



## **OHJELMISTOHAAVOITTUVUUDET TUTKIMUKSISSA JA KÄYTÄNNÖSSÄ**

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintyö

2023

Lassi Laukkarinen

Tarkastaja(t): Tutkijaopettaja Jouni Ikonen

## TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tietotekniikka

Lassi Laukkarinen

### **Ohjelmistohaavoittuvuudet tutkimuksissa ja käytännössä**

Tietotekniikan kandidaatintyö

30 sivua, 7 kuvaa, 1 taulukko ja 1 liite

Tarkastaja(t): Tutkijaopettaja Jouni Ikonen

Avainsanat: haavoittuvuus, ohjelmistohaavoittuvuus, tietoturva, kirjallisuuskatsaus, NVD

Tässä kandidaatintyössä verrataan NVD-tietokannassa hyväksikäytetyiksi kirjattujen ohjelmistohaavoittuvuuksien kategorioita niihin kohdistetun tieteellisen tutkimuksen määrään. Työssä tehdään systemaattinen kirjallisuuskatsaus vertaisarvioituista tieteellisistä teoksista sekä tilastollinen analyysi vapaasti saatavilla olevasta haavoittuvuusdatasta. Työn tavoitteena on selvittää, tutkitaanko tieteellisessä kirjallisuudessa samoja ohjelmistohaavoittuvuuskattegorioita, joita todellisuudessa tapahtuu.

Työn tuloksena havaittiin, että ohjelmistohaavoittuvuuksien tutkimuksessa keskitytään pääasiassa samassa suhteessa samoihin haavoittuvuustyyppisiin kuin mitä havaitaan oikeassa maailmassa vuosina 2018 - 2022. Muistinkäsittelyn virheet ja injektiot olivat kaksi suurinta haavoittuvuuskattegoriaa molemmissa lähteissä. Tarkastelluissa tutkimuksissa syötön validointi oli aliedustettu haavoittuvuuskattegoria ja toiminnonkulun virheet oli puolestaan yliedustettu kattegoria verrattuna samana aikana tapahtuneisiin todellisiin haavoittuvuuksiin.

## ABSTRACT

Lappeenranta–Lahti University of Technology LUT

School of Engineering Science

Software Engineering

Lassi Laukkarinen

### **Software vulnerabilities in research and practice**

Bachelor's thesis

2023

30 pages, 7 figures, 1 table and 1 appendix

Examiners: Associate professor Jouni Ikonen

Keywords: vulnerability, software vulnerability, cybersecurity, literature review, NVD

This bachelor's thesis compares the distribution of scientific research on software vulnerability categories with the distribution of vulnerabilities in the NVD database that have been reported as having been exploited in a data breach. The thesis consists of a systematic literature review on peer-reviewed scientific articles and a statistical analysis of freely available software vulnerability data.

It was found that the types of software vulnerabilities examined in scientific articles are similar to the types of exploited vulnerabilities during the years 2018-2022. Memory errors and injections were the two largest vulnerability categories in both cases. In scientific articles input validation was found to be underrepresented and control-flow errors were found to be overrepresented.

## Sisällysluettelo

Tiivistelmä

Abstract

1	Johdanto.....	5
2	Aiempi tutkimus .....	8
3	Tutkimusmenetelmä ja työn toteutus .....	10
3.1	Systemaattinen kartoitustutkimus.....	10
3.1.1	Valittujen aineistoiden kuvaus.....	16
3.2	Tilastodataan pohjautuva kategorisointi .....	21
4	Tulokset ja huomiot .....	23
5	Yhteenveto .....	29
	Lähteet .....	31

Liitteet

Liite 1. Python-lähdekoodi datan keräämiseen NIST NVD-tietokannasta

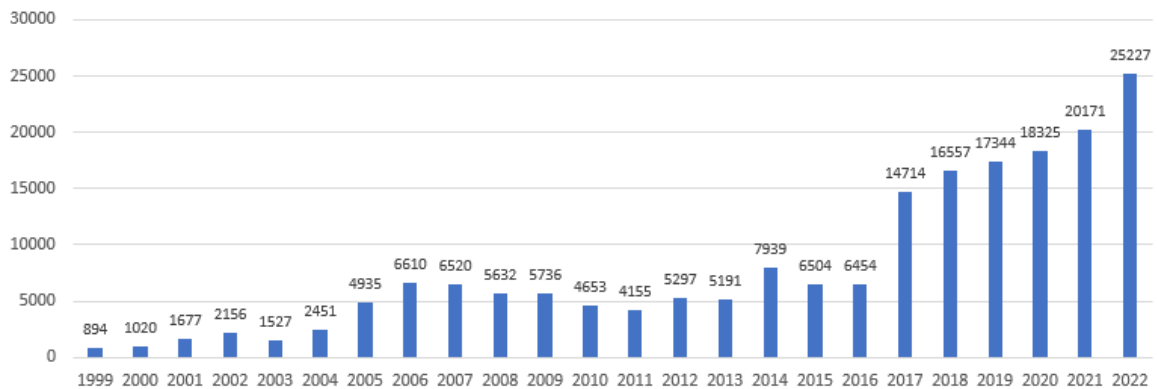
# 1 Johdanto

Hyvän tietoturvan tärkeys ohjelmistokehityksessä on kasvanut sitä mukaan, kun ohjelmistot enemmässä määrin kommunikoivat toistensa kanssa ja käsittelevät arvokasta tai arkaluontoista dataa. Nykypäivän maailmassa tietoturvan rooli on monimuotoinen: uhkatekijät voivat olla organisaation sisäisiä tai ulkoisia, fyysisiä tai digitaalisia, sosiaalista manipulointia tai ohjelmistoheikkouksien hyväksikäyttämistä. Ohjelmistojen tietoturvan varmistaminen on loppujen lopuksi sen tekijöiden omalla vastuulla, ja tietoturvan rooli on sitä tärkeämpi, mitä arvokkaampaa tietoa ohjelmisto käsittelee. Tämä työ tarkastelee tietoturvaa ohjelmistojen haavoittuvuuksien näkökulmasta tutkimalla vastaako tieteellisessä kirjallisuudessa tarkastellut haavoittuvuustyypit oikeasti tapahtuneiden tietomurtojen taustalla olleita haavoittuvuuksia.

Suurin englanninkielinen taho, joka pitää yllä tietokantaa ohjelmistohaavoittuvuuksista, on MITRE. Tietokanta on nimeltään Common Vulnerability Enumeration (CVE), ja sitä on pidetty yllä jo vuodesta 1999. Sen tavoitteena on listata kaikki julkisesti tunnetut tietoturva- haavoittuvuudet mahdollisimman kattavasti. MITRE määrittelee ohjelmistohaavoittuvuuden ohjelmistoheikkoudesta johtuvaksi viaksi ohjelmassa tai palvelussa, jonka hyväksikäyttö voi aiheuttaa negatiivisen vaikutuksen kyseessä olevaan komponenttiin. (CVE.org 2022)

Toinen aiheeseen liittyvä, mutta erillinen taho on National Institute of Standards and Technology (NIST). He ovat pitäneet yllä haavoittuvuustietokantaa nimeltä National Vulnerability Database (NVD) vuodesta 1999 asti. Tietokanta luotiin aluksi nimellä Internet Category of Attack toolkit, ja se uudelleennimettiin NVD:ksi vuonna 2005. NVD-tietokanta ja CVE-tietokannat ovat suoraan linkitettyjä toisiinsa; kaikki muutokset ja lisäykset CVE-tietokantaan näkyvät heti myös NVD-tietokannassa. NVD-tietokanta rakentuu siis CVE-tietokannan päälle ja antaa laajempaa tietoa listatuista haavoittuvuuksista, esimerkiksi niiden vakavuudesta, korjausmetodeista ja vaikutuksesta. (MITRE.org 2022)

CVE-tietokantaan kirjattujen haavoittuvuuksien määrä on ollut kasvussa vuodesta 2015 asti, kuten kuvasta 1 näkyy. Trendi vaikuttaa siltä, että haavoittuvuudet tulevat olemaan vakava uhka tulevaisuudessakin, ja tietoturvan tärkeys tulee kasvamaan entistäkin suuremmaksi.



Kuva 1. Uusien ohjelmistohaavoittuvuuksien lukumäärä vuosittain (Cvedetails.com 2022)

Ohjelmistohaavoittuvuuksia ja niiden aiheuttamia tietomurtoja on tutkittu paljon ja monista eri näkökulmista. Esimerkiksi mobiiliapplikaatioiden eri käyttäjätyyppikategorioiden mielipiteistä tietomurtojen henkilökohtaisista vaikutuksista on tehty tutkimusta (Stiakakis, E et al. 2016). Tietomurtojen taloudellisista vaikutuksista on tehty myös monia tutkimuksia. Goelin ja Shawkyn tutkimus (Goel, S. & Shawky, H. A. 2009) tarkasteli suurien amerikkalaisten yritysten markkina-arvon muutoksia, kun yrityksissä ilmoitettiin tietomurron tapahtuneen. Tämä tutkimuksen tuloksena havaittiin, että tilastollisesti merkittävä vaikutus tapahtuneen tietomurron ilmoituksesta oli 1% yrityksen tapahtumahetken markkina-arvosta. Samantyylinen tutkimustyö (Hinz, O. et al. 2015) on tehty vuonna 2014, jossa tutkittiin vain pienenElektroniikkaa myyvien kauppojen osakkeiden arvoa, ja tämänkin tulokset vahvistavat sen, että tietomurrot heikentävät osakearvoa. Myöhemmässä tutkimuksessa (Tayaksi, C. et al. 2022) on verrattu eri sektoreiden yritysten osakkeiden arvoa ennen ja jälkeen tietomurrosta tiedottamista, ja sen mukaan finanssiala ja teknologia-ala kärsivät eniten tietomurroista. Yllä mainitut tutkimukset ovat kuitenkin rajoittuneita vain muutamaan päivään sen jälkeen, kun tietomurrosta tiedotettiin. Toisen tutkimuksen (Juma'h, A. H. & Alnsour, Y. 2020) mukaan tietomurron tiedotuksella ei ole 3 kuukauden aikavälillä tilastollisesti merkittävää vaikutusta yrityksen osakkeen arvoon. Nämä tutkimukset osoittavat, että vaikka tietomurrot eivät välttämättä ole katastrofaalisia

yritysten osakkeen arvolle, ne kuitenkin vaikuttavat yrityksen toimintaan ja markkina-arvoon.

Myös ohjelmistohaavoittuvuuksia on tutkittu paljon. Monet tutkimukset erityisesti liittyvät uusiin metodeihin tunnistaa haavoittuvuuksia lähdekoodista. Eräs tutkimus (Subhan, F. et al. 2022) esitti mallin, joka käyttää koneoppimista haavoittuvuuksien tunnistamiseen, kun taas toinen tutkimus (Elder, S. Et al. 2022) vertaili eri metodeita tunnistaa haavoittuvuuksia ja niiden tehokkuutta. Tämän tyyppistä teknistä tutkimusta ja uusien tunnistamismallien esittämistä tehdään paljon, mutta ne eivät ota kantaa haavoittuvuuksien syntymisen estämiseen.

Asiaa on siis tutkittu paljon, ja monesta eri näkökulmasta. Tämän työn tavoitteena on saada yleiskäsitys siitä, minkälaisia ohjelmistohaavoittuvuuksia käsitellään tieteellisessä kirjallisuudessa, sekä selvittää onko tieteellinen keskustelu linjassa NVD:n keräämään haavoittuvuustilastodatan kanssa. Työssä esitetään seuraavat tutkimuskysymykset:

*Tutkimuskysymys 1: Minkä tyyppisiä haavoittuvuuksia on tarkasteltu vertaisarvioituissa tieteellisissä julkaisuissa viimeisen viiden vuoden aikana?*

*Tutkimuskysymys 2: Minkä tyyppiset haavoittuvuudet ovat olleet yleisimpiä viimeisen viiden vuoden aikana sellaisissa tapauksissa, joissa haavoittuvuus on johtanut ainakin yhteen tietomurtoon?*

Kandidaatintyön laajuuden vuoksi työ rajautuu vain ohjelmistojen tietoturvaan, ja siinä vielä tarkemmin ohjelmakoodiin liittyviin heikkouksiin. Esimerkiksi verkkoprotokollien haavoittuvuudet on rajattu työn ulkopuolelle, kuin myös ohjelmiston suunnitteluvaiheessa aiheutetut heikkoudet. Toinen tärkeä raja on se, että työ ottaa kantaa vain haavoittuvuustyyppihin, joiden tiedetään aiheuttaneen tietomurtoja oikeassa maailmassa, ja joista on saatavilla tietoa vapaista lähteistä. Luonnollisesti haavoittuvuuksia, joiden hyväksikäyttöä ei ole raportoitu ei voida ottaa tarkasteluun. Tämän työn tavoitteena on tunnistaa yleisimmin käytetyt haavoittuvuustyyppit ja verrata niitä tieteellisen tutkimuksen kattavuuteen.

## 2 Aiempi tutkimus

Tämä kappale käsittelee ohjelmistohaavoittuvuuksista tehtyä tutkimusta, joka on verrattavissa tämän työn tuloksiin. Koska tässä kappaleessa listatut tutkimukset eivät ole täysin verrannollisia tämän työn tuloksiin, käydään tässä myös läpi eroavaisuudet tämän työn ja tarkasteltujen tutkimuksien kanssa.

Haavoittuvuuksien historiaan liittyen on tehty tutkimus (Murtaza, S. S. et al. 2016), jossa on tarkasteltu muun muassa raportoitujen haavoittuvuuksien muutoksia tyypeittäin vuosien mittaan, samaan tyyliin kuin tässä työssä. Tutkimuksen tuloksena suurimmaksi kategoriaksi nousi ”buffer errors”. Kerätty tieto on kuitenkin vuodesta 2009 vuoteen 2014 asti, joten on mahdollista, että kategorioissa on tapahtunut muutoksia viimeisen 8 vuoden aikana. Murtazan tutkimuksessa otettiin myös huomioon kaikki haavoittuvuudet sen sijaan, että huomioitaisiin vain sellaiset haavoittuvuudet, joista tiedetään että niitä on käytetty hyväksi, joten tutkimus ei täysin vastaa tämän työn tavoitteita.

Myös NVD-katalogin ylläpitäjäjärjestö NIST tarkasteli NVD-katalogiin kirjattujen haavoittuvuuksien kategorioita vuosilta 2008-2016 (Kuhn, R. et. al. 2017). Tässä tutkimuksessa eri haavoittuvuudet kategorisoitiin niiden syntyyn johtavan suunnittelu- implementointi- tai asetustason virheiden perusteella. Tämä työ keskittyy vain implementointitason virheisiin, joten Kuhnin tutkimus on laajuudeltaan isompi kuin tämä työ. Kuitenkin koko tarkastelujakson aikana suurin osuus virheistä olivat juurikin implementointitason virheitä, vaikka muiden tasojen osuus olikin kasvanut ajan myötä. Tutkimus selvitti, että isoin implementointitason virhe on muistinkäsittelyn virheet, seuraavana Cross-site scripting, ja kolmanneksi suurin on implementoititason virhe oli SQL-injektiot. Tämä tutkimuksen tarkastelujakso päättyy juuri vuoteen 2016, mistä tämän työn tarkastelujakso alkaa. Toinen ero tähän työhön näkyy siinä, että tässä työssä käsitellään vain haavoittuvuuksia, jotka ovat johtaneet tietomurtoon.

Tammikuussa 2020 julkaistu kyberturvallisuuteen liittyvä systemaattinen kartoitustutkimus (Humayun, M. et. al. 2020) tutki, mitkä haavoittuvuus kategoriat ilmestyvät yleisimmin tutkimuksissa. Erona tähän työhön on tutkimuksen laajuus, joka kattaa myös tutkittuja haavoittuvuustyyppisiä, jotka eivät suoraan liity ohjelmointiin, esimerkiksi phishing-, denial-

of-service- ja man-in-the-middle-hyökkäykset, jotka rajattiin ulos tästä työstä. Tutkimuksen toteutustapa vastaa hyvin tämän työn toteutustapaa, mutta tämän työn rajausta on paljon spesifisempi.

Näiden erojen takia uskon, että on hyödyllistä koota kasaan viimeaikaista ohjelmistohaavoittuvuuksiin liittyvää tieteellistä kirjallisuutta, ja tutkia mistä eri näkökulmista haavoittuvuuksia on tutkittu, ja kuinka hyvin kirjallisuus vastaa tietomurtoihin johtaneiden haavoittuvuuksien jakaumaan. Haavoittuvuuksien nykytilanteen seuraaminen on varmasti mielenkiintoinen tieto sellaisenaan tutkijoille ja ohjelmoijille, jotka haluavat saada näkemyksen siitä, minkälaiset virheet ovat yleisimpiä ja minkälaisia virheitä käytetään hyväksi.

### 3 Tutkimusmenetelmä ja työn toteutus

Työssä tehtiin systemaattinen kartoitustutkimus sekä tilastodataan pohjautuva visualisointi, joiden tuloksia verrataan keskenään. Systemaattisessa kartoituksessa etsittiin vertaisarvioituja julkaisuja käyttäen LUT Primo-tietokantaa lähteenä. Kartoitustutkimuksen tuloksia verrattiin kategorisoituun NIST:n haavoittuvuusdataan. Käytetyt tutkimusmenetelmät kuvataan tarkemmin, jotta tutkimuksen voi toistaa myöhemmin uuden datan saamiseksi vertailua varten.

#### 3.1 Systemaattinen kartoitustutkimus

Kartoitustutkimus toteutettiin Petersenin, Feldtin, Mujtaban ja Mattsonin esittämän kartoitustutkimuksen mallin mukaan (Petersen et. al. 2008). Heidän mallinsa esittää viisi askelta, joiden avulla kartoitus toteutetaan.

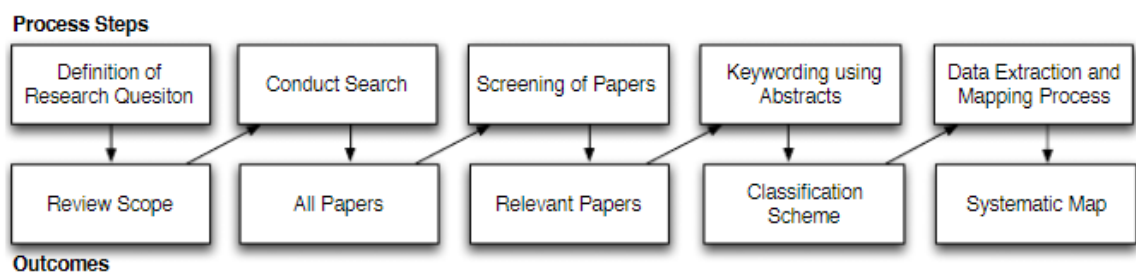
Ensimmäisessä askeleessa määritetään tutkimuskysymykset, joiden pohjalta tehdään etsintätyö halutuista lähteistä. Tämän lisäksi valitaan, mistä tietokannoista tutkimustietoa etsitään.

Seuraavassa vaiheessa luodaan yleistasoinen hakutermi, jonka tavoitteena on kattaa kaikki aiheeseen liittyvät tutkimukset, jotka ovat saatavilla digitaalisista tietokannoista. Samaa hakutermiä voi käyttää useammassa tietokannoissa, jos työssä on tarve käyttää useamman tietokannan lähteitä. Usein monissa eri tietokannoissa ei voi suoraan käyttää täysin samaa hakutermiä, sillä eri tietokannoilla on erilaiset määritellyt muodot hakutermeille. Näissä tapauksissa on hyväksyttävää tehdä pieniä muutoksia hakutermeihin, kuten muuttaa lainausmerkit suluiksi, tai muuten vaihtaa merkkejä niin, että hakutermi hyväksytään toisessa tietokannassa. Itse hakusanojen muuttamista tulisi välttää, jotta eri tietokantojen tulokset olisivat mahdollisimman verrannollisia. Hakutermi ja valitut tietokannat, joista tutkimuksia haetaan tulee myös kirjata ylös tässä vaiheessa.

Edellinen vaihe tuottaa suuren määrän tutkimusjulkaisuja, joista karsitaan epärelevantit tulokset pois. Tätä prosessia varten tulee kirjata ylös tutkimusten sisällyttämis- ja karsimiskriteerit. Kriteerejä voi luoda etukäteen, mutta on käytännöllistä luoda kriteerejä

samalla, kun tuloksia karsitaan. Tämä tapahtuu niin, että kun hakutuloksen tutkimusta tarkasteltaessa todetaan, että se ei ole relevantti tutkimuskysymystä varten, kirjataan ylös perustelu tämän tuloksen karsimiselle. Nämä perustelut kannattaa pitää mahdollisimman yleisinä, jotta niitä voi soveltaa myös jatkossa vastaan tuleviin tutkimuksiin.

Kun epärelevantit työt on karsittu, jäljelle jää vain aiheeseen liittyvät julkaisut. Näistä kootaan yhteen töiden avainsanat, jotka liittyvät tutkimuskysymykseen. Avainsanat luodaan lukemalla julkaisujen abstraktit, tai mahdollisesti johdanto- ja tulokset-osiot. Osiot lukemalla tutkija saa yleisen kuvan julkaisusta, ja voi sen perustella määrittää sille avainsanat, esimerkiksi aiheet, joita julkaisu käsittelee. Lopuksi avainsanoista muodostetaan visuaalinen esitys tuloksista, joiden avulla voidaan esittää mihin aiheisiin tutkimuksissa ollaan keskitytty enemmän, mihin vähemmän ja onko olemassa sellaisia aiheita, joita ei olla tutkittu paljon.



Kuva 2 Kartoitustutkimuksen prosessikaavio (Petersen et. al. 2008)

Työssä toteutettu kartoitus noudatti kuvassa 2 mainittuja vaiheita ja se toteutettiin seuraavalla tavalla: ensimmäisessä vaiheessa käytimme työn ensimmäistä tutkimuskysymystä: ”Minkä tyyppisiä haavoittuvuuksia on tarkasteltu vertaisarvioituissa tieteellisissä julkaisuissa viimeisen viiden vuoden aikana?” Työssä valittiin lähdetietokannaksi LUT Primo-järjestelmä. Primo hakee tuloksia monista tietokannoista, myös niistä, joissa on paljon tietoa ohjelmistokehitykseen liittyen kuten IEEE ja ACM (LUT Primo 2023). LUT Primossa on myös ominaisuus, jolla voi filteröidä pois lähteet, joita ei olla vertaisarvioitu. Näistä syistä työssä käytettiin vain LUT Primoa hakujen suorittamiseen ja lähteiden noutamiseen. Jatkotutkimuksessa tätä työtä voi laajentaa ottamalla mukaan muita tietokantoja, jotka ovat LUT Primon tietokantojen ulkopuolella.

Työssä käytetty hakutermin muodostui seuraavaksi: (security threat\*) AND vulnerabili\* AND software AND NOT (hardware OR IDS OR intrusion detection system). Hakutermin rajattiin käsittelemään vain englanninkielisiä julkaisuja, jotta prosessi pysyy

yksinkertaisempana. Vastaavanlaisia tutkimuksia voi jatkossa laajentaa sisällyttämällä myös muunkielisiä julkaisuja.

Hakutermissä sulkeet tarkoittavat termihakua. Esimerkiksi (security threat\*) tarkoittaa, että hakutuloksessa on oltava sanat security ja threat, tässä järjestyksessä. Tähtimerkki kertoo hakukoneelle, että sen tilalle voi sijoittaa mitä tahansa merkkejä. Tähtimerkit tämän työn hakutermissä kattavat monikkumuotoiset sanat. ”Security threat” hakusanana pyrkii rajoittamaan haun turvallisuuden pariin. Tämä ei yksinään riittänyt rajoittamaan hakua tarpeeksi, joten ”software” sisällytettiin hakutermiin, jotta tulokset liittyisivät juurikin ohjelmistoihin ja niiden turvallisuuteen. ”Vulnerability” lisättiin mukaan, jotta saamme vain teokset, joissa käsitellään jonkinlaisia haavoittuvuuksia. Yhdessä muiden hakusanojen kanssa saamme tulokset, jotka liittyvät turvallisuuteen, ohjelmistoihin ja jotka käsittelevät haavoittuvuuksia.

Ensimmäisten tulosten joukosta löytyi useita yksittäisiin, fyysisiin laitteisiin keskittyviä tuloksia. Tässä työssä haluttiin keskittyä yleisesti ohjelmistojen, ei fyysisten laitteiden haavoittuvuuksiin, joten sana ”hardware” rajattiin pois hakutermeistä. Hakutuloksissa oli useita teoksia liittyen luvattoman käyttäjän tunnistamiseen. Myös ”intrusion detection system” rajattiin pois, sillä työssä haluttiin keskittyä haavoittuvuuksien synnyn torjumiseen, ei tunkeilijoiden tunnistamiseen. Tämän lisäksi haulle määriteltiin kaksi rajoitusta. Hakuun otettiin mukaan vain vertaisarvioidut teokset, jotka on julkaistu vuoden 2018-2022 välillä. Aikarajaus tehtiin, jotta tiedot olisivat suhteellisen tuoreita, ja vertaisarvioinnilla haluttiin pitää lähteet mahdollisimman luotettavina.

Seuraava askel prosessissa oli tulosten seulominen. Hakutermi tuotti hakuhetkellä 349 tulosta. Työssä haluttiin keskittyä julkaisuihin, joissa tarkastellaan ohjelmistojen haavoittuvuuksia ja niiden torjumista. Seulominen tehtiin lukemalla hakutulosten otsikot ja abstraktit. Seulonnan aikana listattiin kriteerejä, joiden perusteella hakutuloksia rajattiin pois sitä mukaan kun epärelevantteja tuloksia saatiin. Alla listattuna rajaukset, joiden perusteella teoksia seulottiin pois:

- Teokset jotka tutkivat haavoittuvuuksien havaitsemista
- Teokset, jotka eivät liity ohjelmointiin
- Teokset, jotka tutkivat verkkojen ja protokollien haavoittuvuuksia

- Liiketaloutta tutkivat teokset
- Tietoturvatietoja tutkivat teokset
- Sosiaalisen manipuloinnin tuottamat haavoittuvuudet
- Fyysiseen turvallisuuteen liittyvät teokset
- Teokset, joissa vain mainitaan haavoittuvuustyyppi nimeltä

Rajausten jälkeen teosten lukumäärä laski 53 teokseen. Tässä vaiheessa teoksia tutkittiin tarkemmin. Jokaisesta teoksesta etsittiin kappaleet, jotka liittyvät haavoittuvuuksiin, ja ne luettiin. Monissa teoksissa oli paljon muuta materiaalia, joka joko selitti tarkemmasta teoksen aiheesta, kuten Androidista tai Internet of Thingsistä, ja haavoittuvuuksille oli oma osionsa. Tässä vaiheessa käytettiin samoja rajauskriteereitä kuin aikaisemmassa vaiheessa, ja siten saatiin otoksesta pois viimeiset epärelevantit teokset, jotka aluksi vaikuttivat lupaavilta abstraktin ja otsikon perusteella, mutta joiden sisältö ei käsitellyt haavoittuvuuksia tarpeeksi paljon. Viimeisen seulontakierroksen jälkeen teosten määrä oli 20 kappaletta.

Kun seulonta oli valmis, teoksien sisällöstä kirjattiin ylös haavoittuvuudet, joita teoksissa käsiteltiin, jonka jälkeen ne kategorisoitiin eri haavoittuvuustyyppihin. Kukin teos voitiin lisätä kerran kuhunkin kategoriaan, jos niissä käsiteltiin useampaa haavoittuvuutta. Kategorisointi täytyi tehdä, jotta se olisi vertailukelpoista seuraavassa vaiheessa hankittavan haavoittuvuusdatan kanssa. Lopulliset kategoriat olivat seuraavat:

- Muistinhallintaan liittyvät virheet (CWE-119)
- Injektiot (CWE-943)
- Ei-tuettujen kolmannen osapuolen komponenttien käyttö (CWE-1357)
- Kryptografiset virheet (CWE-310)
- Toiminnonkulun virheet (CWE-691)
- Huono syötön validointi (CWE-20)
- Kokonaisluvun ylivuoto (CWE-190)

Kategoriat pohjautuvat MITRE:n CWE-heikkoustyypin listaukseen (Common Weakness Enumeration 2022), jotka parhaiten täsmäävät teoksessa esiteltyyn haavoittuvuuteen. Jotta

data ei pirstaloituisi liikaa, osa haavoittuvuuksista luokiteltiin yhteen. Esimerkiksi sen sijaan, että eroteltaisiin SQL injektiot, Komento-injektiot, XPath-injektiot yms, nämä kaikki yhdistettiin vain sateenkaritermille ”injektiot”. Loppujen lopuksi datapisteiden määrä nousi 32:een, sillä monessa teoksessa käsiteltiin useampaa eri haavoittuvuustyyppiä. Lopullinen lista valituista aineistoista ja niissä käsitellyistä haavoittuvuuksista näkyy taulukossa 1. Taulukon otsikot ovat lyhenteitä, ja vastaavat seuraavia kategorioita:

- MEM, Muistinkäsittelyn virheet
- INJ, Injektiot
- 3RD, Ei-tuetun kolmannen osapuolen komponenttien käyttö
- CRY, Kryptografiset virheet
- CF, Toiminnonkulun virheet
- INP, Syötön validointivirheet
- INT, Kokonaisluvun ylivuoto

Teoksen nimi	MEM	INJ	3RD	CRY	CF	INP	INT
An Analysis of Smart Contracts Security Threats Alongside Existing Solutions (López Vivar et al. 2020)					X		
Perspectives and Reviews in the Development and Evolution of the Zero-Day Attacks (Teodorescu 2022)	X		X				
A Large Scale Analysis of Android — Web Hybridization (Tiwari et al. 2020)				X			
Serverless computing: a security perspective (Marin et al. 2022)		X	X		X		
On the Design of IoT Security: Analysis of Software Vulnerabilities for Smart Grids (Mathas et al. 2021)	X			X	X		
Software vulnerabilities in TensorFlow-based deep learning applications (Filus & Dománska 2023)	X					X	X
Collecting Vulnerable Source Code from Open-Source Repositories for Dataset Generation (Raducu et al. 2020)	X						
Control-Flow Integrity: Attacks and Protections (Sayeed et al. 2019)	X						X
An Experimental Analysis of Security Vulnerabilities in Industrial IoT Devices (Jiang et al. 2020)			X	X			
Internet of Things: Security and Solutions Survey (Sadhu et al. 2022)		X					
Security vulnerabilities related to web-based data (Awad et al. 2019)	X	X					
Implementation of Cyber Security Attacks and Strategic Mitigation Mechanisms (Kela et al. 2022)		X					
Evading Security Products for Credential Dumping Through Exploiting Vulnerable Driver in Windows Operating Systems (Dang et al. 2021)	X						
SQL Injection Prevention in Web Application: A Review (Abdullah et al. 2021)		X				X	
Call Me Back, I Have a Type Invariant (Sekerinski et al. 2020)					X		
SoK: Secure Memory Allocation (Conti et al. 2021)	X						
A Review of Memory Errors Exploitation in x86-64 (Pirry et al. 2020)	X						
Mitigating Data-only Attacks by Protecting Memory-resident Sensitive Data (Palit et al. 2020)	X						
CIMA: Compiler-Enforced Resilience Against Memory Safety Attacks in Cyber-Physical Systems (Chekole et al. 2020)	X						
Robotics cyber security: vulnerabilities, attacks, countermeasures, and recommendations (Yaacoub et al. 2022)	X	X					

Taulukko 1 Kirjallisuuskatsauksen lopulliset teokset ja niissä käsitellyt haavoittuvuus kategoriat.

### 3.1.1 Valittujen aineistoiden kuvaus

Tässä kappaleessa perustellaan tarkemmin, miksi kukin työ on kirjattu minkäkin kategorian alle. Teoksista annetaan myös lyhyt kuvaus, joka kertoo enemmän siitä, mihin asioihin teoksissa ollaan keskitytty, ja miten teos liittyy ohjelmistohaavoittuvuuksiin.

Antonio López Vivar, Alberto Turégano Castedo, Ana Lucila Sandoval Orozco ja Luis Javier García Villalba tarkastelivat älysovimuksien haavoittuvuuksia. Tutkimuksessa esitetään case-esimerkki 2016 tapahtuneesta tietomurrosta, jossa 3.6 miljoonaa Ether-kryptovaluuttaa arvoltaan noin 70 miljoonaa dollaria. Tapauksessa hyökkääjä pystyi kutsumaan rekursiivisesti älysovimuksen rahan nostometodia, joka älysovimuksen virheestä johtuen ei päivittänyt nostettavissa olevan rahan määrää, jolloin hyökkääjä pystyi ohittamaan tarkistuksen, joka normaalisti estäisi nostamisen. Tämä kirjattiin toiminnonkulun virheeksi. (López Vivar et al. 2020)

Cosmin Alexandru Teodorescu tarkasteli tunnettuja nollapäivähaavoittuvuuksia. Työssä esitellään SolarWinds-hyökkäys, jossa haitallista koodia ajettiin muun muassa kyberturvallisuusyritysten ohjelmistoissa. Koodi päätyi ohjelmaan Orion-nimisen ohjelmiston päivitysten mukana. Tämä kirjattiin ei-tuetun kolmannen osapuolen koodin käytöksi. Teoksessa käsiteltiin myös muistinkäsittelyn virheitä Yhdysvaltojen armeijan verkkopalvelimilla. Tapauksessa hyökkääjät olivat päässeet käsiksi armeijan palvelimelle. Kyseessä oli puskurin ylivuoto, joka olisi voinut antaa hyökkääjälle täyden pääsyn järjestelmään, jossa hän voisi katsoa ja muokata tiedostoja sekä asentaa mitä tahansa muuta koodia palvelimille. (Teodorescu 2022).

Abishek Tiwari, Jyoti Prakash, Sascha Groß ja Christian Hammer tutkivat Android-applikaatioiden verkon käyttöä. Tutkimus on laaja, mutta siinä tehtiin mielenkiintoinen huomio applikaatioiden käyttämistä web-url:sta. Noin 11% applikaatioiden käyttämistä loadUrl()-kutsuista käyttivät salaamatonta http-protokollaa. Tämä kirjattiin kryptografisiin virheisiin. (Tiwari et al. 2020)

Eduard Marin, Diego Perino ja Roberto Di Pietro tutkivat minkälaisia haavoittuvuuksia palveliton arkkitehtuuri voi aiheuttaa. Teoksessa mainittiin tämän arkkitehtuurin oma versio injektiohyökkäyksistä, joka kirjattiin injektioiden alle. Myös typosquatting, jossa pyritään asentamaan haitallista koodia julkaisemalla haitallinen kopio kirjastosta hieman väärin

kirjoitetulla nimellä mainittiin uhkatekijänä. Tämä kirjattiin ei-tuetun kolmannen osapuolen koodin käyttönä. Teoksessa tunnistettiin myös kolmas haavoittuvuusluokka, joka kirjattiin toiminnonkulun viheeksi: kilpajuoksuolosuhteet. Jos jokin komponentti päivitetään käyttäytymään eri tavalla, pilvipalvelun tarjoajat saattavat käyttää vanhaa versiota, joka voi johtaa toiminnonkulun ongelmiin. (Marin et al. 2022)

Christos-Minas Mathas ja muut tutkivat ohjelmistohaavoittuvuuksia älyverkkoihin tarkoitetuissa ohjelmistoissa. He tunnistivat tarkastelluissa ohjelmissa kryptografisia virheitä, liian heikko sertifikaattien tarkistus, muistinkäsittelyn virheitä kuten puskurin ylivuoto sekä toiminnonkulun virheitä kuten kilpajuoksuolosuhteet. Tutkimuksessa kuvataan haavoittuvuuksia tarkemmin niiden ohjelmistojen valossa, joissa tutkijat haavoittuvuudet löysivät. (Mathas et al. 2021)

Katarzyna Filus ja Joanna Domanska tarkastelivat TensorFlow:n perustuvien ohjelmistojen haavoittuvuustyyppisiä. He tunnistivat eri haavoittuvuustyyppisiä ja kirjasiivat minkälaisilla metodeilla niitä voidaan käyttää hyväksi. Tuloksissa tunnistettiin useita muistinkäsittelyn virheitä, kuten ohjelman muistialueen ulkopuolelle kirjoittaminen, puskurin ylivuoto, muistin vapauttamisen virheitä. Nämä kirjattiin yhdeksi muistinkäsittelyn virheeksi. Myös kokonaisluvun ylivuoto ja virheellinen syötön käsittely tunnistettiin mahdolliseksi haavoittuvuudeksi, ja nekin kirjattiin ylös. (Filus & Domanska 2023)

Razvan Raducu, Gonzalo Esteban, Francisco J. Rodríguez Lera ja Camino Fernández keräsivät avoimen lähdekoodin projekteista esimerkkidataa, jota voi käyttää esimerkiksi koneoppimismallien kouluttamiseen. Tutkimus keskittyi erityisesti C-kielen muistinkäsittelyn virheisiin, ja tuotti mielenkiintoisen huomion, jossa suurin osa haavoittuvuuksista liittyy funktioihin `sprintf()`, `strcpy()`, `strcat()` ja `strlen()`. Tämä tarkastelu kirjattiin muistinkäsittelyn virheisiin. (Raducu et al. 2020)

Sarwar Sayeed, Hector Marco-Gisbert, Ismael Ripoll ja Miriam Birch tutkivat matalan tason ohjelmointikieliin liittyviä hyökkäyksiä ja puolustusmetodeja. He kuvasivat tutkimuksessaan muistinkäsittelyn virheitä, kuten puskurin ylivuotoa. Myös kokonaisluvun ylivuoto oli esitelty tutkimuksessa. Molemmista tyypeistä oli myös esitetty tapoja käyttää haavoittuvuutta hyväksi. (Sayeed et al. 2019)

Xingbin Jian, Michelle Lora ja Sudipta Chattopadhyay tutkivat haavoittuvuuksia esineiden internetissä. He esittelevät erilaisia haavoittuvuuksia, joista osa liittyvät laitteen ohjelmistoon. Heikkojen salasanojen käyttö kirjattiin kryptografiseksi virheeksi, haitallisen koodin lataaminen ei-turvattujen päivitysten mukana kirjattiin ei-tuetun kolmannen osapuolen koodin käytöksi. Työssä on esiteltyä myös muita, laitteisiin ja fyysiseen turvallisuuteen liittyviä haavoittuvuuksia, mutta niitä ei otettu huomioon tässä työssä. (Jiang et al. 2020)

Pintu Kumar Sadhu, Venkata P. Yanambaka ja Ahmed Abdelgawad tutkivat erilaisia hyökkäyksiä, joita voidaan tehdä esineiden internetin kautta. Tutkimuksessa löydettiin paljon erilaisia hyökkäyksiä eri tasoilla, kuten fyysisiä hyökkäyksiä ja verkkohyökkäyksiä, mutta myös ohjelmistoon liittyviä hyökkäyksiä. Jotkut ohjelmistoihin liittyvistä hyökkäyksistä olivat sosiaalista manipulointia, kuten phishing. Työstä kirjattiin injektiot, sillä niitä kuvailtiin haitallisina skripteinä, jotka asentuvat käyttäjän vieraillessa esimerkiksi haavoittuneella verkkosivulla. (Sadhu et al. 2022)

Mohammed Awad, Muhammed Ali, Maen Takruri ja Shereen Ismail tarkastelivat yleisellä tasolla internetiin liittyviä haavoittuvuuksia ja niiden ennaltaehkäisyä. Työssä kuvaillaan haavoittuvuustyyppit ja annetaan niistä esimerkkejä. Työ ottaa suoraan kantaa SQL-injektioihin ja puskurin ylivuotoon, joten teos merkittiin injektioiden sekä muistinkäsittelyn virheiden alle. (Awad et al. 2019)

Rushabh Kela, Abhinav Chawla, Pratishta Gaur ja Manikandan K tekivät käytännön kokeiluja erilaisista hyökkäysmetodeista rakentamaansa haavoittuvaista verkkopalvelua vastaan, joista yksi tähän työhön sopivista oli NoSQL injektio, joka kirjattiin injektioiden kategorian alle. Teoksessa kuvattiin hyökkäys, sekä esitettiin tapoja estää vastaavat hyökkäykset. Myös muita hyökkäystyyppisiä käsiteltiin, esimerkiksi salasanan arvaamista brute-force-metodilla, mutta ne rajattiin tämän työn ulkopuolelle. (Kela et al. 2022)

Huu-Danh Pham, Vu Thanh Nguyen, Mai Viet Tiep, Vu Thanh Hien, Phu Phuoc Huy ja Pham Thi Vuong tutkivat vanhojen laiteajureiden haavoittuvuuksia Windows-käyttöjärjestelmässä. Teoksessa käytettiin hyväksi haavoittuvuutta ajurissa, joka antoi tutkijoille pääsyn fyysiseen muistiin, jonka kautta he kirjoittivat haitallista koodia erääseen windowsin funktioon, jolla he pystyivät näkemään tietokoneen käyttäjätunnukset. (Dang et al. 2021)

Joanna Hazaline Binti Johny, Wafa Athilah Fikriah Binti Nordin, Nurrina Mizana Binti Lahapi ja Yu-Beng Leau tekivät kattavan selvityksen SQL-injektioista. Tutkimuksessa kuvailtiin tarkoin erilaisia injektiotyyppejä sekä syitä näiden haavoittuvuuksien syntymiselle, kuin myös niiden torjumista. Teos keskittyi injektioihin, mutta käsitteli myös syötön validoinnin tärkeyttä injektioiden estämiseksi, joten teokselle kirjattiin sekä injektiot että syötön validointivirheet. (Abdullah et al. 2021)

M. Anthony Aiello, Johannes Kanig ja Taro Kurita tutkivat älysovimusten haavoittuvuuksia. Työssä tarkasteltiin älysovimuksien re-entrancy-haavoittuvuuksia, samaan tyyliin kuin aikaisempi López Vivar ja muiden tutkimus, joten kyseessä oli toiminnonkulun virhe. (Sekerinski et al. 2020)

Bojan Novković ja Marin Golub tutkivat kekomuistin korruptiota hyväksikäyttäviä haavoittuvuuksia ja esittivät turvallisia tapoja allokoida muistia. Koko tutkimus perustui muistinkäsittelyn virheisiin, joten teos kirjattiin samannimiseen kategoriaan. (Conti et al. 2021)

Conor Pirry, Hector Marco-Gisbert ja Carolyn Begg tutkivat muistinkäsittelyn virheitä x86-64 arkkitehtuurin tietokoneissa. Heidän työssään selitetään muistipinon toimintaa ja annetaan esimerkkejä erilaisista muistinkäsittelyn virheistä, kuten puskurin ylivuodosta. Työssä myös esitellään tapoja, joilla muistinkäsittelyn virheitä voidaan välttää. (Pirry et al. 2020)

Tapti Palit, Fabian Monroe ja Michalis Polychronakis esittivät tavan suojautua datan vuotamiselta suojaamalla muistissa olevaa arkaluontoista dataa enkryptiolla. Tutkimus käsittelee siis muistinkäsittelyn virheitä, ja esittää uuden tavan suojata data muistissa. Teos kirjattiin siis muistinkäsittelyn virheiden alle. (Palit et al. 2020)

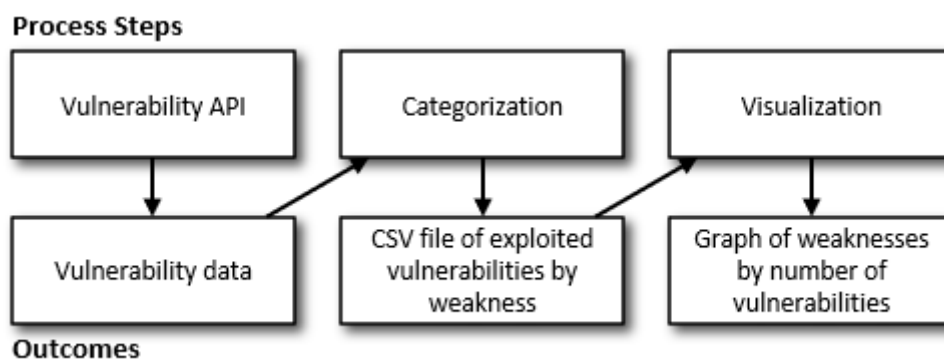
Eyasu Getahun Chekole, Sudipta Chattopadhyay, Martín Ochoa, Huaqun Guo ja Unnikrishnan Cheramangalath esittivät tekniikan, jolla he pystyvät vähentämään muistia hyväksikäyttävien hyökkäysten tehokkuutta. Työssä esitellään erilaisia muistinkäsittelyn virheitä hyödyntäviä hyökkäyksiä, ja esitellään metodeita, joilla niitä voidaan ennaltaehkäistä. Teos kirjattiin muistinkäsittelyn virheiden alle. (Chekole et al. 2020)

Jean-Paul Yaacoub, Hassan Noura, Ola Salman ja Ali Chehab tarkastelivat robotiikkaan liittyviä ohjelmistohaavoittuvuuksia. Teoksessa esitellään erilaisia yhteiskunnalle tärkeitä robotiikan sovellustapoja, sekä turvallisuusuhkia, joita robotiikan käyttö voi aiheuttaa.

Työssä mainitaan fyysisiä ja sosiaalisia uhkia, mutta myös ohjelmistohaavoittuvuuksien aiheuttamia uhkia. Näissä mainitaan esimerkiksi injektiot, jotka voivat johtaa robotin paljastamaan salattua tietoa sekä puskurin ylivuodon hyväksikäyttäminen, jossa robotin voisi kaapata kokonaisuudessaan. Näiden valossa työ kirjattiin muistinkäsittelyn sekä injektioiden alle. (Yaacoub et al. 2022)

### 3.2 Tilastodataan pohjautuva kategorisointi

Tilastodatan visualisoinnissa käytettiin vapaata menetelmää sen sijaan, että seurattaisiin tarkoin määriteltyä tutkimusmenetelmää. Menetelmä on kuvattu kuvassa 3. Työ aloitettiin etsimällä sopiva tietoaaineisto. Haavoittuvuusdatasta haluttiin saada tietoa siitä, minkälaiset haavoittuvuudet ja ohjelmistojen heikkoudet ovat aiheuttaneet tietomurtoja. Tästä syystä prosessin ensimmäiseen vaiheeseen valittiin juuri NVD-tietokanta, jossa on listattuna kaikki NIST:lle raportoidut haavoittuvuudet, sekä tieto siitä, onko haavoittuvuutta hyväksikäytetty.



Kuva 3 Tilastodataan pohjautuvan visualisoinnin prosessikaavio

Haavoittuvuuksien raportoiduista hyväksikäytöistä pitää kirjata Cybersecurity & Infrastructure Security Agency (CISA). He pitävät yllä julkista tietokantaa tunnetuista hyväksikäytetyistä haavoittuvuuksista nimeltä ”Known Exploited Vulnerabilities Catalog” (KEV). Tämä katalogin listaamat haavoittuvuudet ovat samat kuin NIST:n CVE tietokannassa, ja NVD-tietokannan haavoittuvuuksiin on lisätty linkki KEV-katalogiin, jos kyseinen haavoittuvuus löytyy KEV-katalogista. NVD-tietokannan API:ssa onkin parametri, jolla voi filteröidä pois kaikki sellaiset haavoittuvuudet, joita ei ole listattu hyväksikäytetyiksi KEV-katalogissa. Ominaisuuden saa käyttöön lisäämällä API-kutsuun ”hasKev”-nimisen argumentin. Tässä työssä käytettiin tätä ominaisuutta vain KEV-katalogissa hyväksikäytetyiksi merkattujen haavoittuvuuksien etsimiseen.

Kun tietoaaineisto oli valittu, data täytyi ottaa ulos järjestelmästä sen käsittelyä varten. Tämä toteutettiin Python-ohjelmalla, joka yhdistää NIST:n tietokantaan ja hakee sieltä KEV-katalogista löydettyihin haavoittuvuuksiin filteröidyn datan. Ohjelma sen jälkeen karsii pois tulokset, jotka on julkistettu ennen vuotta 2018, jotta haavoittuvuuksien data pysyy

vertailukelpoisena kartoitustutkimuksen kanssa. Ohjelman lähdekoodi on saatavilla liittestä 1.

Kun data on haettu tietokannasta, se kategorisoidaan samoihin kategorioihin, mitkä tunnistettiin tutkimusten kartoitusvaiheessa. Haavoittuvuuksien kategorisointi perustui kullekin haavoittuvuudelle määritellyn ohjelmistoheikkouden perusteella. NIST:n haavoittuvuusdatassa kaikille haavoittuvuuksille on annettu ns. Common Weakness Enumeration (CWE) -kategoria. Tämän tarkoituksena on kertoa, minkälainen heikkous ohjelmistossa on aiheuttanut haavoittuvuuden. Erilaisia CWE:ssä määriteltyjä heikkouksia on tämän tekstin kirjoituksen aikana 933, joka on liian suuri määrä tämän työn tavoitteisiin nähden. CWE-heikkoudet ovat kuitenkin myös kategorisoitu ylemmillä tasoilla, joten työssä käytettiin laajempia kategorioita avuksi. (Common Weakness Enumeration 2022)

Kategoriat määrittyivät seuraavasti:

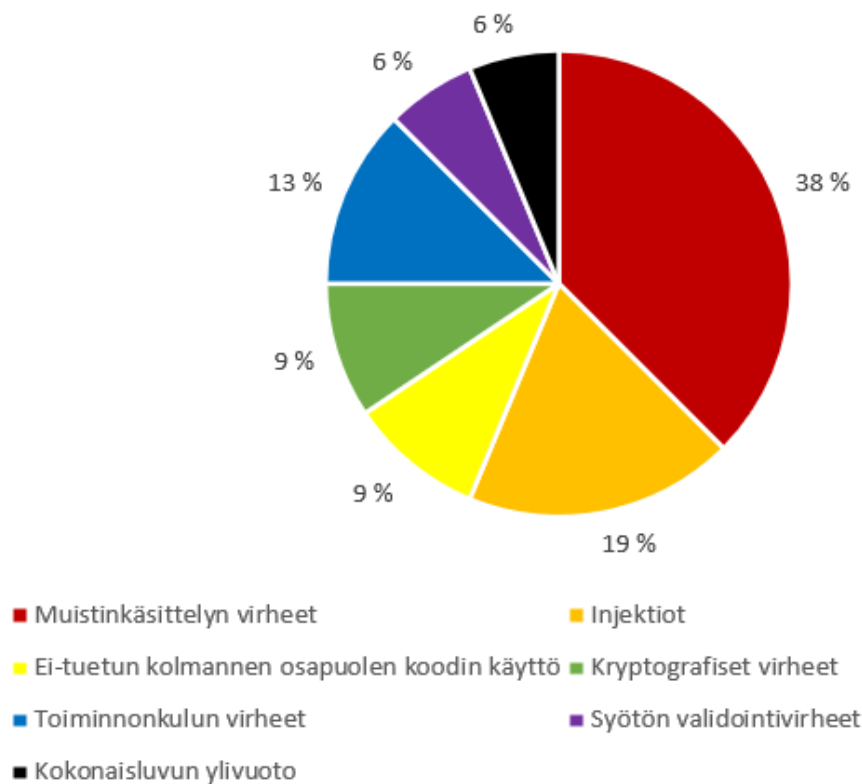
- Muistinhallintaan liittyvät virheet (CWE-119, CWE-787, CWE-416, CWE-125, CWE-415, CWE-131 & CWE-120)
- Injektiot (CWE-97, CWE-77, CWE-89, CWE-74, CWE-917, CWE-91 & CWE-78)
- Ei-tuettujen kolmannen osapuolen komponenttien käyttö (CWE-1357)
- Kryptografiset virheet (CWE-310, CWE-326 & CWE-347)
- Toiminnonkulun virheet (CWE-362)
- Huono syötön validointi (CWE-20)
- Kokonaisluvun ylivuoto (CWE-190)

Eri CWE-kategorioiden määrä on nähtävissä kuvasta 7.

Lopuksi kategorisoitu data visualisoitiin Excelillä, ja sitä verrattiin vastaavaan visualisaatioon kartoitustutkimuksen datasta. Haavoittuvuusdatasta otettiin visualisointiin mukaan vain sellaiset haavoittuvuustyypit, jotka tunnistettiin kartoitustyössä.

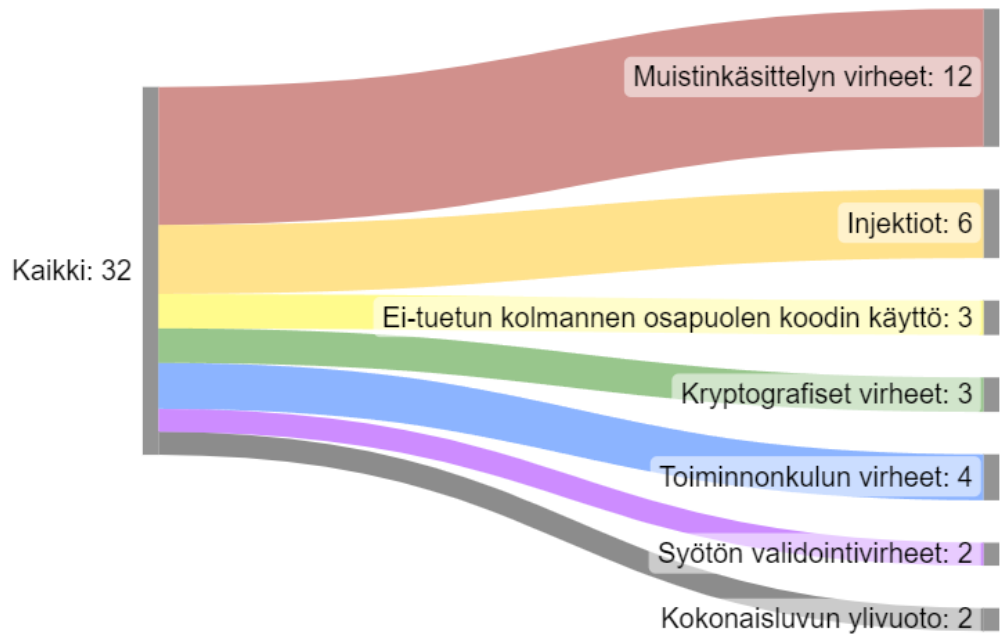
## 4 Tulokset ja huomiot

Kun kirjallisuuskartoituksen tulokset ja haavoittuvuusdata oli kerätty, pystyimme vertaamaan niitä keskenään. Vertailu paljastaa, että viimeisen viiden vuoden aikana tehdyissä haavoittuvuuksiin liittyvissä tutkimuksissa ollaan käsitelty samantyyppisiä haavoittuvuuksia melko lailla samassa suhteessa.



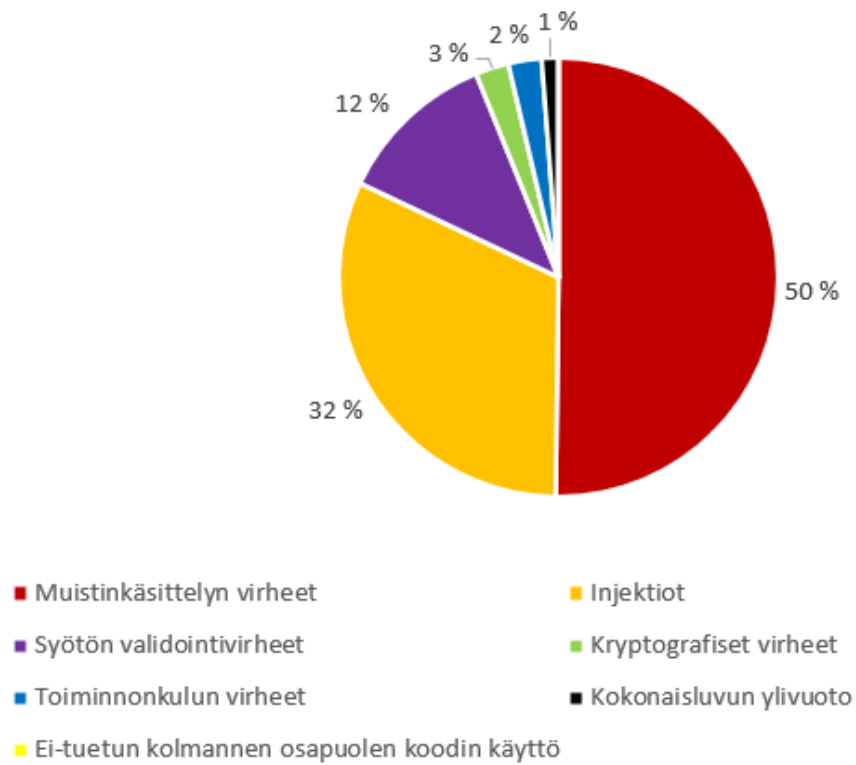
Kuva 4 Tutkimuksista havaitut ohjelmistohaavoittuvuus kategoriat vuosilta 2018-2022

Kuvassa 4 näkyy kirjallisuuskatsauksesta havaitut haavoittuvuus kategorioiden osuudet. Kokonaisuudessaan datapisteitä oli 32. Suurin kategoria oli muistinkäsittelyn virheet 12 teoksella, toiseksi suurin injektiot 6 teoksella. Pienimmissä kategorioissa, syötön validoinnissa ja kokonaisluvun ylivuodossa oli kummassakin vain 2 teosta. Kaikkien datapisteiden jakaumat näkyvät kuvasta 5.



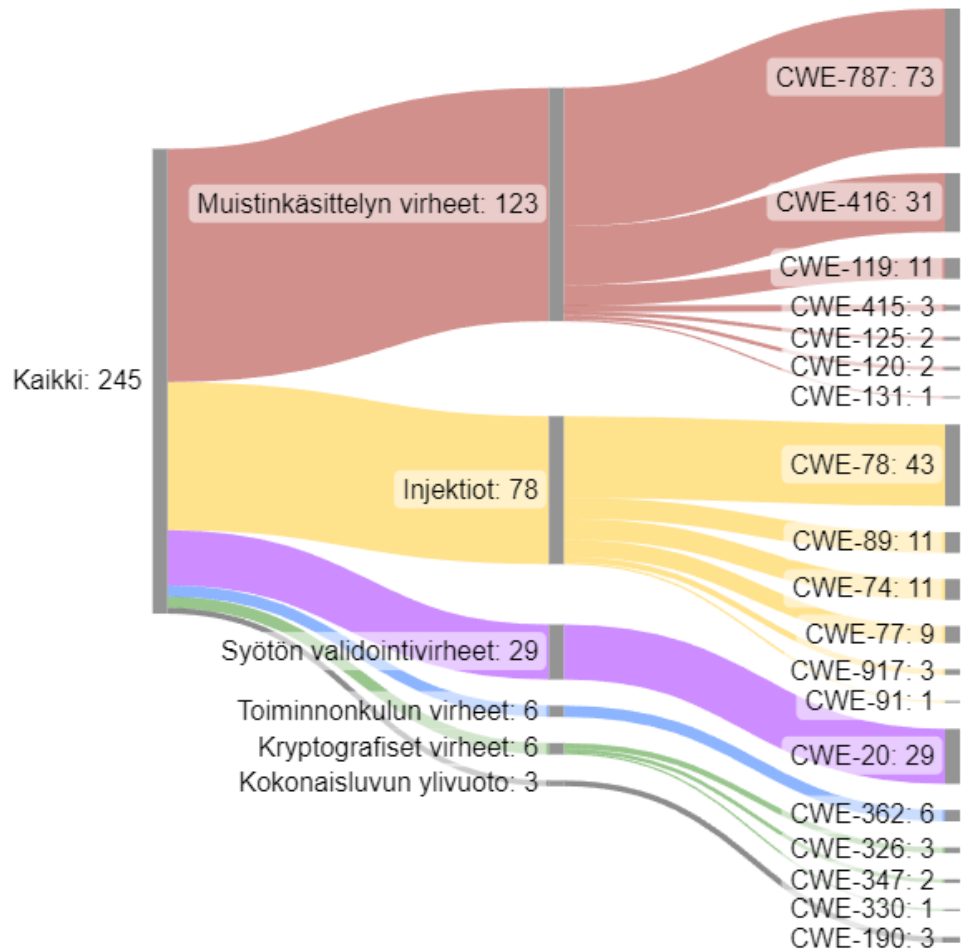
Kuva 5 Sankey-kaavio Tutkiumuksista havaituista ohjelmistohaavoittuvuuskategorioiden vuosilta 2018-2022.

Haavoittuvuusdatan kategoriat näkyvät kuvasta 6. Tarkasteltuja haavoittuvuuksia oli yhteensä 245 kappaletta, joiden suurin kategoria oli muistinkäsittelyn virheet 123 haavoittuvuudella. Toiseksi suurin kategoria oli injektiot 78 haavoittuvuudella. Pienimmät kategoriat sisälsivät vain muutamia haavoittuvuuksia; kokonaisluvun ylivuotoja oli raportoitu 3 kappaletta, toiminnonkulun virheitä 6 kappaletta ja kryptografisia virheitä 6 kappaletta. Ei-tuetun kolmannen osapuolen koodin käyttölle ei oltu kirjattu yhtään tietoturtoon johtanutta haavoittuvuutta.



Kuva 6 Hyväksikäytettyjen haavoittuvuuksien data kategorioittain vuosilta 2018-2022

Haavoittuvuusdatan tarkempi jaottelu sekä datapisteiden lopullinen määrä kategorioittain sekä jaottelu siitä, mistä CWE-heikkouksista kategoriat rakentuvat on nähtävissä kuvasta 7. Jos esimerkiksi injektioita ei olisi yhdistetty samaan kategoriaan, data olisi melko pirstailoutunutta, ja vertailu kirjallisuuteen olisi vaikeampaa.



Kuva 7 Sankey-kaavio hyväksikäytettyjen haavoittuvuuksien kategorioista ja niiden jaottelusta CWE-kategorioihin vuosilta 2018-2022.

Aineistoja verratessa muistinhallintaan liittyvät virheet olivat molemmissa suurin yksittäinen kategoria. 50% tietomurtoja aiheuttaneista haavoittuvuuksista johtuivat virheistä muistinhallinnassa, ja 38% tähän työhön valituista tutkimuksista käsitteli muistinhallintaan liittyviä haavoittuvuuksia. Myös toiseksi suurin kategoria, injektiot, oli edustettuna haavoittuvuuksissa 32% osuudella, ja vastaavan kategorian osuus tutkimusaineistosta oli 19%.

Huomattavia eroavaisuuksia tutkimusten ja haavoittuvuuksien välillä kolmannen osapuolen ei-tuettujen komponenttien käytössä, toiminnonkulun virheissä ja syötön validoinnissa. Toiminnonkulun virheitä tutkitaan suhteessa enemmän kuin niitä tapahtuu oikeassa maailmassa. Nämä tutkimukset usein liittyivätkin johonkin uuteen teknologiaan, kuten lohkoketjuun. Vakiintuneissa teknologioissa tällaiset virheet tyypillisesti on jo korjattu silloin, kun ne olivat vielä uusia. Syötön validointi edustaa 12% tietomurtoon johtaneista virheistä, mutta sen osuus tutkimustiedosta on paljon pienempi. Tämä voi selittyä sillä, että

syötön validointi on melko hyvin tunnettu haavoittuvuus, jolle on tehty oma artikkeli OWASP:n Cheat sheet-artikkelisarjassa, jonka tarkoituksena on tuottaa yksinkertaisia ohjeita yleisten haavoittuvuuksien torjuntaan. (OWASP.org, 2023)

Vaikka ei-tuettujen komponenttien käytölle on määritelty oma heikkoustyyppi, sitä ei NVD:n tietokannassa käytetä haavoittuvuuksien ensisijaiseksi taustatekijäksi (NIST.gov, 2022). Ohjelmiston haavoittuvuus voi toki johtua kolmannen osapuolen kirjastossa olevasta haavoittuvuudesta, mutta tällaista tilannetta ei julkisteta erillisenä haavoittuvuutena, sillä se perustuu täysin kirjaston haavoittuvuuteen, jolla pitäisi olla oma haavoittuvuusjulkaisu. Tämä voi olla syynä sille, miksi kolmannen osapuolen komponenttien käytölle ei ole merkattu yhtään haavoittuvuusjulkaisua viimeisen viiden vuoden aikana, vaikka tällaisia haavoittuvuuksia on varmasti ilmennyt kyseisenä aikana. Työn tulokset viittaavat siihen, että ainakin suurimpia haavoittuvuustyppejä käsitellään tutkimuksissa melko lailla samassa suhteessa kun niitä esiintyy sellaisissa haavoittuvuuksissa, jotka ovat johtaneet tietomurtoihin viimeisen viiden vuoden aikana.

Tutkimuksen tulokset vastasivat hyvin sille asetettuihin tavoitteisiin. Tuloksena selvitettiin, että viimeisen viiden vuoden aikana julkaistuissa tieteellisissä teoksissa pitkälti käsitellään erityyppisiä haavoittuvuuksia samassa suhteessa kun niitä on esiintynyt oikeassa maailmassa. Suurimmat erot ovat kolmannen osapuolen komponenttien käytön ylitarkastuksessa tutkimuksissa ja syötön validoinnin alitarkastus tutkimuksissa. Suurimmat kategoriat tutkimuksessa ja haavoittuvuusraporteissa olivat samat: Muistinkäsittelyn virheet ja injektiot.

Tärkein huomio tulosten analysoimisessa on kiinnittää huomiota otoskokojen eroihin. Kirjallisuuskartoituksessa on vain 32 datapistettä, kun taas haavoittuvuusdatassa on 245. Kategorioiden määrän ollessa sama, kirjallisuuskartoituksen tuottamissa tuloksissa yksittäiset teokset vaikuttavat paljon enemmän kategorioiden osuuteen. Esimerkiksi kokonaisluvun ylivuodosta löydettiin vain kaksi teosta, jotka edustavat 6%:a kokonaismäärästä. Tämä vastaisi 14.7:ää raportoitua haavoittuvuutta haavoittuvuusdatassa. Siksi pienimmät kategoriat voivat olla hieman vääristäviä.

Työn tuloksia voi verrata aikaisempaan tutkimukseen, Murtazan vuonna 2016 julkaistuun tutkimukseen (Murtaza, S. S. et al. 2016), kategorioissa on joitain yhtäläisyyksiä. Tutkimuksessa tarkasteltiin haavoittuvuuskatteja vuosilta 2009-2014, mutta dataa ei

oltu rajoitettu vain tietomurtoihin johtaneisiin haavoittuvuuksiin. Murtazan tutkimuksessa suurin kategoriatyyppejä kaikkina muina vuosina kuin 2014 oli ”buffer errors”. Vuonna 2014 ”cryptographic errors” pomppasi ykkössijalle. Tutkimuksessa spekuloidtiin, että se voi johtua siitä, että vuonna 2014 lisättiin paljon Android-aplikaatioihin liittyviä haavoittuvuuksia, joita ei ollut edellisellä vuonna. Myös injektioiden yhteenlaskettu määrä oli top-3:n joukossa useina vuosina. Injektioiden osuus oli laskenut huomattavasti vuosina 2009-2011. Sitä ennen injektiot olivat suurin osuus haavoittuvuuksista. Injektiot, muistinkäsittelyn virheet ja syötön validointi ovat suurimmat kategoriat sekä Murtazan tutkimuksessa, kuin myös tässä työssä.

Toinen samantyyppinen tutkimus on Richard Kuhnin julkaisema tarkastelu NVD-katalogissa listattujen haavoittuvuuksien trendeistä vuosilta 2008-2016 (Kuhn, R. et al. 2017). Tähän tutkimukseen verrattuna näyttää siltä, että suurin osuus ohjelmiston implementaatiosta johtuvista haavoittuvuuksista on muistinkäsittelyn virheet. Suurin eroavaisuus Kuhnin tuloksiin on Cross-Site scriptingissä. Tämä ero selittyy todennäköisesti sillä, että KEV-katalogissa on tähän mennessä raportoituna vain 8 haavoittuvuutta, jotka on linkitetty vain cross-site scriptingille (CISA Known Exploited Vulnerabilities, 2022). Ero tämän työn tuloksiin selittyy siis työn rajauksista vain sellaisiin haavoittuvuuksiin, jotka ovat listattuna KEV-katalogissa. Jos Kuhnin tutkimuksessa olisi käytetty samoja kriteerejä, tulokset todennäköisesti näyttäisivät erilaisilta.

Työn tulokset viittaavat siihen, että haavoittuvuuksiin liittyvä tutkimus on melko hyvin linjassa todellisuuden kanssa viimeisen viiden vuoden ajalta. Syötön validointivirheet ovat aliedustettuja tutkimustiedossa, kun taas toiminnonkulun virheet ovat yliedustettuja. Vertailu muihin vertaisarvioituihin haavoittuvuusdatan analyysihin viittaavat siihen, että muistinkäsittelyn virheet ovat aiheuttaneet haavoittuvuuksia menneisydessäkin. Tämän työn tuottama tieto voi olla hyödyllistä tutkijoille, jotka etsivät tutkimusaukkoja haavoittuvuuksiin liittyen. Työ antaa myös tilastotietoa hyväksikäytetyistä haavoittuvuuksista, josta voi olla hyötyä ohjelmoijille haavoittuvuuksien välttämiseksi omissa projekteissaan sekä ylesimpien haavoittuvuustyyppien tunnistamiseen koodin tarkastelussa. Jatkossa tietoa olisi hyvä päivittää aika ajoin, koska uudet teknologiat ja työskentelytavat voivat aiheuttaa muutoksia siinä, minkälaisia haavoittuvuuksia tietomurroissa käytetään hyväksi tulevaisuudessa. Haavoittuvuuksiin liittyviä kirjallisuuskatsauksia tulisi tehdä myös muista tietokannoista laajemman kuvan tuottamiseksi.

## 5 Yhteenveto

Työn tavoitteena oli ensin vastata tutkimuskysymykseen ”Minkätyyppisiä haavoittuvuuksia on tarkasteltu vertaisarvioituissa tieteellisissä julkaisuissa viimeisen viiden vuoden aikana?” Työn seurauksena saatiin selville, että viimeisen viiden vuoden aikana tieteelliset julkaisut ovat käsitelleet eniten muistinkäsittelyn virheitä, injektioita, suoritusjärjestyksen virheitä, kryptografisia virheitä, kolmannen osapuolen komponenttien käyttöä, syötön validointia ja kokonaisluvun ylivuotoja.

Toinen tutkimuskysymys, johon tämä työ pyrkii vastaamaan on ”Minkätyyppiset haavoittuvuudet ovat olleet yleisimpiä viimeisen viiden vuoden aikana sellaisissa tapauksissa, joissa haavoittuvuus on johtanut ainakin yhteen tietomurtoon?” Tuloksena saatiin selville, että oikeassa maailmassa tietomurtoja aiheuttaneet haavoittuvuudet ovat liittyneet muistinkäsittelyvirheisiin, injektioihin, syötön validointiin ja suoritusjärjestyksen virheisiin, kryptografisiin virheisiin ja kokonaisluvun ylivuotoihin.

Työssä toteutettiin systemaattinen kirjallisuuskartoitus ohjelmistohaavoittuvuuskategorioiden tarkastelusta vertaisarvioituissa tieteellisissä julkaisuissa vuosien 2018-2022 aikana. Kartoituksen tulosta verrattiin todellisuudessa tapahtuneiden tietomurtojen taustalla oleviin haavoittuvuuksista kerättyyn dataan. Kartoitus toteutettiin luomalla hakutermi, ja käyttämällä sitä LUT Primo-palvelussa. Tuloksista karsittiin pois epärelevantit teokset, ja teoksissa mainitut haavoittuvuustyypit kirjattiin ylös. Haavoittuvuudet kategorisoitiin, jonka jälkeen niistä tehtiin visualisaatio.

Toteutuneet haavoittuvuudet ja niihin liittyvät ohjelmistoheikkoudet haettiin NVD-tietokannan API:sta Python-ohjelman avulla, jonka jälkeen tiedot siirrettiin Exceliin, ja niistä luotiin vastaavat visualisaatiot kuin kirjallisuuskartoituksen datasta. Aineistoja verrattiin, ja tuloksena huomattiin, että kirjallisuus vastaa melko hyvin toteutuneita haavoittuvuuksia. Suurimmat kategoriat täsmäsivät molemmissa aineistoissa, mutta pienemmissä kategorioissa oli eroja. Erot voivat johtua pienen otoskoon aiheuttamasta vääristymästä.

Työn tuloksena selvitetiin, että tutkimuksissa käsitellyt ohjelmistohaavoittuvuudet ovat pitkälti linjassa tietomurtoihin johtaneiden haavoittuvuuksien kanssa. Muistinkäsittelyn

virheet sekä injektiot ovat taustalla suurimmassa osassa NVD-katalogin tietomurtoihin johtaneissa haavoittuvuuksista. Samat kategoriat ovat myös eniten käsiteltyjä tarkastelluissa tutkimuksissa.

## Lähteet

Abdullah, N. et al. (2021) 'SQL Injection Prevention in Web Application: A Review'. Advances in Cyber Security. [Verkkokirja]. Singapore: Springer Singapore Pte. Limited. 1487 s.568-585

Awad, M et al. (2019) Security vulnerabilities related to web-based data. Telkomnika [Verkkokirja] 17 (2) s.852-856

Chekole, E. G. et al. (2020) CIMA: Compiler-Enforced Resilience Against Memory Safety Attacks in Cyber-Physical Systems. Computers & security. [Verkkolehti] 94 101832.

Common Weakness Enumeration 2022 [Verkkosivu] [Viitattu 15.11.2022] Saatavilla: <https://cwe.mitre.org/data/index.html>

Conti, M. et al. (2021) SoK: Secure Memory Allocation. Cryptology and Network Security: 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings. Vol. 13099.[Verkkokirja] Springer International Publishing AG. s.372-391

CISA Known Exploited Vulnerabilities Catalog 2022 [Verkkosivu] [Viitattu 8.12.2022] Saatavilla: <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>

CVE.org, Glossary 2022 [Verkkosivu] [Viitattu 8.11.2022] Saatavilla: <https://www.cve.org/ResourcesSupport/Glossary>

Cvedetails.com 2022 [Verkkosivu][Viitattu 29.3.2023] Saatavilla: <https://www.cvedetails.com/browse-by-date.php>

Dang, T. K. et al. (2021) 'Evading Security Products for Credential Dumping Through Exploiting Vulnerable Driver in Windows Operating Systems'. Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4. 0 Applications. [Verkkokirja]. 1500 s.486-495.

Elder, S. et al. (2022) Do I really need all this work to find vulnerabilities?: An empirical case study comparing vulnerability detection techniques on a Java application. Empirical software engineering: an international journal. [Verkkolehti] 27, 154 Saatavilla:

[https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_gale\\_infotracademiconefile\\_A712812309](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_gale_infotracademiconefile_A712812309)

Filus, K. & Domańska, J. (2023) Software vulnerabilities in TensorFlow-based deep learning applications. *Computers & security*. [Verkkolehti] 124 102948.

Goel, S. & Shawky, H. A. (2009) Estimating the market impact of security breach announcements on firm values. *Information & management*. [Verkkolehti] 46 (7), 404–410. [Viitattu 8.11.2022] Saatavilla:

[https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_journals\\_237019235](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_journals_237019235)

Hinz, O. et al. (2015) The influence of data theft on the share prices and systematic risk of consumer electronics companies. *Information & management*. [Verkkolehti] 52 (3), 337–347. [Viitattu 8.11.2022] Saatavilla:

[https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_journals\\_1664473889](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_journals_1664473889)

Humayun, M. et. al. (2020) Cyber Security Threats and Vulnerabilities: A Systematic Mapping Study. Saatavilla:

[https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_journals\\_2386945004](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_journals_2386945004)

Jiang, X. et al. (2020) An Experimental Analysis of Security Vulnerabilities in Industrial IoT Devices. *ACM transactions on Internet technology*. [Verkkolehti] 20 (2), artikkeli 16

Juma'h, A. H. & Alnsour, Y. (2020) The effect of data breaches on company performance. *International journal of accounting and information management*. [Verkkolehti] 28 (2), 275–301. [Viitattu 8.11.2022] Saatavilla: [https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_crossref\\_primary\\_10\\_1108\\_IJAIM\\_01\\_2019\\_0006](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_crossref_primary_10_1108_IJAIM_01_2019_0006)

Kela, R. et al. (2022) IMPLEMENTATION OF CYBER SECURITY ATTACKS AND STRATEGIC MITIGATION MECHANISMS. *International journal of advanced research in computer science*. [Verkkolehti] 13 (4), 28–34.

Kuhn, R. Et. al. An Analysis of Vulnerability Trends, 2008-2016. Saatavilla: [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=923379](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=923379)

López Vivar, A. et al. (2020) An Analysis of Smart Contracts Security Threats Alongside Existing Solutions. *Entropy*. [Verkkolehti] 22, 203

LUT Primo, Database search 2023 [Verkkosivu] [Viitattu 12.2.2023] Saatavilla: [https://lut.primo.exlibrisgroup.com/discovery/dbsearch?query=contains,dbcategory.&tab=search\\_slot&sortby=title&vid=358FIN\\_LUT:LUT&offset=30&databases=category,LUT](https://lut.primo.exlibrisgroup.com/discovery/dbsearch?query=contains,dbcategory.&tab=search_slot&sortby=title&vid=358FIN_LUT:LUT&offset=30&databases=category,LUT)

Marin, E. et al. (2022) Serverless computing: a security perspective. *Journal of cloud computing : advances, systems and applications*. [Verkkolehti] 11 (1), 1–12.

Mathas, C.-M. et al. (2021) On the Design of IoT Security: Analysis of Software Vulnerabilities for Smart Grids. *Energies*. [Verkkolehti] 14 (10), 2818.

MITRE.org, CVE and NVD Relationship 2022 [Verkkosivu] [Viitattu 8.11.2022] Saatavilla: [https://cve.mitre.org/about/cve\\_and\\_nvd\\_relationship.html](https://cve.mitre.org/about/cve_and_nvd_relationship.html)

Murtaza, S. S. et al. (2016) Mining trends and patterns of software vulnerabilities. *The Journal of systems and software*. [Verkkolehti] 117, 218–228. [Viitattu 8.11.2022] Saatavilla: [https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_miscellaneous\\_1825477671](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_miscellaneous_1825477671)

NIST.gov, NVD Categories 2022 [Verkkosivu] [Viitattu 5.12.2022] Saatavilla: <https://nvd.nist.gov/vuln/categories>

OWASP.org, Input Validation Cheat Sheet, 2023 [Verkkosivu] [Viitattu 19.2.2023] Saatavilla: [https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)

Palit, T. et al. (2020) Mitigating Data-only Attacks by Protecting Memory-resident Sensitive Data. *Digital threats*. [Verkkolehti] 1 (4), Artikkelin 20.

Petersen, K. Et al. (2008) Systematic mapping studies in software engineering. [Viitattu 8.11.2022] Saatavilla: [https://www.scienceopen.com/document\\_file/5c16e1ce-c17c-4035-b7db-080757194d49/ScienceOpen/001\\_Petersen.pdf](https://www.scienceopen.com/document_file/5c16e1ce-c17c-4035-b7db-080757194d49/ScienceOpen/001_Petersen.pdf)

Pirry, C. et al. (2020) A Review of Memory Errors Exploitation in x86-64. *Computers*. [Verkkolehti] 9 (2), 48.

- Raducu, R. et al. (2020) Collecting Vulnerable Source Code from Open-Source Repositories for Dataset Generation. *Applied sciences*. [Verkkolehti] 10 (4), 1270.
- Sadhu, P. K. et al. (2022) Internet of Things: Security and Solutions Survey. *Sensors*. [Online] 22 (19), 7433.
- Sayeed, S. et al. (2019) Control-Flow Integrity: Attacks and Protections. *Applied sciences*. [Verkkolehti] 9 (20), 4229.
- Sekerinski, E. et al. (2020) ‘Call Me Back, I Have a Type Invariant’. *Formal Methods. FM 2019 International Workshops*. [Verkkokirja]. Switzerland: Springer International Publishing AG. s. 325–336.
- Stiakakis, E. et al. (2016) Users’ perceptions about mobile security breaches. *Information systems and e-business management*. [Verkkolehti] 14 (4), 857–882. Saatavilla: [https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_miscellaneous\\_1864572322](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_miscellaneous_1864572322)
- Subhan, F. et al. (2022) A deep learning-based approach for software vulnerability detection using code metrics. *IET software*. [Verkkolehti] 16 (5), 516–526. [Viitattu 8.11.2022] Saatavilla: [https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_doaj\\_primary\\_oai\\_doaj\\_org\\_article\\_e04b2d33932743a187e6c12e560d924c](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_doaj_primary_oai_doaj_org_article_e04b2d33932743a187e6c12e560d924c)
- Tayaksi, C. et al. (2022) The financial impacts of information systems security breaches on publicly traded companies: reactions of different sectors. *Journal of enterprise information management*. [Verkkolehti] 35 (2), 650–668. [Viitattu 8.11.2022] Saatavilla: [https://lut.primo.exlibrisgroup.com/permalink/358FIN\\_LUT/vvk1gv/cdi\\_proquest\\_journals\\_2636213708](https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/vvk1gv/cdi_proquest_journals_2636213708)
- Teodorescu, C. A. (2022) Perspectives and Reviews in the Development and Evolution of the Zero-Day Attacks. *Informatica economica*. [Verkkolehti] 26 (2/2022), 46–56.
- Tiwari, A. et al. (2020) A Large Scale Analysis of Android — Web Hybridization. *The Journal of systems and software*. [Online] 170 110775.

Yaacoub, J.-P. A. et al. (2022) Robotics cyber security: vulnerabilities, attacks, countermeasures, and recommendations. *International journal of information security*. [Verkkolehti] 21 (1), s.115–158.

## Liite 1: Python-lähdekoodi datan keräämiseen NIST NVD-tietokannasta

```
## This program creates a text file with all vulnerabilities in the NVD database which have a KEV associated with them
## starting from a specified date.
## The text file has the date of the vulnerability's publishing to the database, its description and the CWE weakness
## class associated with the vulnerability.
## The text file is semicolon separated

import requests
from datetime import datetime

startDateString = "01/01/2018" ## you can change the start date from which onwards the results get shown here. For-
## mat dd/mm/yyyy
startDate = datetime.strptime(startDateString, ("%d/%m/%Y"))

api_url = "https://services.nvd.nist.gov/rest/json/cves/2.0?hasKev"
try:
    response = requests.get(api_url)
except ConnectionRefusedError as err:
    print(
        "Error connecting to API. Note that NVD API has rate limit of 5 requests in 30 a rolling second window"
        + err
    )
data = response.json()
response.close()

vulnerabilities = data["vulnerabilities"]
outputfile = open("cwes.txt", "w", encoding="UTF-8")

i = 0
for vulnerability in vulnerabilities:
    outputString = ""
    try:
        j = 0
        cve = vulnerabilities[i]["cve"]
        publishtime = datetime.strptime(cve["published"][:13], "%Y-%m-%d")
        if publishtime >= startDate:
            outputString = (
                str(publishtime)
                + ";"
                + (cve["descriptions"][0]["value"]).replace(";", ",")
                + ";"
            )
            for weakness in cve["weaknesses"][0]["description"]:
                outputString = (
                    outputString + cve["weaknesses"][0]["description"][j]["value"]
                )
                j = j + 1
            outputString = outputString + "\n"
        i = i + 1
    except KeyError:
        print("Error with current CVE")
    outputfile.write(outputString)
    print("Added CVE")
print("Done! CVEs in file 'cwes.txt'")
outputfile.close()
```