo mittaa kuinka hyvin luokittelija erottelee positiiviset ja negatiiviset tapaukset eli miten paljon mallin ennusteet eroavat satunnaisesta arvauksesta. Korkea AUC-arvo (> 0.9) osoittaa, että malli ennustaa luokat tarkasti. Se kuvaa ROC-käyrän alle jäävää pinta-alaa ja sen laskeminen on esitetty kaavassa 7.[4]

$$AUC = \int_0^1 \frac{oikeat\ positiiviset}{oikeat\ positiiviset + väärät\ negatiiviset} d\left(\frac{väärät\ positiiviset}{oikeat\ positiiviset + väärät\ positiiviset}\right) \quad (7)$$

4.2 Syväoppimismallien suorituskyky

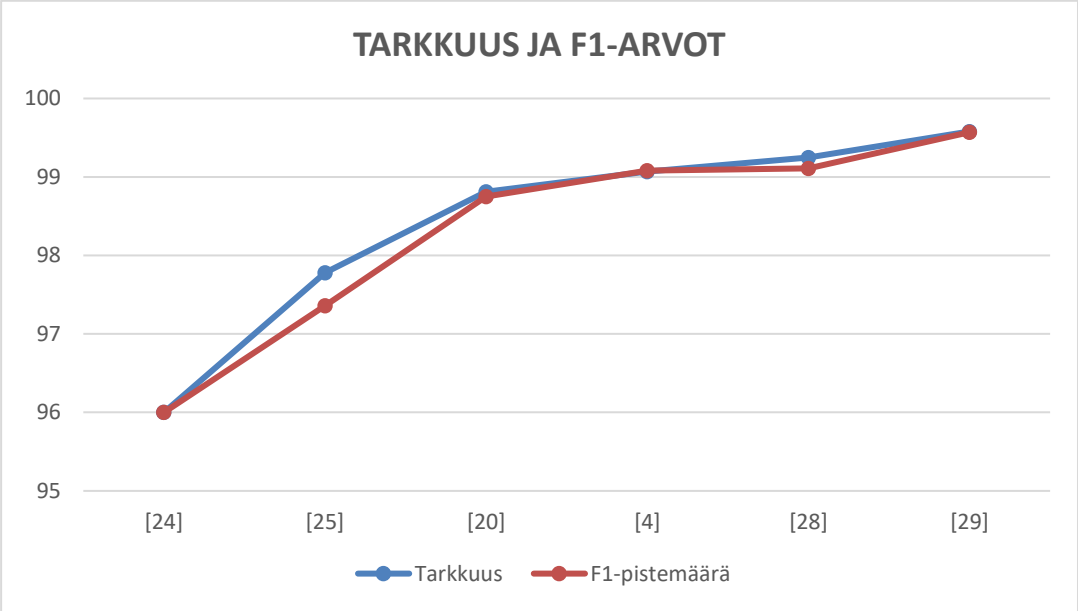
Tarkastellaan syväoppimismallien suorituskykyä edellä esitettyjen mittareiden avulla. Taulukossa 3 esitetään tutkimusten raportoimat tarkkuus-, täsmällisyys-, sensitiivisyys- ja F1-arvot. Näiden lisäksi tarkastellaan TPR-, FPR-, FNR- ja AUC-arvoja, mikäli ne ovat tutkimuksessa ilmoitettu. Tulosten avulla arvioidaan eri syväoppimismallien tunnistustehokkuutta ja hyödyllisyyttä haittaohjelmien havaitsemisessa.

Taulukko 3. Syväoppimismallien tarkkuus, täsmällisyys, sensitiivisyys ja F1-arvo.

Artikkeli	**Käytetty syväoppimismalli**	**Tarkkuus (%)**	**Täsmällisyys**	**Sensitiivisyys**	**F1-arvo**
Maniriho [4]	CNN-BiGRU	99.07	0.9909	0.9907	0.9908
Karat [24]	CNN-LSTM	96	0.96	0.96	0.96
Ji Liu [25]	BERT, Resnet34, DNN	97.78	0.9796	0.9777	0.9736
Bao [28]	CNN, GIN	99.25	1.00	0.9824	0.9911
Alrehaili[5]	CNN-LSTM	99.7	-	-	0.998
Ju Liu [20]	BERT, CNN, LSTM	98.81	-	-	0.9875
Hien [29]	CNN, Bi-LSTM	99.58	0.9958	0.9957	0.9957

Tutkimusten tarkkuus ja F1-arvot osoittavat pääosin kaikkien mallien korkean suorituskyvyn ja tasapainoisuuden. Tuloksissa on kuitenkin havaittavissa pientä vaihtelua. Korkeimman tarkkuuden (99,7 %) ja F1-arvon (0,998) saavutti Alrehaili ym. [5] CNN-LSTM-malli. Myös Bao ym. [28], Maniriho ym. [4] ja Hien ym. [29] mallit suoriutuivat erinomaisesti ylittäen 99 prosentin tarkkuuden ja F1-arvon. Heikoiten suoriutui Karat ym. [24] CNN-LSTM-malli, joka saavutti vain 96 % tarkkuuden ja F1-arvon. Kuviosta 1 nähdään, että tarkkuuden ja F1-arvon välinen ero on useimmissa malleissa pieni, mikä osoittaa mallien tasapainoisen suorituskyvyn. Ne pystyvät säilyttämään hyvän tasapainon täsmällisyyden ja herkkyyden välillä, eli mallit tunnistavat haittaohjelmat hyvin, sekä osaavat välttää väärät hälytykset. Tasapainoinen suorituskyky on tärkeää haittaohjelmien tunnistuksessa, missä tulee minimoida väärät hälytykset ja saavuttaa korkea tarkkuus haitallisten näytteiden tunnistuksessa.

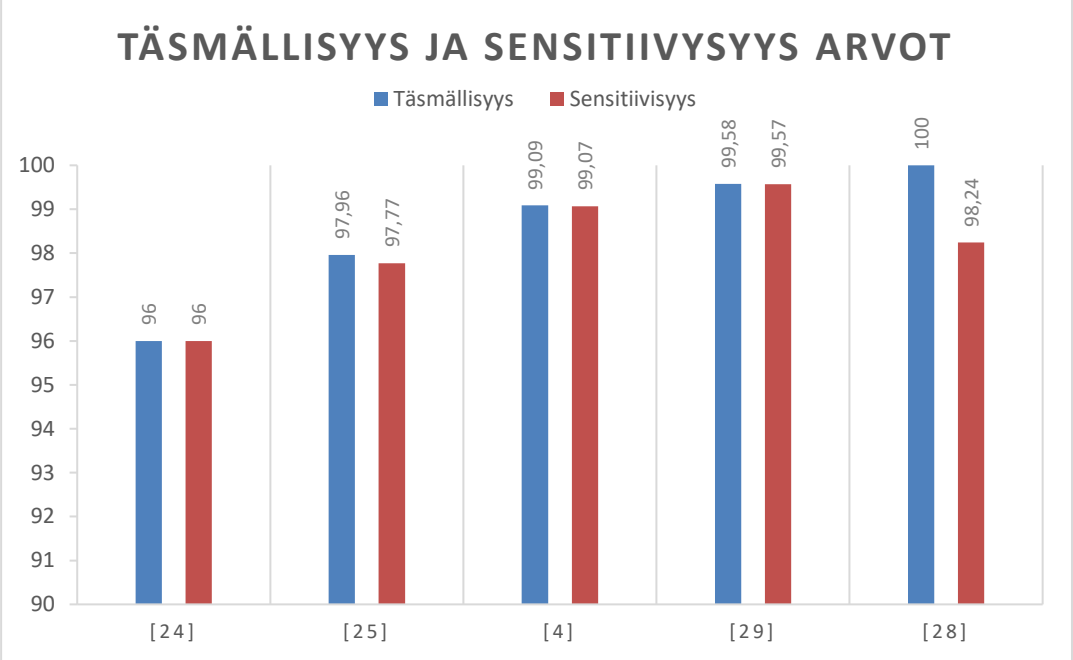
Kuvio 1. Yhdistelmäsyväoppimismallien tarkkuus ja F1-arvojen vertailu.



Mallien korkeiden tarkkuus ja F1-arvojen lisäksi myös täsmällisyydet ja sensitiivisyydet saavuttivat korkeat tulokset. Kuviossa 2 esitetään tutkimusten raportoimat täsmällisyys- ja sensitiivisyysarvot. Tutkimusten [29], [4] ja [25] mallit osoittavat tasapainoista suorituskykyä, missä edellä mainitut arvot ovat lähes yhtä korkeita. Tämä viittaa luotettavaan tunnistukseen ja vähäisiin virhehälytyksiin. Bao ym. [28] mallin täsmällisyys (100 %) korostaa mallin kykyä välttää väärät positiiviset tulokset täydellisesti. Sensitiivisyys (98.24 %) on kuitenkin hieman alempi, jolloin osa haittaohjelmista jää tunnistamatta. Karat ym. [24] mallin saavuttamat arvot (96 %) ovat selvästi muita matalampia, mikä osoittaa rajoitteita tunnistustarkkuudessa muihin

malleihin verrattuna. Tutkimukset [5] ja [20] eivät raportoineet ollenkaan täsmällisyys- ja sensitiivisyysarvoja.

Kuvio 2. Syväoppimismallien täsmällisyys- ja sensitiivisyysarvot.

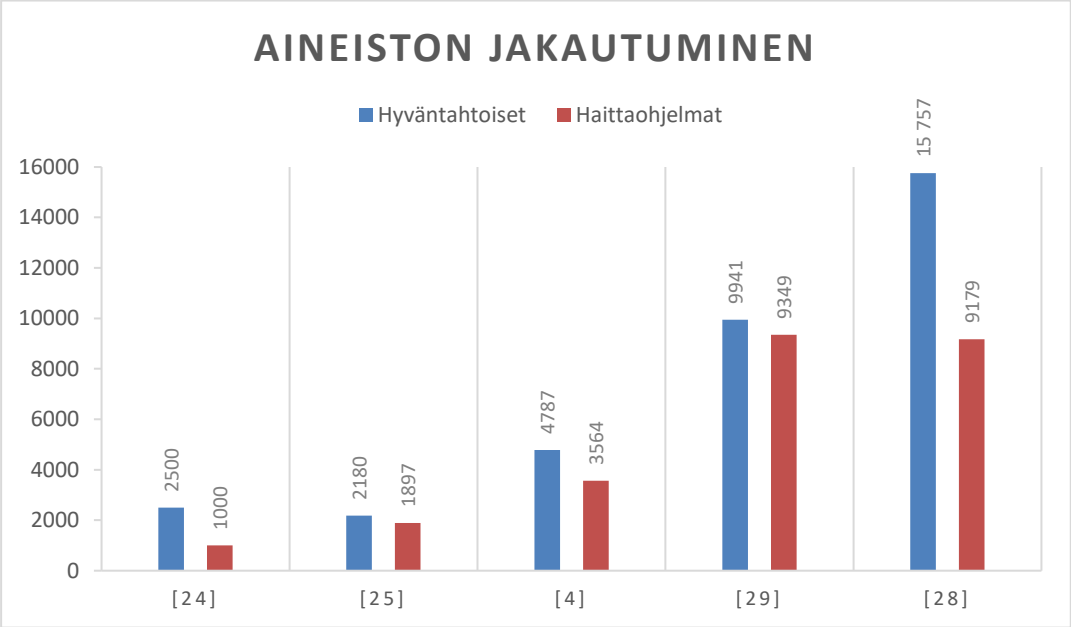


Kuviossa 3 esitetään syväoppimismallien koulutuksessa ja testauksessa käytettyjen aineistojen jakautuminen hyväntahtoisiin ja haitallisiin näytteisiin. Tutkimukset [28], [29], [20] ja [5] hyödynsivät suuria ja monipuolisia aineistoja. Näiden tutkimusten aineistot sisälsivät sekä haittaohjelmanäytteitä että hyväntahtoisia näytteitä useita tuhansia. Ainoastaan Ju Liu ym. [20] ja Alrehaili ym. [5] käyttivät valmiita EMBER- ja DAPT2020-datasettejä, muiden tutkimusten kerätessä näytteet eri tavoin. Edellä mainittujen syväoppimismallien tarkkuus, täsmällisyys, sensitiivisyys ja F1-arvot ylsivät yli 99 prosentin tai hyvin lähelle sitä. Huomattavasti pienempiä aineistoja käyttivät tutkimukset [24], [4] ja [25]. Pienemmästä aineistosta huolimatta Maniriho ym. [4] syväoppimismalli saavutti yli 99 prosentin tarkkuuden, täsmällisyyden, sensitiivisyyden ja F1-arvon. Sen sijaan Karat ym. [24] sekä Ji Liu ym. [25] raportoimat suorituskykyä mittaavat arvot jäivät edellisiä alhaisemmiksi.

Tutkimuksista ainoastaan Ju Liu ym. [20] käyttämä EMBER-datasetti sisälsi yhtä paljon haittaohjelmia sekä hyväntahtoisia näytteitä. Muut tutkimukset hyödynsivät epätasapainoisempaa aineistoa, joissa hyväntahtoisten ohjelmien osuus oli suurempi. Näytemäärän ja tasapainon erot voivat vaikuttaa merkittävästi mallien suorituskykyyn ja siihen, kuinka hyvin ne pystyvät erottamaan haittaohjelmat todellisessa käyttöympäristössä.

Mallien tarkkuutta arvioitaessa epätasapainoisissa aineistoissa otettiin huomioon painotetut keskiarvot, jotta tulokset kuvaisivat paremmin mallien suorituskykyä.

Kuvio 3. Syväoppimismallien koulutuksessa ja testauksessa käytetyn Datan jakautuminen hyväntahtoisiin ja haittaohjelmiin.



Tarkastellaan seuraavaksi syväoppimismallien suoritustehokkuutta niiltä osin kuin tutkimuksissa on raportoitu. Ainoastaan Maniriho ym. [4] ilmoittivat mallin koulutus- ja testausajan. Yhdistelmämalli koulutettiin ja testattiin eri pituisilla API-kutsusekvensseillä, jolloin koulutusaika nousi 36 sekunnista 221 sekuntiin kutsusekvenssien pidentyessä. Lisäksi mallin keskimääräinen havaitsemisaika oli 0.298 sekuntia. Tarkkuus parani pidemmillä kutsusekvensseillä, joten mallin hyödyllisyyttä on kriittistä tarkastella suhteessa suoritusaikaan. Bao ym. [28] toteavat, että GIN-pohjaisen mallin graafirakenteiden luominen on laskennallisesti raskasta, mikä heikentää mallin käytännön soveltuvuutta, vaikka malli saavuttaakin korkean tarkkuuden haittaohjelmien tunnistuksessa. Muut tutkimukset eivät raportoineet koulutukseen ja testaukseen käytettyä aikaa.

Tarkastelluissa tutkimuksissa raportoitiin vaihtelevasti AUC- FPR- ja FNR-arvoja, mikä vaikuttaa mallien vertailtavuuteen ja arviointiin. Alrehaili ym.[5] ilmoittivat AUC-arvoksi 0,982, mutta eivät raportoineet FPR- ja FNR-lukuja. Karat ym.[24] saavuttivat korkean AUC-arvon (0,99), mutta mallille laskettu FNR (2,81 %) ja FPR (7,23 %) osoittavat haasteita varsinkin hyväntahtoisten luokittelussa virheellisesti haitallisiksi. Maniriho ym.[4] yhdistelmämallin suorituskyky oli myös näillä mittareilla vahva. AUC- arvoksi raportoitiin 0,9992, samalla kun FNR- (0,15 %) ja FPR- (1,04 %) arvot pysyivät matalina. Bao ym. [28]

eivät raportoineen AUC-arvoa lainkaan, mutta ilmoittivat TPR:n olevan 98,24 ja FPR:n 0, mikä viittaa hyvään havaitsemiskykyyn ja vähäisiin virhehälytyksiin. Sen sijaan tutkimukset [20], [25], [29] eivät raportoineet lainkaan FPR-, FNR- ja AUC-arvoja, mikä vaikeuttaa mallien todellisen suorituskyvyn arviointia.

Hien ym. [29] arvioivat kehittämänsä yhdistelmämallin perinteisen luokittelutarkkuuden lisäksi sen kykyä tunnistaa uusia, aiemmin näkemättömiä haittaohjelmia. Tätä varten malli koulutettiin aineistolla, joka sisälsi haittaohjelmanäytteitä kolmelta eri aikaleimalta vuosilta 2021–2023. Testiaineistona käytettiin tuoreita, vasta julkaistuja hyväntahtoisia ja haitallisia näytteitä. Tulosten perusteella mallin suorituskyky heikkeni selvästi: tarkkuus laski 96,84 prosenttiin ja sensitiivisyys 95,82 prosenttiin. Myös täsmällisyys ja F1-arvo heikentyivät. Tämä viittaa siihen, että malli ei kykene havaitsemaan uusia ja muuntuvia haittaohjelmia yhtä tehokkaasti kuin sellaisia, jotka muistuttavat koulutusaineiston näytteitä.

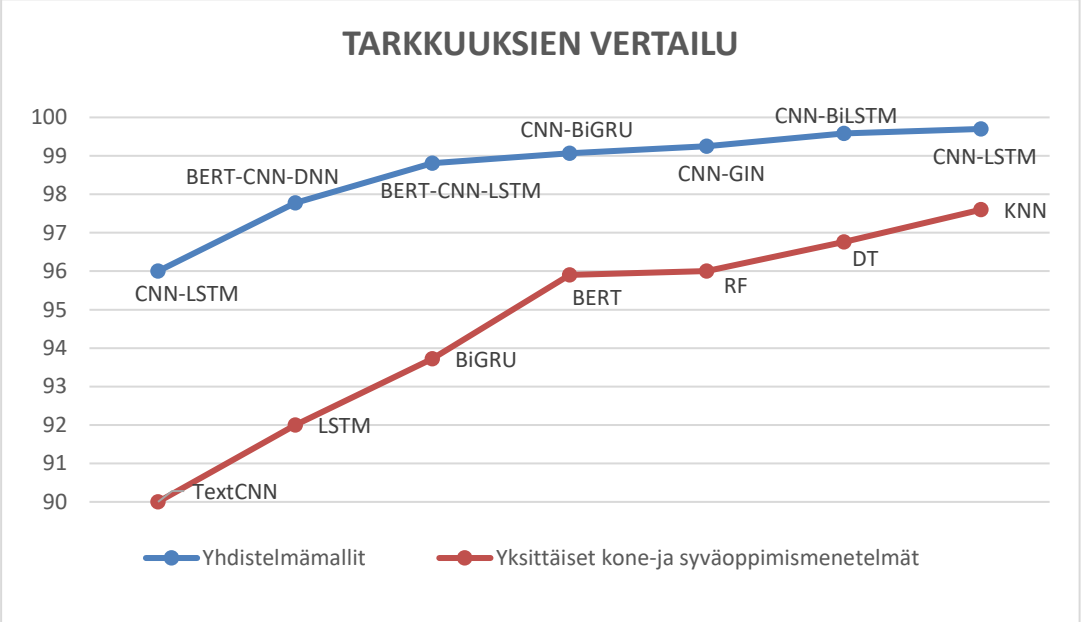
Maniriho ym. [4] CNN-BiGRU-mallin tarkkuus, AUC- ja F1-arvot paranivat pidemmillä API-kutsusekvensseillä. Mallia testattiin kolmen testiaineiston ja eripituisten API-kutsusekvenssien avulla. Vaikka mallin tarkkuus, AUC- ja F1-arvot paranivat, FNR-arvo kasvoi kahdessa ensimmäisessä testissä API-kutsusekvenssien pidentyessä. Kolmannessa testissä FNR-arvo laski 0.87 prosentista 0.59 prosenttiin, mikä viittaa siihen, että sekvenssipituuden vaikutus riippuu käytetystä aineistosta ja haittaohjelmien tyypistä. Tämän vuoksi API-kutsusekvenssien pituutta tulisi optimoida huolellisesti, jotta haittaohjelmien tunnistuksen tehokkuus ei heikkene. Vaikka FNR-arvo kasvaa kahdessa testissä, voidaan sitä silti pitää riittävän alhaisena tuloksena, mikä tarkoittaa, että malli tunnistaa lähes kaikki haittaohjelmat.

4.3 Johtopäätökset

Tulokset osoittavat, että yhdistelmäsyväoppimismallit saavuttavat erittäin korkeat tarkkuudet, täsmällisyydet, sensitiivisyydet ja F1-arvot haittaohjelmien tunnistuksessa. Kuviossa 4 esitetään yhdistelmäsyväoppimismallien sekä yksittäisten kone- ja syväoppimismenetelmien tarkkuuksien vertailu. Kuviossa koneoppimismenetelmiä ovat KNN (*eng.* K-Nearest Neighbor), DT (*eng*. Decision Tree) ja RF (*eng*. Random Forest) [4]. Huomataan, että useimmat yhdistelmämallit ylittävät tarkkuudeltaan yksittäiset syvä- ja koneoppimismallit. Ainoastaan Karat ym. [24] CNN-LSTM-mallin tarkkuus jää vertailussa alhaisemmaksi kuin yksittäisen BERT- tai KNN-mallin. Huomattavaa on myös, että yksittäisten syväoppimismallien tarkkuudet jäävät alhaisemmiksi yhdistelmämalleihin sekä

koneoppimismenetelmiin verrattuna. Näiden tulosten pohjalta voidaan todeta, että yhdistelmäsyväoppimismallit ovat tehokkaampia yksittäisiin malleihin verrattuna. Tulos vahvistaa käsitystä siitä, että syväoppimisen yhdistäminen monipuoliseen ominaisuusanalyysiin on lupaava ratkaisu haittaohjelmien havaitsemisessa.

Kuvio 4. Yhdistelmäsyväoppimismallien vertailu yksittäisten Kone- ja syväoppimismallien tuloksiin.[4]



Tarkkuus on hyvä yleinen arviointimittari, mutta se ei välttämättä kerro todenmukaisesti haittaohjelmien tunnistusmallin suorituskyvystä. Tarkkuus riippuu usein käytetystä koulutus- ja testiaineistosta, kuten aineiston monimutkaisuudesta tai haittaohjelmien varianttien määrästä. Tutkimuksissa on käytetty toisistaan poikkeavia aineistoja, joten parempi suorituskyky saattaa osittain selittyä laadukkaammalla ja suuremmalla aineistolla. Pääsääntöisesti suuremmat koulutus- ja testausaineistot johtivat korkeampiin tuloksiin, mutta Maniriho ym. [4] saavuttivat korkean tunnistustarkkuuden myös pienemmällä aineistolla.

Useimmissa tutkimuksissa käytetty aineisto oli jakautunut epätasapainoisesti, jolloin mallin korkea tarkkuus voitaisiin saavuttaa ennustamalla useimmat havainnot enemmistöluokaksi. Tutkimusten raportoimat F1-arvot ovat kuitenkin korkeat sekä AUC-arvot nousevat lähelle 1.0, minkä perusteella mallit ovat tasapainoisia. F1-arvot kertovat, että yhdistelmämallit tunnistavat tehokkaasti haittaohjelmat sekä välttävät hyvin virheellisiä luokituksia. Korkeat AUC-arvot kuvaavat mallien hyvää kykyä erotella haittaohjelmat ja hyväntahtoiset toisistaan. Tulosten perusteella voidaan todeta, että yhdistelmämallien tekemät luokitukset eivät perustu pelkästään arvaukseen tai enemmistöluokan suosimiseen. Osassa tutkimuksissa ei kuitenkaan

raportoida kaikkia suorituskykymittareita tai AUC-arvoja. Vaikka suurin osa malleista suoriutui tarkastelluilla mittareilla hyvin, ei niiden tuloksia voida pitää täysin luotettavina. Kaikkien keskeisten mittareiden raportointi olisi tärkeää vertailukelpoisuuden ja käytännön luotettavuuden kannalta.

Reaalimaailman haittaohjelmien tunnistuksessa pelkkien suorituskykymittareiden lisäksi on tärkeää kiinnittää huomiota myös keskeisiin haasteisiin, kuten suoritusnopeuteen, resurssirajoituksiin, mallin joustavuuteen sekä kykyyn tunnistaa uusia tai muuntuneita haittaohjelmia. Harva tutkimuksista raportoi mallien laskennallista tehokkuutta tai soveltuvuutta kevytresurssisiin ympäristöihin. Yhdistelmäsyväoppimismallit ovat rakenteeltaan monimutkaisia, sillä ne hyödyntävät useita syväoppimisarkkitehtuureja. Vaikka mallit saavuttivat korkeat tarkkuudet, niiden laskennallinen kuormitus on tyypillisesti suurempi verrattuna perinteisiin koneoppimismenetelmiin. Erityisesti analysoitaessa pitkiä API-kutsujonoja tai monimutkaisia graafirakenteita. Tämä heikentää mallien suoraa sovellettavuutta käytännön sovelluksiin. Tulevissa tutkimuksissa tulisi huomioida suorituskykymittareiden lisäksi myös mallien optimointi reaaliaikaiseen tai resurssirajoitettuun käyttöön.

Haittaohjelmien nopea kehitys ja muuntautuminen edellyttää myös mallien säännöllistä päivittämistä. Aiemmin toimineet piirteet eivät välttämättä ole enää käyttökelpoisia uusien uhkien tunnistamisessa. Varsinkin nollapäiväuhkien havaitseminen on edelleen haastavaa syväoppimismallien avulla [13]. Tutkielmassa huomattiin myös, että mallien tulokset laskivat huomattavasti, kun niitä testattiin pelkästään uusia näytteitä sisältävillä aineistoilla.

Mallin valinta riippuu aina käyttökontekstista ja priorisoitavista tekijöistä, kuten saatavilla olevasta aineistosta, halutusta havaintotarkkuudesta, käytettävissä olevista resursseista sekä hyväksyttävien virheiden tyypeistä. Esimerkiksi kriittisillä toimialoilla tavoitteena voi olla havaita kaikki mahdolliset haittaohjelmat, vaikka se lisäisi väärien positiivisten määrää. Sen sijaan käytettävyyttä korostavissa ympäristöissä on tärkeämpää minimoida väärät hälytykset, jotka voivat estää laillisten sovellusten toimintaa ja heikentää käyttökokemusta.

Useissa artikkeleissa hyödynnettiin valmiiksi merkittyjä haittaohjelmanäytteitä, jotka eivät välttämättä edusta uusimpia tai muuntautuvia haittaohjelmia. Tämä asettaa rajoituksia mallien yleistettävyydelle ja käytettävyydelle todellisissa ympäristöissä. Esimerkiksi Hien ym. [29] yhdistelmämallin suorituskyky heikkeni huomattavasti, kun sitä testattiin ajallisesti uudella aineistolla. Lisäksi aineistojakaumat vaihtelivat suuresti eri tutkimusten välillä. Realistisissa

ympäristöissä haittaohjelmien osuus on tyypillisesti vain noin 10 prosenttia tai vähemmän kaikista näytteistä [4], [29]. Tasapainoisilla tai lähes tasapainoisilla aineistolla koulutettujen mallien suorituskyky saattaa olla koulutusolosuhteissa tehokas, mutta ei vastaa todellisen ympäristön suorituskykyä.

Monissa tutkimuksissa mallien vertailu aiempiin menetelmiin oli rajallista tai puutteellista. Vertailukohteina toimivat mallit eivät välttämättä edustaneet alan edistyneimpiä tutkimuksia. Lisäksi osa tarkastelluista artikkeleista jättivät raportoimatta keskeisiä suorituskykymittareita tai ajallisen suoritustehokkuuden, mikä vaikeuttaa menetelmien systemaattista arviointia ja keskinäistä vertailua.

Tutkielmassa tarkastellaan ainoastaan seitsemää yhdistelmäsyväoppimismallia, joiden avulla ei voida tehdä kattavaa päätelmää syväoppimismallien tehokkuudesta. Vaikka tässä tutkielmassa tarkastellut mallit saavuttivat korkean suorituskyvyn, ei niiden perusteella voida vielä osoittaa, että yhdistelmäpohjaiset mallit olisivat yleisesti paras ratkaisu haittaohjelmien tunnistamiseen. Jatkotutkimuksissa tulisikin kiinnittää enemmän huomiota mallien yleistettävyyteen, realistisiin testausympäristöihin sekä vertailun kattavuuteen ja läpinäkyvyyteen.

Pelkästään näiden suorituskykymittareiden perusteella ei voida yksiselitteisesti osoittaa, mikä malleista toimisi parhaiten haittaohjelmien tunnistuksessa. Korkeat tulokset kuitenkin osoittavat, että yhdistelmäsyväoppimismenetelmät ovat mahdollisesti lupaava menetelmä haittaohjelmien tunnistuksessa tulevaisuudessa. Ne suoriutuvat tarkkuuden perusteella paremmin kuin yksittäiset syvä- tai koneoppimismallit, mikä korostaa yhdistelmämallien potentiaalia erityisesti tilanteissa, joissa laskentatehoa on riittävästi tarjolla.

Klassinen koneoppiminen on kuitenkin edelleen varteenotettava vaihtoehto erityisesti pienemmän skaalan, rajoitetun laskentatehon ympäristöissä sekä tilanteissa, joissa tarvitaan nopeaa ja selkeää mallintamistapaa. Syväoppimismenetelmien yhdistelmiä tulisi kehittää entisestään, jotta tulevaisuudessa voidaan tarjota tarkempia ja tehokkaampia automatisoituja ratkaisuja haittaohjelmien tunnistukseen. Jatkossa olisi tärkeää selvittää näiden mallien soveltuvuutta myös muihin käyttöympäristöihin, kuten Android-pohjaisiin sovelluksiin. Lisäksi mallien käytettävyyttä ja luotettavuutta todellisissa tuotantoympäristöissä tulisi tutkia syvällisemmin. Klassisten koneoppimismenetelmien ja syväoppimisen yhdistäminen voisi osoittautua hyödylliseksi erityisesti resurssirajoitteisissa ympäristöissä, joissa vaaditaan sekä tehokkuutta että laskennallista keveyttä.

5 Yhteenveto

Windows-käyttöjärjestelmä suosio tekee siitä erityisen houkuttelevan kohteen haittaohjelmahyökkäyksille. Kehittyneet ja muuntautuvat haittaohjelmat haastavat perinteisiä tunnistusmenetelmiä, mikä korostaa tarvetta tehokkaille automaattisille ratkaisuille. Tämän tutkielman tavoitteena oli selvittää yhdistelmäsyväoppimismenetelmien tehokkuutta haittaohjelmien tunnistuksessa erilaisten suorituskykymittareiden avulla sekä tarkastella, millaisia syväoppimisarkkitehtuureja ja piirteitä näissä malleissa hyödynnetään.

Haittaohjelmien tunnistuksessa hyödynnetään usein staattista-, dynaamista- tai muistianalyysiä sekä näiden yhdistelmiä. Analyysimenetelmien avulla voidaan tuottaa joko rakenteellisia piirteitä tai toiminnallisia tunnusmerkkejä, kuten PE-tiedostojen rakenne-elementtejä, API-kutsuja tai merkkijonoja. Näiden piirteiden avulla syväoppimismallit voidaan kouluttaa erottamaan haitalliset ja hyväntahtoiset ohjelmat toisistaan.

Tutkielmassa tarkastellut yhdistelmämallit sisälsivät konvoluutioneuroverkkojen lisäksi yhden tai useamman syväoppimisalgoritmin, kuten LSTM-, BERT- tai GRU-arkkitehtuurin. Konvoluutioneuroverkkoja hyödynnettiin erityisesti toistuvien tai paikallisten rakenteellisten piirteiden havaitsemiseen, kun taas LSTM- ja BiGRU-malleja sovellettiin järjestyksessä etenevän aineiston ajalliseen analyysiin. BERT-pohjaiset ratkaisut puolestaan mahdollistivat syvällisemmän semanttisen ymmärryksen.

Kaikki seitsemän yhdistelmämallia saavuttivat vähintään 96 prosentin tarkkuuden, ja useat mallit ylsivät jopa yli 99 prosentin tarkkuuteen. Myös täsmällisyys, sensitiivisyys ja F1-arvot ylsivät korkeisiin tuloksiin, mikä viittaa mallien tasapainoiseen suorituskykyyn sekä haittaohjelmien havaitsemisessa että väärien hälytysten välttämisessä. Pelkästään tarkkuuden perusteella suurinta osaa yhdistelmämalleista voidaan pitää tehokkaampina yksittäisiin syvä- ja koneoppimismenetelmiin verrattuna.

Laajoja ja monipuolisia aineistoja hyödyntävissä tutkimuksissa raportoitiin pääsääntöisesti korkeampi tarkkuus, täsmällisyys, sensitiivisyys sekä F1-arvo. Aineiston laatu, määrä ja tasapaino ovat keskeisiä tekijöitä, jotka vaikuttavat syväoppimismallien tehokkuuteen. Vaikka yhdistelmäsyväoppimismallit saavuttivat lupaavia tuloksia haittaohjelmien tunnistuksessa, niihin liittyy edelleen rajoitteita. Useissa tarkastelluissa tutkimuksissa hyödynnetyt aineistot eivät välttämättä vastaa realistista, epätasapainoista uhkaympäristöä, jossa haittaohjelmien osuus on tyypillisesti pieni. Tämä heikentää mallien tulosten yleistettävyyttä todellisiin

käyttötilanteisiin. Lisäksi osa tutkimuksista ei raportoinut keskeisiä suorituskykymittareita, mikä vaikeuttaa mallien vertailua ja arviointia.

Mallit ylsivät pääsääntöisesti korkeisiin tarkkuuksiin kontrolloiduissa olosuhteissa, mutta ne voivat osoittautua laskennallisesti raskaiksi ja vaatia suuria määriä laadukasta koulutusdataa. Erityisiä haasteita havaittiin esimerkiksi pitkien API-kutsujonojen käsittelyssä, koulutusaineistojen rajallisuudessa sekä mallien heikossa kyvyssä yleistää uusiin ja muuntuviin haittaohjelmaversioihin.

Yhdistelmäsyväoppimismallit tarjoavat potentiaalisen ratkaisun haittaohjelmien tunnistukseen tulevaisuudessa. Jotta niiden sovellettavuus todelliseen ympäristöön voisi toteutua, jatkotutkimuksessa tulisi keskittyä erityisesti mallien testaamiseen realistisissa ja epätasapainoisissa ympäristöissä, yleistettävyyden arviointiin sekä läpinäkyvään suorituskyvyn vertailuun. Lisäksi yhdistelmämallien ja perinteisten koneoppimismenetelmien yhdistäminen saattaa mahdollistaa kevyemmät ja tehokkaammat ratkaisut erityisesti resurssirajoitteisiin sovelluskohteisiin.

Lähteet

[1] European Union Agency for Cybersecurity., *ENISA threat landscape 2024: July 2023 to June 2024.* LU: Publications Office, 2024. Viitattu: 16. tammikuuta 2025. [Verkossa]. Saatavissa: https://data.europa.eu/doi/10.2824/0710888

[2] G. M. ja S. C. Sethuraman, ”A comprehensive survey on deep learning based malware detection techniques”, *Computer Science Review*, vsk. 47, s. 100529, helmi 2023, doi: 10.1016/j.cosrev.2022.100529.

[3] AV-TEST GmbH, *AV-Test Antivirus Comparisons*, Antivirus Test Results. Viitattu: 20. maaliskuuta 2025. [Verkossa]. Saatavissa: https://portal.av-atlas.org/

[4] P. Maniriho, A. N. Mahmood, ja M. J. M. Chowdhury, ”API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques”, *Journal of Network and Computer Applications*, vsk. 218, s. 103704, syys 2023, doi: 10.1016/j.jnca.2023.103704.

[5] M. Alrehaili, A. Alshamrani, ja A. Eshmawi, ”A Hybrid Deep Learning Approach for Advanced Persistent Threat Attack Detection”, teoksessa *The 5th International Conference on Future Networks & Distributed Systems*, Dubai United Arab Emirates: ACM, joulu 2021, ss. 78–86. doi: 10.1145/3508072.3508085.

[6] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, ja S. Venkatraman, ”Robust Intelligent Malware Detection Using Deep Learning”, *IEEE Access*, vsk. 7, ss. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[7] O. Or-Meir, N. Nissim, Y. Elovici, ja L. Rokach, ”Dynamic Malware Analysis in the Modern Era—A State of the Art Survey”, *ACM Comput. Surv.*, vsk. 52, nro 5, ss. 1–48, syys 2020, doi: 10.1145/3329786.

[8] J. Chandy, ”Review on Malware, Types, and its Analysis”, *IJRASET*, vsk. 10, nro 12, ss. 386–390, joulu 2022, doi: 10.22214/ijraset.2022.47887.

[9] H. Oz, A. Aris, A. Levi, ja A. S. Uluagac, ”A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions”, *ACM Comput. Surv.*, vsk. 54, nro 11s, ss. 1–37, tammi 2022, doi: 10.1145/3514229.

[10] M. Jemal ja D. C.-T. Lo, ”Detection of Ransomware Attack Using Deep Learning”, teoksessa *2023 IEEE Conference on Dependable and Secure Computing (DSC)*, Tampa, FL, USA: IEEE, marras 2023, ss. 1–9. doi: 10.1109/DSC61021.2023.10354186.

[11] A. Bensaoud, J. Kalita, ja M. Bensaoud, ”A survey of malware detection using deep learning”, *Machine Learning with Applications*, vsk. 16, s. 100546, kesä 2024, doi: 10.1016/j.mlwa.2024.100546.

[12] O. Aslan ja R. Samet, ”A Comprehensive Review on Malware Detection Approaches”, *IEEE Access*, vsk. 8, ss. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[13] F. Deldar ja M. Abadi, ”Deep Learning for Zero-day Malware Detection and Classification: A Survey”, *ACM Comput. Surv.*, vsk. 56, nro 2, ss. 1–37, helmi 2024, doi: 10.1145/3605775.

[14] P. Maniriho, A. N. Mahmood, ja M. J. M. Chowdhury, ”A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms”, *ACM Comput. Surv.*, vsk. 56, nro 6, ss. 1–41, kesä 2024, doi: 10.1145/3638240.

[15] N. Kriegeskorte ja T. Golan, ”Neural network models and deep learning”, *Current Biology*, vsk. 29, nro 7, ss. R231–R236, huhti 2019, doi: 10.1016/j.cub.2019.02.034.

[16] O. A. Montesinos López, A. Montesinos López, ja J. Crossa, *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Cham: Springer Nature, 2022.

[17] Y. LeCun, Y. Bengio, ja G. Hinton, ”Deep learning”, *Nature*, vsk. 521, nro 7553, ss. 436–444, touko 2015, doi: 10.1038/nature14539.

[18] A. Vaswani *ym.*, ”Attention Is All You Need”, 2. elokuuta 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[19] A. Gillioz, J. Casas, E. Mugellini, ja O. A. Khaled, ”Overview of the Transformer-based Models for NLP Tasks”, esitetty tilaisuudessa 2020 Federated Conference on Computer Science and Information Systems, syys 2020, ss. 179–183. doi: 10.15439/2020F20.

[20] J. Liu, Y. Zhao, Y. Feng, Y. Hu, ja X. Ma, ”SeMalBERT: Semantic-based malware detection with bidirectional encoder representations from transformers”, *Journal of Information Security and Applications*, vsk. 80, s. 103690, helmi 2024, doi: 10.1016/j.jisa.2023.103690.

[21] I. Kara, ”Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges”, *Expert Systems with Applications*, vsk. 214, s. 119133, maalis 2023, doi: 10.1016/j.eswa.2022.119133.

[22] D. Gibert, C. Mateu, ja J. Planes, ”The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”, *Journal of Network and Computer Applications*, vsk. 153, s. 102526, maalis 2020, doi: 10.1016/j.jnca.2019.102526.

[23] P. Maniriho, A. N. Mahmood, ja M. J. M. Chowdhury, ”A systematic literature review on Windows malware detection: Techniques, research issues, and future directions”, *Journal of Systems and Software*, vsk. 209, s. 111921, maalis 2024, doi: 10.1016/j.jss.2023.111921.

[24] G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. V. G, ja P. Poornachandran, ”CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection”, *Procedia Computer Science*, vsk. 233, ss. 492–503, 2024, doi: 10.1016/j.procs.2024.03.239.

[25] J. Liu *ym.*, ”MemAPIDet: A Novel Memory-resident Malware Detection Framework Combining API Sequence and Memory Features”, teoksessa *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Tianjin, China: IEEE, touko 2024, ss. 2918–2924. doi: 10.1109/CSCWD61410.2024.10580589.

[26] Y. Liu, H. Fan, J. Zhao, J. Zhang, ja X. Yin, ”Efficient and Generalized Image-Based CNN Algorithm for Multi-Class Malware Detection”, *IEEE Access*, vsk. 12, ss. 104317–104332, 2024, doi: 10.1109/ACCESS.2024.3435362.

[27] N. Sharma, I. Batra, ja A. Malik, ”A Comparative Study of Malware Detection Using Hybrid Deep Learning Techniques”, teoksessa *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, Manama, Bahrain: IEEE, tammi 2024, ss. 1260–1266. doi: 10.1109/ICETSIS61505.2024.10459638.

[28] P. T. Bao, D. T. Thu Hien, N. T. Cam, ja V.-H. Pham, ”A multimodal Windows malware detection method based on hybrid analysis and graph representations”, teoksessa *2024 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, Da Nang, Vietnam: IEEE, elo 2024, ss. 1–6. doi: 10.1109/MAPR63514.2024.10660853.

[29] D. T. T. Hien, N. Q. Huy, B. D. Hoang, N. T. Cam, ja V.-H. Pham, ”A study on natural language processing-based method for Windows malware detection”, teoksessa *2024 Tenth International Conference on Communications and Electronics (ICCE)*, Danang, Vietnam: IEEE, heinä 2024, ss. 403–408. doi: 10.1109/ICCE62051.2024.10634666.