
Henrik Aho

# Tekoälymallinnuksen kehittäminen väliintulohyökkäyksiä vastaan

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

23.11.2023

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Henrik Aho |
| Otsikko: | Tekoälymallinnuksen kehittäminen väliintulohyökkäyksiä vastaan |
| Sivumäärä: | 33 sivua |
| Aika: | 23.11.2023 |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja viestintätekniikan tutkinto-ohjelma |
| Ammatillinen pääaine: | Ohjelmistotuotanto |
| Ohjaajat: | Vastuuarvioija Vesa Ollikainen |


---


Tämän insinöörityön tarkoituksena on tutkia väliintulohyökkäysten toimintaa, kehitystä sekä torjuntaa tekoälyä hyväksikäyttäen. Tämän lisäksi käydään yleisesti läpi tietoa kyberhyökkäyksistä, pakettien haistajista, koneoppimisalgoritmeista ja tekoälyn kehittymisestä.

Työssä kehitetään tekoälypohjainen mallinnus, jonka avulla olisi mahdollista kehittää tietoturvaa väliintulohyökkäysten havaitsemiseksi ja torjumiseksi sekä hyökkäyksistä hälyttämiseksi. Työssä käydään läpi eri suunnittelun, toteutuksen sekä jatkokehityksen vaiheita. Tekstissä käsitellään ohjelmointirakennetta Python-kielellä.

Mallinnuksen ideana on antaa lukijalle ymmärrys sekä ideoita tekoälypohjaisen tietoturvan kehityksestä, toiminnasta ja haasteista. Mallinnusta on mahdollista käyttää apuna muidenkin kyberhyökkäystyyppien torjumiseen.



Avainsanat: kyberhyökkäys, tekoäly, tietoturva, väliintulohyökkäys


---

# Abstract


| | |
|---|---|
| Author: | Henrik Aho |
| Title: | Developing AI-Model against Man-In-The-Middle Attacks |
| Number of Pages: | 33 pages |
| Date: | 23 November 2023 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information and Communication Technology |
| Professional Major: | Software Engineering |
| Supervisor: | Vesa Ollikainen, Senior Lecturer |


---


The purpose of this thesis is to study the operation, development and prevention of man-in-the-middle attacks using the help of artificial intelligence. In addition, general information on cyber attacks, packet sniffers, machine learning algorithms and the development of artificial intelligence is provided.

The study consisted of developing an AI-model, which would make it possible to develop information security that detects, repels and warns of possible man-in-the-middle attacks. This paper goes through the different stages of planning, implementation and further development. It also discusses the programming structure in the Python language.

The idea of the model is to give the reader an understanding and ideas about the development, operation and challenges of artificial intelligence-based information security. It is possible to use this model against other types of cyber attacks as well.



Keywords: cyber attack, artificial intelligence, information security, man-in-the-middle attack


---

# Sisällys

## Lyhenteet

ARP: *Address Resolution Protocol*. Protokolla, jolla hankitaan fyysinen osoite, joka vastaa loogista osoitetta Ethernet-verkoissa.

CSR: *Certificate Signing Request.* Hakijan lähettämä viesti, jolla pyydetään digitaalisen identiteetin varmennusta varmenneviranomaisilta.

DNS: *Domain Name System.* Verkkotunnuksia IP-osoitteiksi muuttava nimipalvelujärjestelmä.

LDAP: *Lightweight Directory Access Protocol.* Verkkoprotokolla, joka on tarkoitettu hakemistopalvelujen käyttöä varten. Protokollaa käytetään tietojen hakemiseen verkon yli keskitetystä palvelusta.

MITM: *Man-in-the-middle*. Väliintulohyökkäys on kyberhyökkäys, jossa hyökkääjä tunkeutuu kahden tai useamman osapuolen välisen viestinnän väliin ja esiintyy usein yhtenä osapuolena keskustelussa.

OCSP: *Online Certificate Status Protocol.* Yhteyskäytäntö varmenteen tilan kyselyä varten. OCSP:n avulla voidaan tarkistaa varmenteen oikeellisuus.

POODLE: *Padding Oracle On Downgraded Legacy Encryption.* Tietoturvahaavoittuvuus, jossa käytetään hyväkseen vanhentunutta salausmuotoa.

SSL: *Secure Sockets Layer.* Suojausprotokolla, jonka avulla voidaan luoda salattu yhteys selaimen ja verkkopalvelimen välille.

# 1 Johdanto

Teknologian kehittyessä yhä useammat palvelut ja toiminnot ovat siirtyneet verkkoon. Tämä on mahdollistanut sen, että monet asioinnit pystytään suorittamaan entistä nopeammin, mutta se on tuonut mukanaan paljon riskejä. Verkkorikollisuus on digitalisoitumisen ansiosta laajentunut perinteisistä sähköpostihuijauksista vaikeasti tunnistettaviin hyökkäyksiin.

Termi ”tekoäly” on noussut yhä suuremmaksi puheenaiheeksi viimeisen vuosikymmenen aikana. Tekoälyn avulla pystytään automatisoimaan verkossa monia toimintoja. Esimerkiksi kyberturvallisuudessa tekoälyä voidaan käyttää kyberhyökkäysten havaitsemiseen ja estämiseen. Tekoälyn nopea kehittyminen on tehnyt myös mahdolliseksi kehittää yhä monimutkaisempia kyberhyökkäyksiä. Hyökkäysten torjumiseen vaaditaan vuosi vuodelta kehittyneempää tekniikkaa, johon myös tekoälyä käytetään. Tekoälyn avulla pystytään vähentämään inhimillisiä virheitä, kuten konfiguroinnin hallintaa. Monia manuaalisia tehtäviä ei aina pysty toistamaan täydellisesti joka kerta, joten tekoälyn avulla pystytään automatisoimaan tehtäviä.

Erilaisia kyberhyökkäystapoja on olemassa valtava määrä. Tässä insinöörityössä syvennytään väliintulo-, eli Man-in-the-middle (MITM) -hyökkäyksiin sekä siihen, miten tekoälyä voitaisiin käyttää hyväkseen kyseisten hyökkäysten havaitsemisessa ja torjumisessa. Ideana on kehittää tekoälymallinnus, jota voitaisiin käyttää apuna tietoturvallisuuden kehittämisessä. Tehokkaan tietoturvan kehittäminen vaatii paljon ammattilaisia ja vuosien kokemusta. Tämän insinöörityön tarkoitus on valmiin, täysin toimivan ohjelman kehittämisen sijaan tutkia tekoälymallin mahdollista rakennetta ja sen toiminnallisuutta. Tämän lisäksi perehdytään tutkimaan, miten tekoäly osaa jo nyt tulkita väliintulohyökkäyksiä.

# 2 Kyberhyökkäys

Kyberhyökkäyksellä tarkoitetaan sähköistä, verkon välityksellä tapahtuvaa rikollisuutta, jolla pyritään esimerkiksi varastamaan tietoja, kuten yrityksien infrastruktuuria, tietoverkkoja ja muita järjestelmiä. Myös yksityishenkilöt tai jopa valtiot voivat olla hyökkäysten kohteena. Hyökkääjänä voi toimia yksilö tai laajempi taho.

## 2.1 Tekoälyn kehitys kyberhyökkäyksissä

Kyberturvallisuuskeskus Traficom on tehnyt tutkimuksen vuonna 2022 liittyen tekoälyn mahdollistamiin kyberhyökkäyksiin. Tekoälymallien kehittyminen on noin viidessä vuodessa mahdollistanut aikaisempaa paremman hyökkäysten automatisoinnin sekä etsimään erilaisia haavoittuvuuksia. Nopea tekoälyn kehittyminen antaa kuvan, että tekoälyä tullaan pian käyttämään hyväksi myös niissä vaiheissa, joita nykyään suoritetaan kyberhyökkäyksissä manuaalisesti. [1.]

Yli 80 % päättäjistä ennusti vuoden 2019 lopulla tehdyssä tutkimuksessa, että tekoälyn mahdollistamat kyberhyökkäykset yleistyvät lähitulevaisuudessa (kuva 1). Monet tiedonkeruumenetelmät sekä käyttäjien manipulointi ovat esimerkkejä nykyisistä tekoälytekniikoista, joita käytetään hyökkäysten alkuvaiheessa. [1.]

Tekoälyn avulla tuodut muutokset pystytään eritellä kolmeen tyyppiin:

1. Ensimmäinen tyyppi käsittelee hyökkäysten automatisointia. Automatisoinnin avulla pystytään esimerkiksi etsimään haavoittuvuuksia, arvaamaan salasanoja tai hankkimaan tunnistetietoja. Automatisoinnilla vähennetään huomattavasti hyökkäysten kestoa, sillä suorituskyky on suurempaa. Tämä myös vähentää kiinnijäämisen riskiä.
2. Toinen tyyppi käsittelee hyökkäysten tehokkuutta. Tekoälyn ansiosta hyökkäyksiä pystytään suorittamaan yhä laajemmassa mittakaavassa. Ensimmäiseen tyyppiin liittyen, automatisoinnin avulla pystytään käynnistämään monia hyökkäyksiä samaan aikaan. Tekoäly pystyy myös räätälöimään kohdennettuja tietojenkalasteluja sopivaksi uhreille.
3. Kolmas tyyppi käsittelee hyökkäysten kattavuutta. Tekoälyn avulla pystytään analysoimaan isoja määriä tietoja, jonka avulla pystytään

löytämään uusia hyökkäystapoja ja pääsyä käsiksi yhä useampaan kohteeseen.


Kuva 1. Tekoälyn käytön kehitys kyberhyökkäyksissä [1].

## 2.2 Väliintulohyökkäys

Tämä insinöörityö syventyy väliintulohyökkäysten havaitsemiseen ja torjumiseen tekoälyn avulla. Tässä hyökkäystavassa hyökkääjä tunkeutuu kahden tai useamman osapuolen välisen viestinnän väliin ja esiintyy usein yhtenä osapuolena keskustelussa.

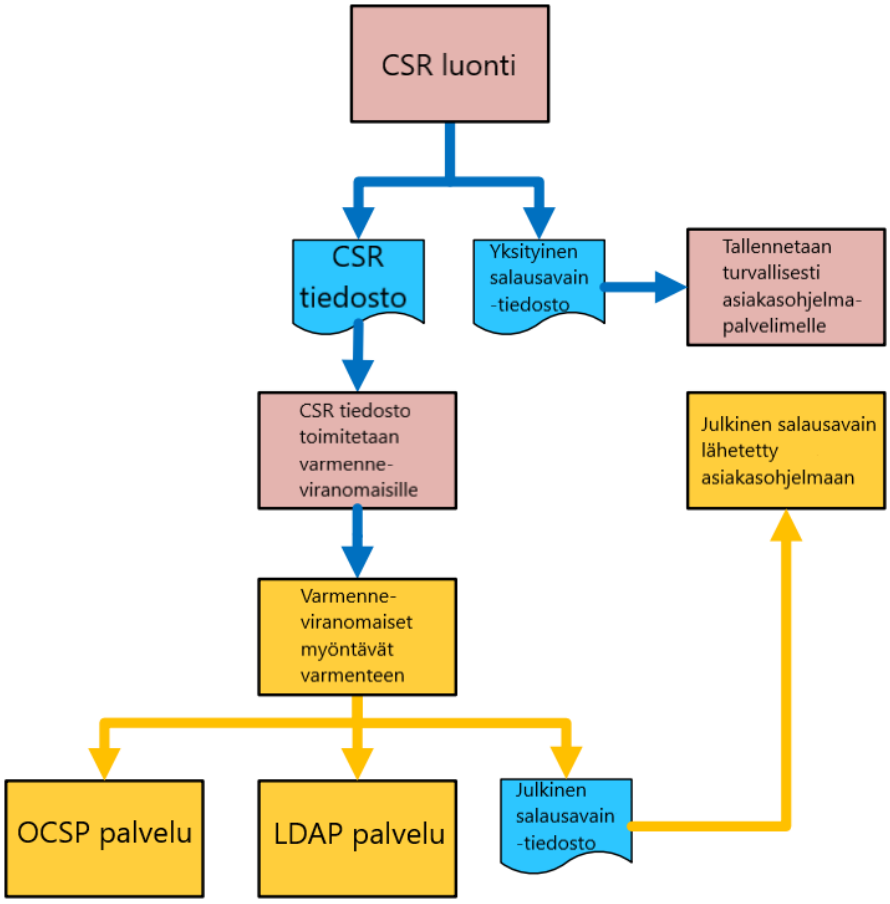
Hyökkääjän tarkoituksena on saada selville esimerkiksi uhrin henkilökohtaisia tietoja, kuten salasanoja, tai asentamaan haittaohjelmia uhrin laitteelle. Suosittu esimerkki väliintulohyökkäyksestä on hyökkäys, jossa hyökkääjä esiintyy verkkopankkina. Uhri syöttää pankkitunnukset väärennettyyn verkkopankkiin, jolloin

hyökkääjä saa talteen uhrin pankkitunnukset ja voi siirtää uhrin omaisuutta haluamalleen tilille.

### 2.2.1 Suojautuminen

Verkossa hyökkäystä vastaan voidaan puolustautua käyttämällä kryptografisia varmenteita (sertifikaatteja). Näiden avulla kummatkin osapuolet todistavat omat henkilöllisyytensä, jolloin hyökkääjä ei pysty imitoimaan toista.

Varmenteet perustuvat julkisen avaimen salaukseen, joista tehdään hierarkkinen infrastruktuuri (kuva 2). Ensiksi asiakasohjelma luo kirjautumisvarmenteen (CSR), jonka jälkeen asiakasohjelmalla on kaksi tiedostoa: CSR-tiedosto ja yksityinen salausavain -tiedosto. Tämän jälkeen CSR-tiedosto lähetetään varmenneviranomaisille sekä yksityinen salausavain tallennetaan asiakasohjelman palvelimelle. Seuraavaksi varmenneviranomaiset tarkastavat varmenteen. Kun varmenne on tarkastettu, julkisen salausavaimen pystyy lataamaan LDAP-palvelusta (Lightweight Directory Access Protocol). OCSP-palvelun (Online Certificate Status Protocol) avulla pystytään tarkistamaan, onko varmenne vielä voimassa. [2.]




Kuva 2. Hierarkkinen infrastruktuuri julkisen avaimen salauksesta [2].

### 2.2.2 Heikkoudet

Vuonna 2014 löydetyn POODLE-hyökkäyksen (Padding Oracle on Downgraded Legacy Encryption) takia nykyaikaiset verkkosivustot ja selaimet pakotetaan käyttämään TLS SSL3.0 -suojausprotokollaa [3]. POODLE:lla tarkoitetaan protokollan alenemista, jonka avulla voidaan hyödyntää vanhentunutta salausmuotoa [3].

TLS 1.3 sisältää ominaisuuden nimeltä 0-RTT Jatkotila. Tämän avulla pystytään tallentamaan esijaetun salausavaimen paikallisesti asiakkaan ja palvelimen välillä. Kun selaimella muodostetaan yhteys palvelimeen uudelleen, pystytään käyttämään samaa salausavainta.

Heikkoutena 0-RTT:ssä on se, että mitään vuorovaikutusta ei vaadita palvelimelta, kun salausavain on ensimmäisen kerran luotu. Tämän takia hyökkääjä pystyy kaappaamaan 0-RTT-tietoja, jonka avulla pystytään lähettämään ne uudelleen palvelimelle. Palvelin pystyy hyväksymään pyynnöt kelvollisiksi, jos palvelin on määritelty väärin. Tällöin hyökkääjä pääsee kuuntelemaan tai kaappaamaan salattuja tietoja. [3.]

### 2.2.3 Esimerkki väliintulohyökkäyksestä

Ensimmäisessä skenaariossa (kuva 3) hyökkääjä käyttää pakettien haistajaa, joka tutkii ja analysoi verkkoliikennettä haavoittuvuuksien löytämiseksi. Uhrin kirjautuessa sivustolle sisään hyökkääjä pystyy hakemaan hänen käyttäjätietonsa, jonka jälkeen uhri voidaan ohjata väärennetylle sivustolle, joka pyrkii matkimaan todellista sivustoa. Väärennetty sivusto kerää talteen uhrin käyttäjätietoja, jonka jälkeen hyökkääjä pystyy käyttämään niitä todellisella sivustolla ja kaapata uhrin tietoja.


Kuva 3. Ensimmäinen skenaario, jossa hyökkääjä esiintyy toisena osapuolena [4].

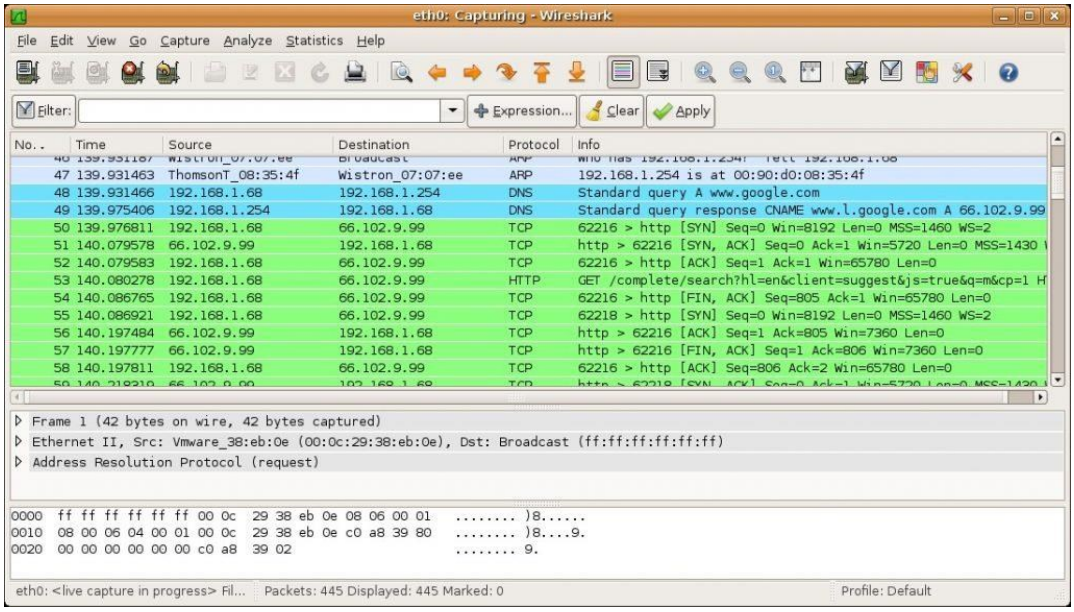
Toisessa skenaariossa (kuva 4) hyökkääjä esiintyy pankin palveluna ja käy keskustelua sekä uhrin että pankin kanssa. Hyökkääjä pystyy käyttämään hyväkseen ensimmäisessä skenaariossa kaapattuja tietoja, jotta imitointi olisi uskottavaa. Kun uhri on antanut tietoja hyökkääjälle, hyökkääjä pystyy aloittamaan keskustelun pankin kanssa imitoimalla uhria. Hyökkäyksen onnistuttua hyökkääjä pääsee käsiksi kohteen tilille.


Kuva 4. Toinen skenaario, jossa hyökkääjä välittää keskustelun osia molemmille osapuolille [4].

### 2.2.4 Pakettien haistajat

Pakettien haistajat keräävät säännöllistä verkkoliikennettä. Niiden avulla pystytään tutkimaan ja havaitsemaan mahdollisia väliintulohyökkäyksiä.

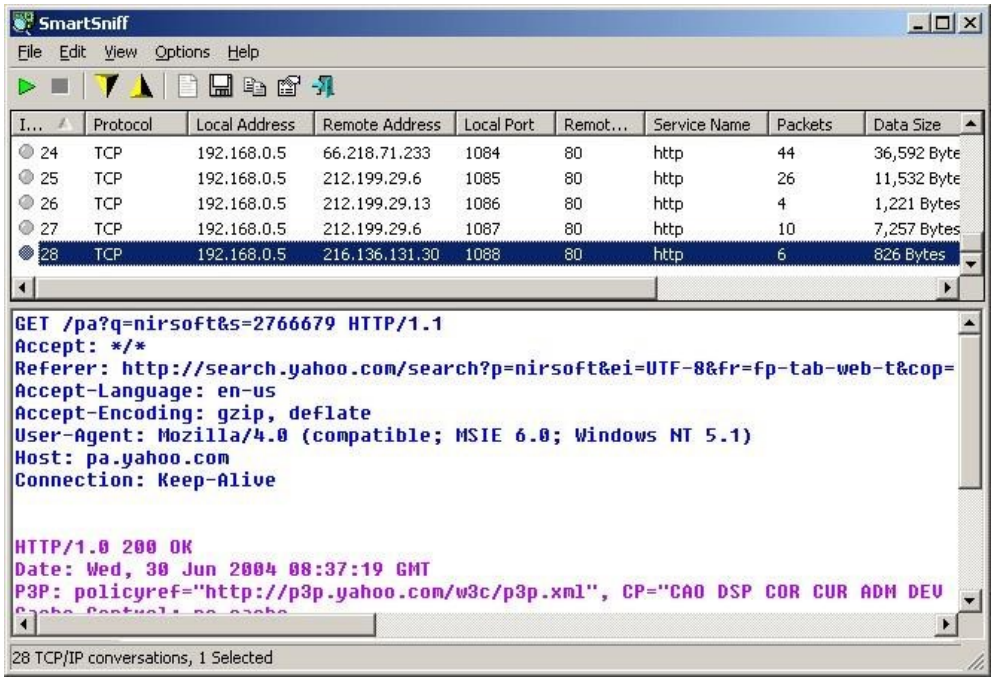
WireShark (kuva 5) on yksi suosituimmista pakettien haistajista. WireShark pystyy analysoimaan live-kaappauksia, tarjoaa kolmiruutuisen pakettiselaimen sekä kykenee syvään protokollan tarkastamiseen. [5.]




Kuva 5. Verkkoliikenteen analysointia WireSharkilla.

Toinen suosittu pakettien haistaja on SmartSniff (kuva 6). SmartSniff pystyy sieppaamaan verkkosovittimen läpi kulkevia TCP/IP-paketteja ja kykenee myös

tarkastelemaan kaapattuja tietoja asiakkaan ja palvelimen välillä keskustelujaksona. [5.]




Kuva 6. Verkkoliikenteen analysointia SmartSniffilla.

# 3 Tekoälyn käyttö väliintulohyökkäyksien torjumisessa

Väliintulohyökkäysten havaitseminen tekoälyllä vaatii paljon verkkoliikenteen ja sen käyttäytymisen analysointia. Tekoälyllä voidaan luoda malleja, jotka oppivat verkon käyttäytymistä ja siten tunnistavat verkon käyttäytymisen poikkeamat, jotka viittaisivat väliintulohyökkäyksiin.

## 3.1 Hyökkäysten havaitseminen

Tekoälyä voidaan opettaa havaitsemaan hyökkäyksiä keräämällä ensin tietojoukko verkkoliikenteestä. Tietojoukon olisi hyvä sisältää normaalin liikenteen lisäksi myös simuloituja väliintulohyökkäyksiä. Verkkoliikenteeseen voi kuulua esimerkiksi lähde- ja kohde IP-osoitteet, pakettikoot, protokollat, portit ja aikaleimat. On tärkeää merkitä tiedot erottamalla ne normaalista liikenteestä ja simuloitujen väliintulohyökkäysten liikenteestä.

Tekoälymallin kouluttamiseen voidaan käyttää koneoppimismalleja. Yleisiä algoritmeja tätä varten ovat satunnaismetsä (Random Forest) [6] ja tukivektorikoneet (Support Vector Machines tai SVM) [7] sekä erikoistuneemmat algoritmit, kuten eristysmetsä (Isolation Forest) [8] ja yksiluokkaiset tukivektorikoneet [9].

Tekoälymalliin tulee toteuttaa reaaliaikainen seurantajärjestelmä. Järjestelmä analysoi tulevaa verkkoliikennettä, jota vertaillaan opittuihin malleihin. Hälytysmekanismin määrittäminen auttaa tietoturvatiimejä ja järjestelmänvalvojia havaitsemaan väliintulohyökkäyksiin liittyvää toimintaa nopeammin.

Palautesilmukan kehittäminen mahdollistaa järjestelmän päivittämisen uusilla tiedoilla parantaakseen suorituskykyä kehittyviä hyökkäystekniikoita vastaan [10]. Tekoälymallia on hyvä kouluttaa ja päivittää säännöllisesti, jotta se pystyy mukautumaan uusiin hyökkäysmalleihin ja käyttäytymisen muutoksiin verkossa.

### 3.2 Hyökkäysten estäminen

Kaikkia tulevia hyökkäyksiä ei välttämättä pysty estämään tekoälymallilla. Verkon segmentointi on hyvä tapa eristää hyökkäyksen kohteena olevat laitteet tai segmentit, jotta isommalta tuholta säästyttäisiin. Segmentoinnilla tarkoitetaan aliverkkojen luomista tietokoneverkkoon. Tämä mahdollistaa hyökkäysten hillitsemistä ja lisää suorituskykyä verkon tehokkuudessa. [11.]

Tekoälyä voisi käyttää hyväkseen analysoimaan hyökkäyksen jälkeisiä seuraamuksia. Esimerkiksi löytämään hyökkäyksen lähteen ja kehittämään tapoja, joilla tulevat hyökkäykset saataisiin torjuttua.

## 4 Tekoälymallinnuksen kehittäminen

Tämän insinöörityön tarkoituksena on luoda tekoälymallinnus, jonka avulla pystyttäisiin kehittämään suojausta väliintulohyökkäyksiä vastaan. Tulen analysoimaan erilaisia rakenteita ja koodiratkaisuja vaihe kerrallaan. Laadukkaan, tekoälyllä toimivan tietoturvan kehittäminen vaatii vuosien kokemusta ja monia alan

ammattilaisia. Tämän työn tarkoituksena onkin antaa ideoita kehittämistä varten sekä tutkia, onko täysin tekoälyllä toimiva ratkaisu kannattavaa.

## 4.1 Suunnittelu

Tekoälyä voidaan opettaa tietojoukoilla. Ensimmäisenä olisi siis hyvä kerätä laaja joukko normaalia verkkoliikennettä sekä esimerkkejä väliintulohyökkäyksistä.

Kerätyistä tietojoukoista olisi sen jälkeen hyvä esikäsitellä kerättyä tietoa. Tähän voi kuulua esimerkiksi kaksoiskappaleiden poistamista tietojoukosta, puuttuvien arvojen lisäämistä sekä eri ominaisuuksien normalisointia.

Tämän jälkeen poimitaan tärkeät tiedot ylös verkkoliikenteestä. Näihin voi kuulua esimerkiksi IP-osoitteet, käytetyt protokollat, pakettien koot sekä SSL-varmennetiedot.

On myös tärkeää merkitä, mitkä näytteet viittaavat väliintulohyökkäyksiin ja mitkä ovat normaalia verkkoliikennettä. Tällöin tekoälymallin kouluttaminen helpottuu huomattavasti.

Oikean koneoppimismallin valinta tekoälymallin kehittämiseen on tärkeä vaihe. Näitä ovat esimerkiksi aiemmin tekstissä mainitut eristysmetsät, yksiluokkaiset tukivektorikoneet sekä satunnaismetsät.

Tekoälymalliin pitäisi myös kehittää ratkaisu käsittelemään virheellisiä analyysejä, joihin voi kuulua esimerkiksi epäilyttävien kohteiden ohilukua tai hyväksymistä. Tällä tarkoitetaan sitä, että malli saattaa tulkita epäilyttävän verkkoliikenteen turvallisena.

Tekoäly vaatii myös reaaliaikaisen seurantajärjestelmän tulevasta verkkoliikenteestä. Tähän kuuluisi myös hälytysjärjestelmä, joka osaa varoittaa mahdollisista hyökkäyksistä.

Tiedonkeruujärjestelmän kehittäminen auttaa jatkokehittämään tekoälymallia. Tätä voisi toteuttaa esimerkiksi keräämällä raportteja hyökkäyksistä.

### 4.1.1 Tietojoukkojen kerääminen

Tietojoukkojen tulisi kuvastaa erilaisia skenaarioita verkkoliikenteestä. Monimuotoisuuden ansiosta tekoälymalli pystyy käsittelemään erilaisia protokollia, verkkoja ja väliintulohyökkäyksiä. Tietojoukon pitäisi toteuttaa suojattavien verkkojen tyypillistä käyttäytymistä ja liikennemalleja. Normaalin verkkotoiminnan kerääminen voisi sisältää esimerkiksi verkkoselailua, tiedostojen siirtoa ja sähköpostiviestintää.

Normaalin verkkotoiminnan lisäksi pitäisi kerätä tietojoukkoja väliintulohyökkäyksistä. Tätä varten joutuisi mahdollisesti kehittämään turvallisten ympäristöjen määrittämistä hyökkäysten suorittamista varten. Muutamia yleisiä hyökkäystapoja, jotka liittyvät väliintulohyökkäyksiin ja joita tekoälymallissa voisi ottaa huomioon, ovat esimerkiksi ARP-huijaus, jolla tarkoitetaan verkkoliikenteen uudelleenohjaamista hyökkääjään lähettämällä uhrille vääriä ARP-viestejä [12], DNS-huijaus, jolla tarkoitetaan uhrin DNS-kyselyiden virheellistä ratkaisua, joka johtaa uhrin väärälle verkkosivulle [13], ja SSL-huijaus, jossa poistetaan HTTPS:n tuoma SSL-salaus, jolloin hyökkääjä pystyy kaappaamaan verkkoja helpommin [14].

Oikeanlaisten työkalujen valinta verkkoliikenteen keräämiseen on tärkeä vaihe. Pakettien analysoimista varten voidaan käyttää erilaisia pakettien haistajia, kuten aikaisemmin mainittua WireSharkia tai SmartSniffia.

Tietoja tulisi myös kerätä laajalta ajanjaksolta. Tällöin voidaan analysoida verkkoliikenteen vaihtelua esimerkiksi vuorokauden, kuukauden tai jopa vuoden aikana. On myös hyvä muistaa, että tietoa pitää kerätä eri tietosuojamääräysten mukaisesti. Kaikki arkaluontoiset tiedot pitää anonymisoida.

Tietojen keräämisen jälkeen täytyy myös kehittää menetelmä, jolla voidaan erottaa normaali verkkoliikenne väliintulohyökkäyksiin liittyvistä liikenteistä.

### 4.1.2 Tietojoukkojen esikäsittely

Tietojoukkojen keräämisen jälkeen on hyvä aloittaa esikäsittely. Tietojoukoista olisi ensin hyvä käsitellä puuttuvia arvoja ja miettiä, mitä niille pitäisi tehdä.

Yksi tapa olisi imputointi eli arvojen arviointi ja niiden lisääminen puuttuvien arvojen tilalle. Jos jokainen tieto poistetaan kokonaan sen perusteella, että siitä puuttuu jokin arvo, voi tietojoukossa esiintyä paljon vaihtelevuutta, eikä tieto ole välttämättä tarpeeksi tarkkaa. [15.] Jos puuttuvia arvoja ei ole paljoa, toinen vaihtoehto on poistaa kyseiset tiedot.

Riippuen minkälaista tietojoukkoa haluaa kerätä, voi joitakin arvoja joutua muuttamaan eri muuttujiksi. Esimerkiksi kategoria (sukupuoli, koulutustaso, ikäryhmä yms.) joudutaan muuttamaan numeeriseksi arvoksi, jotta tekoäly ymmärtäisi tietoa paremmin.

Usein kategorioihin kuuluu moniarvoisia luokkamuuttujia (esimerkiksi Internet-palvelintarjoajat -luokkaan voi kuulua DNA, Elisa, Telia). Tällöin joutuu käyttämään One-Hot koodaamista (One-Hot encoding) [16]. Internet-palvelintarjoajat -esimerkkiluokassa on kolme eri arvoa, joten niitä varten luodaan kolme eri binäärilukua. Taulukko 1 kertoo, kuinka DNA:sta muuttuu arvo 100, Elisasta 010 ja Teliasta 001.

Taulukko 1. Arvoista muutetaan binäärilukuja.

| **DNA** | **Elisa** | **Telia** |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Jos joidenkin tietojen arvot poikkeavat huomattavasti muista, täytyy asialle tehdä jotain. Tietojen poistaminen on yksi vaihtoehto, mutta kuten puuttuvien tietojen käsittelyssä liiallinen poisto voi olla haitaksi. Toinen vaihtoehto voisi olla luoda erillinen luokka tiedoille, jotka poikkeavat paljon muista, jolloin se loisi oman tietojoukon normaalin verkkoliikenteen rinnalle. Kuten aiemmin mainittu, esimerkiksi satunnaismetsä -koneoppimismalli on hyvä työkalu, sillä se pystyy helpommin käsittelemään poikkeavuuksia [17].

### 4.1.3 Väliintulohyökkäysten erittely normaalista verkkoliikenteestä

On tärkeää asettaa erilaisia kriteerejä, minkälainen verkkoliikenne tulkitaan hyökkäykseen osallistuvaksi ja mikä on normaalia verkkoliikennettä. Täytyy myös pitää huoli, että sekä normaaleja tietojoukkoja, että joukkoja hyökkäyksistä on tasainen määrä, jotta tekoäly oppii hyvin tunnistamaan kumpiakin. Joitakin tietoja voi olla vaikea eritellä normaaliksi verkkoliikenteeksi tai hyökkäykseksi. Tässä tapauksessa voi olla järkevää jättää ne erittelemättä, ainakin aikaisessa tekoälymallin kehittämisvaiheessa.

### 4.1.4 Koneoppimisalgoritmin valinta

Oikean koneoppimisalgoritmin valinta on tärkeä vaihe tekoälymallin suunnittelussa. Väliintulohyökkäyksiä varten on hyvä valita algoritmi, joka keskittyy poikkeavuuksiin.

Eristysmetsä on vuonna 2008 kehitetty tekniikka, joka etsii poikkeavuuksia binääripuiden avulla. Käyttö vie vähän muistia ja on hyvä tapa käsitellä suuria määriä tietojoukkoja. Tieto kulkee puussa eteenpäin, jos käsiteltävä tieto täyttää puun asettamat ehdot. Mitä syvemmälle tieto pääsee puussa, sitä pienemmällä todennäköisyydellä kyseinen tieto on poikkeavuus, eli tässä tapauksessa väliintulohyökkäyksiin liittyvää toimintaa. [8.]

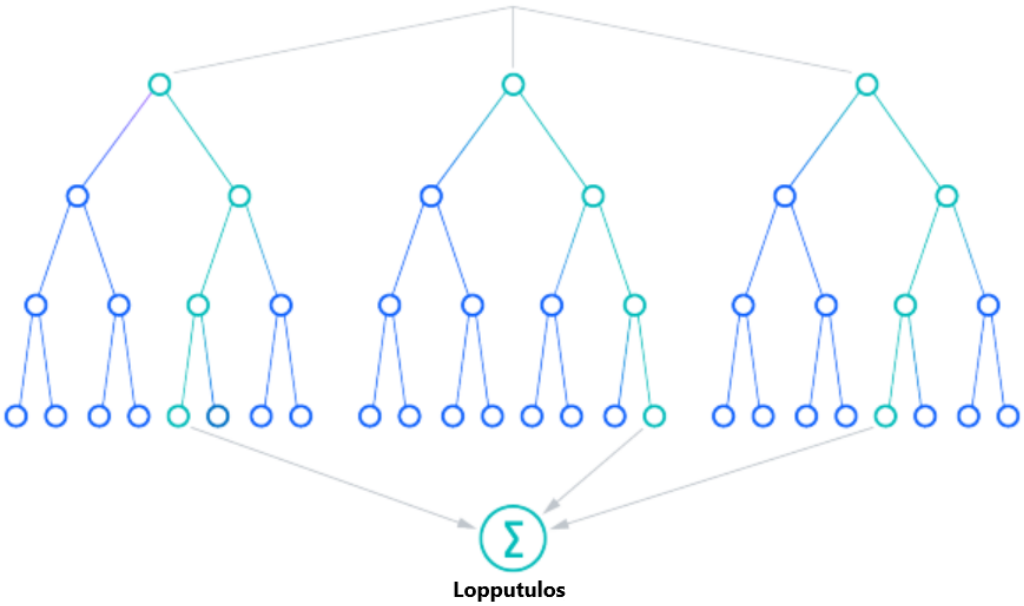
Satunnaismetsä on toinen hyvä algoritmi suodattamaan poikkeavuuksia normaalista verkkoliikenteestä. Algoritmi käyttää poikkeavuuksien suodattamiseen monia puita yhden sijasta, toisin kuin eristysmetsämenetelmässä. Satunnaismetsä käsittelee sekä luokittelu- että regressio-ongelmia. [17.]

Oikean algoritmin valinnassa kannattaa ottaa huomioon, minkälaista tietojoukkoa on käsittelemässä. Verkkoliikennettä tutkittaessa aikasarjapohjaiset lähestymistavat, kuten LSTM (Long Short-Term Memory) -verkko voi olla hyvä valinta [19]. Myös käsiteltävän tietojoukon koko kannattaa ottaa huomioon oikeaa algoritmia valittaessa. ”Väärän” algoritmin käyttö hidastaa tiedon käsittelyä huomattavasti. Useiden eri algoritmien valinta voi myös olla toimiva ratkaisu, sillä tietyntyyppisiin väliintulohyökkäyksiin voi olla parempi tapa käyttää jotain toista algoritmia kuin toisiin.

### 4.1.5 Satunnaismetsä

Tässä insinöörityössä tulen käyttämään esimerkkinä satunnaismetsäalgoritmia. Satunnaismetsä on Leo Breimanin ja Adele Cutlerin kehittämä algoritmi, joka sisältää usein sadoista päätöspuista koostuvan metsän [18].

Päätöspuut rakennetaan solmu kerrallaan testijoukkoa käyttäen. Kun puita on kasvatettu tarpeeksi, voidaan luokitella havaintorivi siihen luokkaan, mikä on kaikkien päätöspuiden yleisin päätös (kuva 7).




Kuva 7. Kuvituskuva satunnaismetsäalgoritmin toimintaperiaatteesta [18].

Satunnaismetsän vahvuudet ovat sen suorituskyky suurissakin aineistoissa. Se kykenee käsittelemään havaintoja, josta puuttuu tietoja, pystyy laskemaan arvion ennustevirheille sekä on hyvä valinta epätasapainoiselle aineistolle. [18.]

### 4.1.6 Tekoälyn opettaminen

Kun tietojoukoista on eroteltu normaali verkkoliikenne ja väliintulohyökkäyksiin viittaava liikenne, on aika alkaa opettamaan tekoälyä havaitsemaan epäilyttävää toimintaa.

Valvotulla oppimistavalla tekoäly oppii valmiiksi eritellyistä tietojoukoista tekemään päätöksiä tulevista, ei eritellyistä tietojoukoista. Ajan kuluessa tekoäly pystyy havaitsemaan eroja ja yhtäläisyyksiä, jolloin se osaa lajitella tulevan verkkoliikenteen turvalliseksi tai ei. [20.]

Tietojoukkoja on hyvä jakaa sekä koulutukseen että arviointiin. Yleisesti joukosta jaetaan 70-80 % kouluttamiseen ja 20-30 % arviointiin. Tämän

tarkoituksena on verrata ja tutkia, onko tekoälyn tuottamat tulokset oikeassa, kun niitä verrataan arviointiin tarkoitettuihin tietojoukkoihin. [21.]

Tekoälyn on hyvä käsitellä tietojoukko läpi useaan otteeseen. Jokaisen läpikäynnin jälkeen voidaan tutkia virheellisten tietojen määrää ja muuttaa eri arvoja, jotta seuraavalla kerralla virheiden määrä vähenisi.

Opettamisen aikana on myös hyvä monitoroida tekoälyn suorituskykyä. Onko esimerkiksi kannattavaa, jos tekoäly alkaa käsittelemään liian yksityiskohtaisesti tietoja, vai viekö se liikaa aikaa? Tarvittaessa on kannattavaa säätää eri parametreja, jotta suorituskyky olisi tehokkaampaa. Olisi myös kannattavaa kehittää tekoälylle toiminta, jolla se osaisi pysäyttää tietojoukon käsittelyn tietyssä vaiheessa, jolloin saataisiin karsittua ”turha oppiminen” ja tehostettua ajanhallintaa.

### 4.1.7 Hälytysjärjestelmä

Yksi oleellisin osa tekoälymallin kehittämisessä on sellaisen hälytysjärjestelmän laatiminen, joka osaa havaita ja hälyttää mahdollisista väliintulohyökkäyksistä.

Ensiksi tulisi asettaa jokin kynnys tai ehto, jonka täyttyessä hälytys suoritetaan. Hälytyksiä voitaisiin myös kategorisoida, mikä riippuu hyökkäyksen laajuudesta, tyypistä ja vaarallisuudesta. Tämä helpottaa manuaalisten toimenpiteiden suorittamista hälytyksen jälkeen.

Tekoälyn olisi hyvä suorittaa toimintoja automaattisesti, mikä riippuu hyökkäyksen tyypistä. Esimerkiksi hyökkäysten kohteena olevien laitteiden eristämistä muusta verkosta.

## 4.2 Toteutus

Toteutusvaiheessa tutkin erilaisia tapoja kehittää tekoälymallia. Käsittelen tietojoukkoja, rakennan koodipohjaa koneoppimismallin avulla, jossa tekoäly oppii paremmin havaitsemaan uhkia sekä käsittelen mahdollista hälytysjärjestelmää.

Toteutuksen tarkoituksena on antaa mallia tekoälypohjaisen tietoturvan rakentamisesta ja kehityksen aloittamisesta. Tarkoituksena ei ole rakentaa valmista mallia. Tässä insinöörityössä ohjelmakoodi tuotetaan Python-ohjelmointikielellä.

### 4.2.1 Tietojoukon keruu ja käsittely

Tietojoukkojen keruu on yksi tärkeimmistä vaiheista tekoälyn oppimisen kannalta. Tietojoukon olisi hyvä sisältää normaalin verkkoliikenteen lisäksi liikennettä mahdollisista hyökkäyksistä. Oman tietojoukon keruu simuloiduilla väliintulohyökkäyksillä on mahdollista ja varmasti kannattavaa, jos tietoturvaa kehitetään esimerkiksi suurelle organisaatiolle. Tässä insinöörityössä käytetään esimerkkinä valmista tietojoukkoa, joka sisältää sekä normaalia verkkoliikennettä että havaintoja hyökkäyksistä.

Kaggle-verkkosivulta löytyy suuri määrä tietojoukkoja, mitä voi käyttää hyväkseen tekoälyn opettamista varten. Kagglesta löytyi erittäin laaja tietojoukko, nimeltään Kitsune Network Attack Dataset [22], joka sisältää miljoonia paketteja ja sisältää myös väliintulohyökkäyksiä.

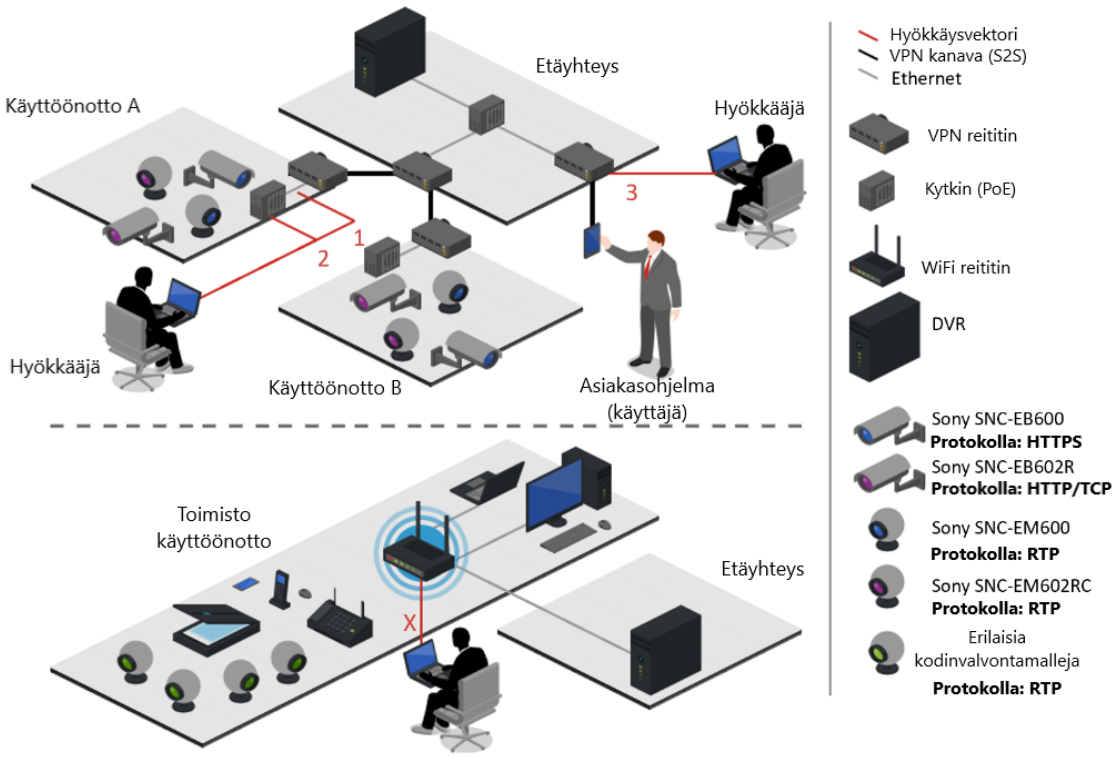
| Attack Type | Attack Name | Tool | Description: *The attacker…* | Violation | Vector | # Packets | Time [min.] |
|---|---|---|---|---|---|---|---|
| *Recon.* | **OS Scan** | Nmap | *…scans the network for hosts, and their operating systems, to reveal possible vulnerabilities.* | **C** | 1 | 1,697,851 | 52.2 |
| | **Fuzzing** | SFuzz | *…searches for vulnerabilities in the camera's web servers by sending random commands to their cgis.* | **C** | 3 | 2,244,139 | 85.5 |
| *Man in the Middle* | **Video Injection** | Video Jack | *…injects a recorded video clip into a live video stream.* | **C, I** | 1 | 2,472,401 | 33.4 |
| | **ARP MitM** | Ettercap | *…intercepts all LAN traffic via an ARP poisoning attack.* | **C** | 1 | 2,504,267 | 28.2 |
| | **Active Wiretap** | Raspberry PI 3B | *…intercepts all LAN traffic via active wiretap (network bridge) covertly installed on an exposed cable.* | **C** | 2 | 4,554,925 | 95.6 |
| *Denial of Service* | **SSDP Flood** | Saddam | *…overloads the DVR by causing cameras to spam the server with UPnP advertisements.* | **A** | 1 | 4,077,266 | 40.8 |
| | **SYN DoS** | Hping3 | *…disables a camera's video stream by overloading its web server.* | **A** | 1 | 2,771,276 | 52.8 |
| | **SSL Renegotiation** | THC | *…disables a camera's video stream by sending many SSL renegotiation packets to the camera.* | **A** | 1 | 6,084,492 | 65.6 |
| *Botnet Malware* | **Mirai** | Telnet | *…infects IoT with the Mirai malware by exploiting default credentials, and then scans for new vulnerable victims network.* | **C, I** | **X** | 764,137 | 118.9 |

Kuva 8. Kerätyt hyökkäykset Kitsune Network Attack -tietojoukosta.

Tämä tietojoukko sisälsi valmiiksi kaksi .csv-tiedostoa, jotka liittyvät väliintulohyökkäyksiin. Ensimmäinen tiedosto sisältää laajan kaappauksen verkkoliikenteestä, jossa väliintulohyökkäys tapahtuu noin miljoonan paketin jälkeen.

Toisessa tiedostossa on yksinkertaistettu, merkitty taulukko, jossa normaali verkkoliikenne esitetään numerolla nolla, ja epäilyttävä liikenne numerolla yksi.

Tietojoukko on kokoelma yhdeksästä eri verkkohyökkäystietojoukosta, jotka on kaapattu joko IP-pohjaisesta, kaupallisesta valvontajärjestelmästä tai täynnä IoT-laitteita olevasta verkosta (kuva 9).




Kuva 9. Havainnollistettu kuva hyökkäyksestä Kitsune Network Attack -tietojoukossa [22].

Tämänkaltaiset, valmiiksi merkityt ja laajat tietojoukot ovat loistavia tekoälyn opettamista varten. Tämä oli vain yksi esimerkki hyvin tuotetusta tietojoukosta. Erityisesti Kagglea kannattaa käyttää hyvien ja luotettavien tietojoukkojen etsimiseen.

Jos käytettävää tietojoukkoa ei ole valmiiksi merkitty (labeling), sen voi tehdä esimerkiksi käyttämällä ”pandas”-kirjastoa.

Esimerkkikoodissa 1 määritetään muuttuja, jonka sisältö (tietojoukko) saadaan ladattua komennolla ”pd.read_csv()”.

```
import pandas as pd

mitm_data = pd.read_csv('dataset_from_kaggle.csv')
```

Esimerkkikoodi 1. Tietojoukon lataaminen [23].

Tämän jälkeen voidaan esimerkkikoodissa 2 poistaa kaksoiskappaleet tietojoukosta [24] sekä muuttaa tieto numeeriseksi [25], jotta tekoäly ymmärtäisi sitä paremmin.

```
mitm_data_cleaned = mitm_data.drop_duplicates()

mitm_data_encoded = pd.get_dummies(mitm_data_cleaned,
columns=['none'])
```

Esimerkkikoodi 2. Kaksoiskappaleiden poistaminen sekä tiedon muuttaminen numeeriseksi.

Seuraavaksi merkitään tietojoukkoon, onko tietyllä aikavälillä hyökkäyksiä vai ei, ja muunnetaan totuusarvot (True tai False) numeeriseksi (0 tai 1). Esimerkkikoodissa 3 ”anomalous”-kolumniin lisätään aikavälillä olevat tiedot joko hyökkäykseksi tai ei ja muunnetaan ne numeeriseksi astype(int)-metodin avulla [26].

```
timestamps = ['2023-09-01 12:00:00', '2023-09-30 12:00:00']

mitm_data_encoded['anomalous'] = mitm_data_encoded['timestamp']
.isin(timestamps).astype(int)
```

Esimerkkikoodi 3. Aikavälillä olevat tiedot merkataan vaaraksi tai ei.

### 4.2.2 Tekoälyn opettaminen

Kun tietojoukko on saatu kerättyä ja esikäsiteltyä, voidaan siirtyä seuraavaan vaiheeseen, jossa alamme opettamaan tekoälyä koneoppimisalgoritmin avulla. Tässä insinöörityössä käytän esimerkkinä satunnaismetsäalgoritmia.

Ensiksi täytyy ladata koneoppimiskirjasto ja tuoda se ohjelmaan. Käytän esimerkkinä suosittua scikit learn -kirjastoa [27]. Esimerkkikoodissa 4 ”train_test_split” -kirjaston avulla pystytään jakamaan tietojoukko tekoälyn kouluttamiseen sekä arviointiin. ”RandomForestClassifier” -kirjastolla voidaan ottaa satunnaismetsäalgoritmi käyttöön. ”accuracy_score”- ja ”classification_report” -kirjastojen avulla voidaan luoda opettamisen tarkkuutta kuvaava raportti.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Esimerkkikoodi 4. Tarvittavien kirjastojen lataaminen.

Kuten aiemmin tekstissä on mainittu, tietojoukko on hyvä jakaa tekoälyn kouluttamiseen ja arviointiin. Scikit learn -kirjaston avulla se onnistuu helposti. Esimerkkikoodissa 5 jaan tietojoukosta 80 % tekoälyn kouluttamiseen ja 20 % arviointiin.

```
x = mitm_data_encoded.drop('label', axis=1)
y = mitm_data_encoded['label']

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
```

Esimerkkikoodi 5. Tietojoukon jakaminen tekoälyn kouluttamiseen ja arviointiin [28].

Esimerkkikoodissa 6 luodaan satunnaismetsä. Käytän esimerkissä sata puuta. Tämän jälkeen voimme aloittaa kouluttamaan tekoälyä kouluttamiseen jaetun tietojoukon avulla käyttämällä fit()-metodia.

```
random_forest = RandomForestClassifier(n_estimators=100,
random_state=42)

random_forest.fit(x_train, y_train)
```

Esimerkkikoodi 6. Satunnaismetsän luonti ja kouluttamisen käynnistäminen [29].

Esimerkkikoodissa 7 voimme puiden avulla ennustaa sekä arvioida lopputulosta.

```
y_pred = random_forest.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

Esimerkkikoodi 7. Lopputuloksen ennustusta ja arviointia [29].

Kun olemme keränneet ja esikäsitelleet tietojoukon sekä kouluttaneet tekoälyä algoritmin avulla, on aika miettiä, miten mallia voisi kehittää paremmaksi.

Esimerkiksi satunnaismetsäalgoritmin eri parametreilla voi saada parempia lopputuloksia. Tekoälymallia kehittäessä on siis hyvä testata algoritmia eri arvoilla. Esimerkiksi lisäämällä tai vähentämällä puiden määrää ”n_estimators” -komennon avulla tai muuttamalla puiden syvyyttä ”max_depth”-komennolla.

### 4.2.3 Monitoroinnin kehittäminen

Jotta tekoälymalli toimisi reaaliaikaisesti, täytyy sitä varten kehittää monitorointisysteemi. Käytän esimerkkinä jälleen scikit learn -kirjaston tarjoamia ominaisuuksia.

Ensiksi olisi hyvä toteuttaa funktio, jonka avulla monitorointia pystytään suorittamaan. Esimerkkikoodissa 8. käytämme hyväkseen aikaisemmin käytettyjä ”accuracy_score”- ”ja classification_report” -toimintoja.

```
def mitm_model_monitor(model, x_new, y_new):
     y_pred = model.predict(x_new)
     accuracy = accuracy_score(y_new, y_pred)
     report = classification_report(y_new, y_pred)
     print(f’Accuracy: {accuracy}’)
     print(‘Classification Report:’)
     print(report)
```

Esimerkkikoodi 8. Monitorointifunktion toteutus.

Tämän jälkeen voidaan asettaa funktio toteutumaan tietyin aikavälein. Esimerkkikoodissa 9 suoritan funktion tunnin välein. Tätä varten olisi myös hyvä kehittää funktio, joka lataa ja esikäsittelee uuden tietojoukon, joka on syntynyt tunnin aikana. Olkoon tässä esimerkissä kyseisen funktion nimi ”data_preprocess”.

```
while True:
     time.sleep(3600)
     x_new, y_new = data_preprocess()
     mitm_model_monitor(random_forest, x_new, y_new)
```

Esimerkkikoodi 9. Monitorointifunktio suoritetaan tunnin välein.

### 4.2.4 Hälytysjärjestelmän kehittäminen

Tekoälyn olisi hyvä osata huomauttaa, jos se havaitsee epäilyttävää verkkoliikennettä. Pohdin pitkään, millaisilla tavoilla hälytys kannattaa suorittaa. Tulin tulokseen, että vähintään sähköposti-ilmoitus on välttämättömyys. Monessa IT-yrityksessä käytetään myös Slack-alustaa, joten Slack-ilmoituksen luominen olisi myös kannattavaa. Olisi myös hyvä kehittää systeemi, jossa tekoäly tarkistaa, onko verkkoliikenne oikeasti uhka vai ei, sekä vaaran havaittua tekee toimenpiteitä kyseiselle lähteelle, esimerkiksi estämällä hyökkäyksen lähteen IP-osoite sekä katkaisemalla yhteydet hyökkäyksen kohteena oleviin laitteisiin.

Aloitin hälytysjärjestelmän sähköposti-ilmoitustoiminnallisuuden kehittämisellä. Python tarjoaa ”smtplib” -moduulin, jonka avulla ilmoituksien luominen onnistuu [30]. Esimerkkikoodissa 10 kehitin funktion, joka toteuttaa sähköpostin lähettämisen. Sen jälkeen funktiota täytyy vain kutsua.

```
import smtplib

def email_alert(subject, body):
     sender = 'sender@example.com'
     receiver = 'receiver@example.com'
     password = 'password'

     message = f"Subject: {subject}\n\n{body}"

     with smtplib.SMTP('smtp.example.com', 465) as server:
          server.starttls()
          server.login(sender, password)
          server.sendmail(sender, receiver, message)

email_alert("Alert: Possible MITM-attack detected!")
```

Esimerkkikoodi 10. Ilmoituksen luominen ja sen lähettäminen.

Esimerkkikoodissa 11 kehitin Slack-ilmoituksen Slack SDK:n tarjoamilla toiminnoilla [31].

```
from slack_sdk import WebClient

def slack_notification(token, channel, message):
     client = WebClient(token=token)
     response = client.chat_postMessage(channel=channel, text=message)

slack_notification('your_token', '#incidents', 'Possible MITM-attack
detected!')
```

Esimerkkikoodi 11. Slack-ilmoituksen kehittäminen ja sen lähettäminen.

Mahdollisen hyökkäyksen tarkistusta varten pitäisi kehittää jonkinlainen anomaly-olio sekä funktio, joka esimerkiksi estää hyökkäyksen lähteen IP-osoitteen sekä suorittaa aikaisemmin luodut hälytysfunktiot. Esimerkkikoodissa 12 kehitin rungon hyökkäysten tarkistamista varten.

```
def threat_check(anomaly):
     if anomaly['type'] == 'False Positive'
          return False
     else:
          return True

def threat_mitigation(anomaly):
     if anomaly['type'] == 'Malicious':
          block_ip (anomaly['attacker_ip_address'])
          email_alert("Alert: Possible MITM-attack detected!")
          slack_notification('your_token', '#incidents', 'Possible
          MITM-attack detected!')
```

Esimerkkikoodi 12. Rungon kehittäminen hyökkäysten tarkistamista varten.

Toteutin edellä mainitussa koodissa olevan ”block_ip” -funktion esimerkkikoodissa 13 käyttäen paramiko-kirjastoa [32].

```
import paramiko

def block_ip(firewall_ip, user, password, blocked_ip):
     ssh_client = paramiko.SSHClient()
     ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

     try:
          ssh_client.connect(firewall_ip, user=user, password=password)
          command = f'block {blocked_ip}'
          stdin, stdout, stderr = ssh_client.exec_command(command)
          print(f'{blocked_ip} has been blocked.')
     except Exception as e:
          print(f'Error: {e}')
     finally:
          ssh_client.close()
```

Esimerkkikoodi 13. Funktio IP-osoitteen estämiseen [32].

### 4.2.5 Varmuuskopiointi ja palautus

Malliin olisi hyvä kehittää toimintoja varmuuskopiointia ja palautusta varten hyökkäyksen tapahtuessa. Shutil-kirjaston [33] avulla pystytään kopiointi toteuttamaan yksinkertaisesti. Esimerkkikoodissa 14 luotu ”backup”-funktio voitaisiin myös liittää esimerkkikoodiin 12, jolloin tekoäly suorittaa varmuuskopioinnin havaittuaan hyökkäyksen. Saman funktion voisi tämän lisäksi lisätä esimerkkikoodiin 9, jolloin varmuuskopiointi suoritetaan tietyin aikavälein.

```
import shutil

def backup(source, destination):
     try:
          shutil.copytree(source, destination)
          print(f”Data copied from {source} to {destination}”)
     except Exception as e:
          print(f”Error: {e}”)

backup(‘/path/to/source’, ‘/path/to/destination’)
```

Esimerkkikoodi 14. Varmuuskopiointi shutil-kirjaston avulla [33].

Verkon konfiguroinnin varmuuskopiointia varten voimme myös käyttää aiemmin mainittua paramiko-kirjastoa. Esimerkkikoodissa 15 luotu ”backup_config” -funktio voidaan myös liittää esimerkkikoodiin 9 ja 12, kuten esimerkkikoodissa 14. luotu ”backup”-funktio.

```
import paramiko

def backup_config(host, user, password):
     ssh_client = paramiko.SSHClient()
     ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
     try:
          ssh_client.connect(host, user=user, password=password)
          stdin, stdout, stderr = ssh_client.exec_command(‘command’)
          config = stdout.read()
          with open(f’{host}_config.txt’, ‘wb’) as config_file:
               config_file.write(config)
          print(f’{host} configuration backed up.’)
     except Exception as e:
          print(f’Error: {e}’)
     finally:
          ssh_client.close()

backup_config(‘router.something.com’, ‘user’, ‘password’)
```

Esimerkkikoodi 15. Rakenne verkon konfiguroinnin varmuuskopioimisesta [32].

## 4.3 Jatkokehittäminen

Edellä mainittujen askeleiden jälkeen tekoälymalli on suunniteltu perustoiminnallisuuksiltaan. Mallin kehittäminen on pitkä prosessi ja jatkokehittämällä siitä rakentuu askel kerrallaan tehokkaampi kokonaisuus. Keräsin muutamia ideoita, joita voisi ottaa huomioon mallin jatkokehittämisen kannalta.

Ensimmäinen kehityksen kohde olisi täydentää tietojoukkoja. Tätä voisi tehdä keräämällä täysin uusia tietojoukkoja, jotka viittaavat väliintulohyökkäyksiin. Kagglea voi hyödyntää tässäkin, tai jopa simuloimalla itse hyökkäyksiä, kunhan ottaa turvatoimet huomioon.

Valmiita, verkosta löytyviä malleja voisi myös käyttää hyödykseen oman tekoälymallin kehityksessä hienosäätämällä niitä haluamallaan tavalla tai vain keräämällä lisää tietoa erilaisten väliintulohyökkäystyyppien toimintatavoista ja niiden torjumisesta. Löysin GitHubista ”frostbits-security”-tiimin kehittämän listan tunnetuista väliintulohyökkäystyypeistä ja tavoista, joilla niitä voisi torjua [34]. Esimerkiksi tätä listaa käyttäen omaa tekoälymallia voisi kehittää havaitsemaan ja torjumaan väliintulohyökkäyksiä yhä laajemmin.


Kuva 10. Luettelo ”frostbits-security” -tiimin GitHub-listasta.

Myös simuloitujen hyökkäysten avulla pystytään tutkimaan oman mallin toiminnallisuutta. Näitä testejä on tärkeä suorittaa tekoälymallin kehityksen aikana, hyvä dokumentointi on tärkeä apu jatkokehityksen kannalta.

Mallin kehittyessä ja kouluttamisessa olisi myös hyvä ottaa huomioon optimointi. Onko esimerkiksi kannattavaa käyttää samaa koneoppimisalgoritmia tietojoukkojen ja verkkoliikenteen kasvaessa suureksi tai onko koodin rakenne paras mahdollinen sen kasvaessa?

Ylipäätänsä jatkuva tiedonhaku väliintulohyökkäyksistä auttaa kehittämään tekoälymallia monella tavalla. Myös vuorovaikuttaminen muiden aiheista kiinnostuneiden tai alan ammattilaisten kanssa on hyvä keino oman tekoälymallin jatkokehittämisessä.

## 4.4 Arviointi

Nostin työssäni esitetyn Kaggle-aineiston esille vain esimerkkinä enkä käyttänyt sitä tekoälyn opettamisessa. Tiedoston koko oli yli seitsemän gigatavua, mikä aiheutti ohjelmakoodin kaatumisen. Kokeilin kehittämälläni ohjelmakoodilla tietojoukkoa, jota myös Jenny Kreiger käytti tekemässään tutkimuksessa [35].

Kreigerin tekemässä tutkimuksessa käytettiin tietojoukkoa, joka sisälsi yli puoli miljoonaa havaintoa. Tutkimuksessa käytettiin samoja metodeja, mitä olin itse laatinut mallinnukseeni. Satunnaismetsän avulla tarkkuuspisteeksi (accuracy score) saatiin 94 %, joka kertoo satunnaismetsän olevan melko tarkka. Tarkkuuspisteiden mittaaminen ei silti ole täydellinen tapa tutkia algoritmin tarkkuutta. Algoritmi ei välttämättä suoriudu yhtä tarkasti jokaisessa käsiteltävässä luokassa. Mallin tarkkuutta pystytään tutkimaan tarkemmin ottamalla huomioon myös täsmällisyys (precision), herkkyys (recall) sekä F1-pisteet.

Täsmällisyydellä tarkoitetaan oikein tunnistettujen kohteiden luokittelua tiettyyn luokkaan jaettuna kaikilla kerroilla, kun malli ennusti kyseistä luokkaa.

Herkkyydellä tarkoitetaan mallin oikein tunnistettujen kohteiden luokittelua tiettyyn luokkaan jaettuna luokan kohteiden kokonaismäärällä.

F1-pisteillä tarkoitetaan täsmällisyyden ja herkkyyden yhdistelmää. Esimerkiksi jos täsmällisyys ja tarkkuus on iso, myös F1-pisteet ovat isoja. Jos vain toinen on iso, F1-pisteet ovat pienet.

Nämä tulokset saadaan esille aikaisemmin mallissani käytetyllä ”classification_report”-metodin avulla. Kreigerin tutkimuksessa tulokset olivat suurilta osin tarkkoja. Monessa luokassa täsmällisyys, herkkyys ja F1-pisteet olivat yli 92 %. Ajoin työssäni kehittämälläni ohjelmakoodilla saman tietojoukon ja ”classification_report”-metodin perusteella (kuva 11) satunnaismetsä on toimiva vaihtoehto oikean koneoppimisalgoritmin valitsemisessa omaan malliin. Tulokset olivat suurin piirtein yhtä tarkkoja Kreigerin tuloksiin nähden. Valitettavasti en löytänyt sopivaa tietojoukkoa testattavaksi, joka liittyisi väliintulohyökkäyksiin, mutta ainakin satunnaismetsä vaikuttaisi toimivan tarpeeksi hyvin.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.94 | 0.94 | 0.94 | 52960 |
| 2 | 0.95 | 0.95 | 0.95 | 70825 |
| 3 | 0.92 | 0.95 | 0.94 | 8934 |
| 4 | 0.93 | 0.84 | 0.89 | 686 |
| 5 | 0.94 | 0.72 | 0.83 | 2375 |
| 6 | 0.92 | 0.83 | 0.88 | 4344 |
| 7 | 0.97 | 0.92 | 0.95 | 5127 |
| micro avg | 0.94 | 0.94 | 0.94 | 145251 |
| macro avg | 0.94 | 0.88 | 0.92 | 145251 |
| weighted avg | 0.94 | 0.94 | 0.94 | 145251 |

Kuva 11. Satunnaismetsän tarkkuus kehittämälläni ohjelmakoodilla.

Edellä mainittujen toteutus- sekä jatkokehitysvaiheiden jälkeen olen mielestäni saavuttanut vakaan mallinnuspohjan tekoälyn käyttöä varten tietoturvallisuudessa. Mallinnuksen tarkoituksena on antaa ideoita ja apua tekoälymallin kehitykseen. Muutama kuukausi ei riitä täydellisen tietoturvan kehittämiseen. Myös

tekoälyn opettaminen ja sen kehittyminen vaatii paljon aikaa sekä lukuisten tietojoukkojen analysoimista ja käsittelyä.

Uskon, että tämä insinöörityö antaa tarpeeksi hyvän käsityksen, mitä toimenpiteitä tekoälymallin kehittäminen vaatii ja mistä olisi hyvä aloittaa. Tekstissä käsittelemäni asiat eivät ole ainoita oikeita ratkaisuja mallin kehittämiseen, ja uskon, että moni vaiheista sopii muidenkin kuin väliintulohyökkäysten käsittelemiseen.

# 5 Yhteenveto

Tekoäly on kehittynyt viime vuosien aikana nopeaa tahtia. Kyberhyökkäysten määrä sekä niiden kompleksisuus jatkaa kasvuaan. Tekoäly on tullut yhä suurempaan käyttöön niin hyökkäyksissä kuin niiden puolustamisessa.

Tätä insinöörityötä tehdessä sain laajemman käsityksen väliintulohyökkäyksistä, sekä miten tekoälyä voisi hyödyntää niiden havaitsemisessa sekä torjumisessa. Toteuttamani mallinnus antaa kuvan tekoälypohjaisen mallin rakenteesta, sekä millaisia käytäntöjä ja vaiheita olisi hyvä toteuttaa mallin kehittämisessä. Näiden vaiheiden sekä laatimieni jatkokehitysideoiden avulla tekoälymallin kehittäminen on mahdollista.

On vaikeaa sanoa, kuinka relevanttia hakemani tieto on esimerkiksi kymmenen vuoden päästä, sillä tekoälyn ja teknologian kehitys kasvaa ja muuttuu nopeaa tahtia. Uskon silti, että tekoälyä tullaan käyttämään yhä enemmän ja monimuotoisemmin tulevaisuudessa.

## Lähteet


1 Kyberturvallisuuskeskus. 2022. Tekoälyn mahdollistamat kyberhyökkäykset. Verkkoaineisto <https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/publication/TRAFICOM_Teko%C3%A4lyn_mahdollistamat_kyberhy%C3%B6kk%C3%A4ykset%202022-12-12_web.pdf> Luettu 5.9.2023.

2 Swedbank. 2018. PKI in a nutshell. Verkkoaineisto <https://web.archive.org/web/20180121095724/https://www.swedbank.ee/download/gateway/PKI-in-nutshell.pdf> Luettu 6.9.2023.

3 Naziridis, N. 2021. Verkkohyökkäykset ja tietoturvaongelmat. Verkkoaineisto <https://www.ssl.com/fi/FAQ/verkkohy%C3%B6kk%C3%A4ykset-ja-turvallisuuskysymykset/> Luettu 8.9.2023.

4 Veracode. Man in the Middle (MITM) Attack. Verkkoaineisto <https://www.veracode.com/security/man-middle-attack> Luettu 8.9.2023.

5 WebTech 360. Pakettihaistarit: mitä ne ovat? Muiden mahtavien pakettihaistelutyökalujen ohella. Verkkoaineisto <https://blog.webtech360.com/fi/miten/pakettihaistarit-mita-ne-ovat-muiden-mahtavien-pakettihaistelutyokalujen-ohella/77700528> Luettu 9.9.2023.

6 Farnaaz, N., Jabbar, M.A. 2016. Random Forest Modeling for Network Intrusion Detection System. Verkkoaineisto <https://www.sciencedirect.com/science/article/pii/S1877050916311127> Luettu 12.9.2023.

7 Ghanem, K., Aparicio-Navarro, F., Kyriakopoulos, K., Lambotharan, S., Chambers, J. 2017. Support Vector Machine for Network Intrusion and Cyber-Attack Detection. Verkkoaineisto <https://ieeexplore.ieee.org/document/8233268> Luettu 12.9.2023.

8 Analytics Vidhya. 2023. Anomaly detection using Isolation Forest – A Complete Guide. Verkkoaineisto <https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/> Luettu 12.9.2023.

9 Zhang, M., Xu, B., Gong, J. 2015. An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrustions. Verkkoaineisto <https://ieeexplore.ieee.org/document/7420931> Luettu 12.9.2023.

10 Truvantis. 2022. Combating Feedback Loops with Attack Surfave Analysis. Verkkoaineisto <https://www.truvantis.com/blog/combating-feedback-loops-with-attack-surface-analysis> Luettu 14.9.2023.

11 The Astrology Page. 2023. Mikä on verkon segmentointi? – määritelmä techopediasta. Verkkoaineisto <https://fi.theastrologypage.com/network-segmentation> Luettu 15.9.2023.

12 The Astrology Page. 2023. Mikä on arp-huijaus? – määritelmä techopediasta. Verkkoaineisto <https://fi.theastrologypage.com/address-resolution-protocol-spoofing> Luettu 15.9.2023.

13 Kaspersky. Mikä on DNS-kaappaus? verkkoaineisto <https://www.kaspersky.fi/resource-center/definitions/what-is-dns-hijacking> Luettu 15.9.2023.

14 Arampatzis, A. 2022. What Are SSL Stripping Attacks? Verkkoaineisto <https://venafi.com/blog/what-are-ssl-stripping-attacks/> Luettu 15.9.2023.

15 Simplilearn. 2023. Introduction to Data Implusion. Verkkoaineisto <https://www.simplilearn.com/data-imputation-article#:~:text=ProgramExplore%20Program-,What%20Is%20Data%20Imputation%3F,from%20a%20dataset%20each%20time.> Luettu 20.9.2023.

16 Brownlee, J. 2020. Why One-Hot Encode Data in Machine Learning? Verkkoaineisto <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/> Luettu 26.10.2023.

17 Srivastava, A. 2021. Let’s Talk about Random Forests! Verkkoaineisto <https://medium.com/analytics-vidhya/lets-talk-about-random-forests-524ae1138d8b> Luettu 20.9.2023.

18 IBM. What is random forest? Verkkoaineisto <https://www.ibm.com/topics/random-forest?mhsrc=ibmsearch_a&mhq=random%20forest> Luettu 20.9.2023.

19 Saxena, S. 2023. What is LSTM? Introduction to Long Short-Term Memory. Verkkoaineisto <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/#:~:text=Bidirectional%20LSTMs%20(Long%20Short%2DTerm,based%20on%20the%20preceding%20context.> Luettu 23.9.2023.

20 SAP. Mitä koneoppiminen on? Verkkoaineisto <https://www.sap.com/finland/products/artificial-intelligence/what-is-machine-learning.html> Luettu 24.9.2023.

21 Gillis, A. 2022. data splitting. Verkkoaineisto <https://www.techtarget.com/searchenterpriseai/definition/data-splitting> Luettu 24.9.2023.

22 Kaggle. 2020. Kitsune Network Attack Dataset. Verkkoaineisto <https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune> Luettu 25.9.2023.

23 pandas. How do I read and write tabular data? Verkkoaineisto <https://pandas.pydata.org/docs/getting_started/intro_tutorials/02_read_write.html> Luettu 27.9.2023.

24 pandas. pandas.DataFrame.drop_duplicates. Verkkoaineisto <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html> Luettu 1.10.2023.

25 pandas. pandas.get_dummies. Verkkoaineisto <https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html> Luettu 1.10.2023.

26 pandas. pandas.DataFrame.astype. Verkkoaineisto <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html> Luettu 1.10.2023.

27 scikit learn. Verkkoaineisto <https://scikit-learn.org/stable/index.html> Luettu 3.10.2023.

28 scikit learn. sklearn.model_selection.train_test_split. Verkkoaineisto <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html> Luettu 3.10.2023.

29 scikit learn. sklearn.ensemble.RandomForestClassifier. Verkkoaineisto <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> Luettu 3.10.2023.

30 Kontorskyy, D. 2022. How to Send Emails in Python with Gmail. Verkkoaineisto <https://mailtrap.io/blog/python-send-email-gmail/> Luettu 5.10.2023.

31 Slack. Python Slack SDK. Verkkoaineisto <https://slack.dev/python-slack-sdk/> Luettu 5.10.2023.

32 Paramiko. Paramiko’s documentation. Verkkoaineisto <https://docs.paramiko.org/en/latest/> Luettu 8.10.2023.

33 Python. shutil. Verkkoaineisto <https://docs.python.org/3/library/shutil.html> Luettu 8.10.2023.

34 frostbits-security. 2021. MITM-cheatsheet. Verkkoaineisto <https://github.com/frostbits-security/MITM-cheatsheet/> Luettu 15.10.2023.

35 Kreiger, J. 2020. Evaluating a Random Forest model. Verkkoaineisto <https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56> Luettu 29.10.2023.