Tampereen yliopisto

Juho Vähätalo

TIETOTURVAHAAVOITTUVUUDET VERKKOSOVELLUSTEN TIETOTURVA-TESTAUKSISSA

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Jukka Koskinen
Tarkastaja: Marko Helenius
Marraskuu 2021

TIIVISTELMÄ

Juho Vähätalo: Tietoturvahaavoittuvuudet verkkosovellusten tietoturvatestauksissa
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-Ohjelma
Marraskuu 2021

Verkkosovellukset mahdollistavat nykyajan toimintoja ja ovat välttämättömiä yhteiskunnalle. Niiden avulla toimivat monet yritykset, organisaatiot ja kansalaispalvelut. Muun muassa näistä syistä ne ovat myös kiinnostava kohde rikolliselle ja haitalliselle toiminnalle. Verkkosovellusten tietoturvalla suojataan niin dataa kuin ihmisiä.

Osana verkkosovellusten tietoturvallisuuden varmistamista on tietoturvatestaus. Hyökkääjän näkökulmasta suoritettu tietoturvatestaus luo kuvaa verkkosovelluksen haavoittuvuuksista ja tietoturvapuutteista. Tietoturvatestauksista saadun tiedon perusteella on mahdollista korjata niissä esiintyviä haavoittuvuuksia ja suunnitella ennakoidusti turvallisia verkkosovelluksia.

Tämän diplomityön tarkoituksena oli tutkia verkkosovellusten tietoturvahaavoittuvuuksien esiintymistä. Tutkimuksessa on kerätty tietoa verkkosovellusten tietoturvatestauksissa havaituista tietoturvahaavoittuvuuksista. Tutkimuksessa vertaillaan verkkosovellusten haavoittuvuuksien esiintymistä suhteutettuna eri sovelluskategorioissa.

Tutkimuksen avulla on todettavissa verkkosovelluksen monimutkaisuuden vaikuttavan vakavien tietoturvahaavoittuvuuksien esiintymiseen. Monimutkaisemmat verkkosovellukset sisälsivät enemmän vakavia tietoturvahaavoittuvuuksia. Tutkimuksen tuloksissa yleisimmin esiintyvät verkkosovellusten haavoittuvuudet liittyivät datan vuotamiseen sovelluksesta. Yleisimmin esiintyviä merkittäviä tietoturvahaavoittuvuuksia olivat cross-site scripting -haavoittuvuudet.

Avainsanat: Tietoturva, tietoturvatestaus, verkkosovellus, tietoturvahaavoittuvuus, kyberturvallisuus

ABSTRACT

Juho Vähätalo: Security vulnerabilities in Web services found in security assessments
Master's Thesis
Tampere University
Master's Programme in Information Technology
November 2021

Web applications are in a vital role in modern times functional society. Web applications enable many services in the field of business, organizations, and citizen services. Because of this, they are also an interesting target for criminal and malicious activities. Secure web applications protect data and people.

Part of ensuring the security of web applications is security assessment. Security assessment conducted by simulating an offensive actor surfaces possible security vulnerabilities and flaws. With security vulnerabilities from security assessment, it's possible to mitigate security threats.

The purpose of this master's thesis was to study web application security vulnerabilities occurrences. For this study, information about vulnerabilities found in web application security assessments was collected. The study compares occurrences of web application vulnerabilities in different types of web applications.

The results of this study indicate that more complicated and multidimensional web applications are prone to having more security vulnerabilities. Complicated and multidimensional web applications also had more significant vulnerabilities. In this study the most common vulnerabilities were vulnerabilities related to data disclosure or leakage. The most common significant vulnerabilities were cross-site scripting vulnerabilities.

Keywords: Information security, information security assessment, penetration testing, Web application, web vulnerabilities

ALKUSANAT

Oloni tällä hetkellä on äärimmäisen kiitollinen. Olen saavuttamassa valmistumisen diplomi-insinöörin opinnoista, mikä on eräänlainen saavutuspiste elämässäni. Kaikki alkoi elokuussa 2013, kun aloitin Tampereen teknillisessä yliopistossa nousten ensin Festian portaita. Noiden portaiden jälkeen olen ponnistanut Tampereen Hervannassa ylöspäin ehjempänä ihmisenä ja kokenut elämässä paljon uutta ja ihanaa. Yli kahdeksanvuotisen opiskelumatkan aikana olen kehittynyt osaavammaksi sekä taidoiltani, että ihmisenä. Yksin en kuitenkaan olisi saavuttanut mitään.

Kiitokset kohdeyritykselle diplomityön aiheesta ja tuesta. Olen saanut teiltä arvokasta tukea ja apuja opiskelujeni loppuun saattamiseksi. Kiitokset työyhteisölleni, olen onnellisessa asemassa saada työskennellä miellyttävässä porukassa. Kiitokset myös työni ohjaajalle Jukka Koskiselle sekä muille Tampereen yliopistossa minua ohjanneille. Tukenne opiskelujeni eteen on ollut tärkeää minulle.

Suuri kiitos vuosien varrella kertyneille opiskelukavereille. Useat iloiset muistot, joita olen kerryttänyt kanssanne, eivät hevillä unohdu. Erityisesti kiitokset Tampereen TietoTeekkarikillan ystävilleni, TEA-club kerhon ystävilleni, sekä oman aloitusvuoden kaveriryhmälle Ajankäytölle. Ystävyys jatkuu opiskeluaikojen jälkeenkin, vaikka ajanjaksoni Tampereella on päättymässä.

Lopuksi haluan kiittää rakasta perhettäni ja puolisoani. Kiitos kaikesta ajastanne ja panostuksestanne minuun. Teiltä saamani kannustus ja tuki kaikissa elämäni vaiheissa on kultaakin kalliimpaa.

Helsingissä, 18.11.2021

Juho Vähätalo

SISÄLLYSLUETTELO

LYHENTEET

API	Application programming interface, Ohjelmointirajapinta
CGI	Common Gateway Interface, verkkosovellustekniikka
CRM	Customer Relationship Management, asiakkuudenhallintajärjestelmä
CVE	Common Vulnerabilities and Exploits, kyberturvallisuus haavoittuvuus ja hyväksikäyttökeinojen standardisoitu yhteisön ylläpitämä kirjasto
CWE	Common Weakness Enumeration, Tunnettujen kyberhaavoittuvuuksien ja heikkouksien standardisoitu yhteisön ylläpitämä kirjasto
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä
HTTPS	Hypertext Transfer Protocol Secure, Selainten käyttämä tiedonsiirtoprotokolla
HTML	HyperText Markup Language, avoimen standardin merkintäkieli
IBAN	International Bank Account Number, kansainvälinen pankkitilinumero standardi
JSON	JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen
JSP	Jakarta Service Pages, tekniikka dynaamisille verkkosovelluksille
OWASP	Open Web Application Security Project, avoin verkkoturvallisuusprojekti
SQL	Structured Query Language, standardoitu relaatiotietokannan kyselykieli
SSH	Secure Shell, salattujen yhteyksien mahdollistava tietoliikenneprotokolla
SSL/TLS	Security Sockets Layer/Transport Layer Security
URL	engl. Uniform Resource Locator, verkkosivun osoite
WASC	Web Application Security Consortium, verkkosovellusten tietoturvaan keskittyvä yhteisö
WWW	World Wide Web, hypertekstijärjestelmä
XSS	Cross-site scripting, tietoturvahaavoittuvuus

1. JOHDANTO

Verkkosovellukset mahdollistavat nykyaikaisessa maailmassa monenlaisia toimintoja, jotka palvelevat sekä yksilöitä että yhteisöjä. Ne ovat lähes korvaamattomassa asemassa mahdollistaen nopean ja käytännöllisen tiedonsiirron. Tässä valossa verkkopalveluiden tietoturvallisuus on avainasemassa yhteiskunnan toiminnan kannalta.

Mahdolliset tietoturvahaavoittuvuudet ovat omiaan aiheuttamaan harmia verkkosovelluksen toiminnalle ja sen myötä myös verkkosovellusten loppukäyttäjille ja omistajille. Verkkosovelluksen turvaaminen tai turvallisuuden varmistaminen on vaikea, lähes mahdoton työ. Yksi keino tämän saavuttamiseksi on tietoturvatestaaminen, jolla vakuututaan verkkosovelluksen turvallisuuden eteen tehdyn työn tuloksista.

Tämä tutkimus kartoittaa verkkosovellusten tietoturvatestausten tuloksia ja arvioi, minkä tyylisiä tietoturvahaavoittuvuuksia niissä yleisesti on havaittu. Tutkimuksen kohteena oli verkkosovellusten tietoturvatestausraportit sekä niissä esiintyvät tietoturvahaavoittuvuudet. Tutkimuksessa tarkasteltiin tietoturvahaavoittuvuuksien esiintymistä verkkosovelluskategorioittain.

Tutkimuksessa hyödynnettiin suomalaisen kyberturvallisuusalan yrityksen verkkosovelluksiin tehtyjen tietoturvatestauksien tuloksia. Tutkimuksessa kerättiin tietoturvatestausraporttien tietoja ja kartoitettiin niiden avulla mahdollisia samankaltaisuuksia verkkosovelluksissa tietoturvatestaushaavoittuvuuksien näkökulmasta. Tietoturvatestausraportit sisälsivät muun muassa arvosanoja verkkosovelluksen kokonaisturvallisuuksista sekä tietoa testauksen aikana löydetyistä tietoturvahaavoittuvuuksista. Tämän tutkimuksen suurinta antia on yleiskuvaus suomalaisen kyberturvallisuusorganisaation tietoturvatestauksissa havaituista tietoturvahaavoittuvuuksista ja niiden esiintyvyydestä verkkosovelluskategorioittain.

Aineiston tarjoavalla yrityksellä on pitkä historia suomalaisena kyberturvallisuuspalveluiden tarjoajana. Yrityksen erityisosaamisalaa on tekninen tietoturvatestaus. Organisaation toiminta-alueena on pääsääntöisesti suomalaiset yritysasiakkaat, mutta asiakaskuntaan kuuluu myös kansallisia palveluita tarjoavia viranomaistahoja sekä yhdistyksiä.

Tässä tutkimuksessa on hyödynnetty tietoturvatestauksissa havaittuja tietoturvahaavoittuvuuksia. Tämän työn tuloksista on pääteltävissä, että verkkosovellusten yleisimmät tie-

toturvahaavoittuvuudet ovat konfiguraatiohaavoittuvuuksia. Työstä on myös pääteltävissä, että korkean vaikutuksen omaavista tietoturvahaavoittuvuuksista yleisimpiä ovat ohjelmointihaavoittuvuudet.

Tämän työn lukijalta odotetaan ymmärrystä verkkosovellusten perusominaisuuksista sekä yleistä ymmärrystä tietoturvatestaamisesta. Tutkimuksessa käytettävä terminologia on vahvasti sidoksissa englanninkielisiin termeihin, johtuen tietoturva-alan käytänteistä. Lukijan ei tarvitse olla täysin perehtynyt jokaiseen tietoturvahaavoittuvuuteen ja mitä haavoittuvuus mahdollistaa potentiaalisille verkkosovellusta vastaan hyökkääville tahoille.

Diplomityö on jaettu neljään lukuun. Luvussa kaksi on teoriaa verkkosovelluksista ja niiden tietoturvatestaamisesta. Lukijalle avataan käsitteistöä sekä käytänteitä, miten tietoturvatestaamista voi toteuttaa. Luvussa on myös teoriaa tietoturvahaavoittuvuuksista ja niiden esiintymisistä. Luvussa kolme paneudutaan tutkimusmenetelmään ja tutkimusaineistoon. Luvun antia on, kuinka tutkimus on toteutettu.

Luvussa neljä käydään läpi tutkimuksen tuloksia. Luvussa myös käsitellään, mitä tuloksista on pääteltävissä. Viimeinen luku viisi on varattu yhteenvedolle koko tutkimuksesta. Luvussa pohditaan tämän työn vaikutuksista, sekä arvioidaan itse tutkimusta ja pohditaan mahdollisia jatkotutkimuskohteita.

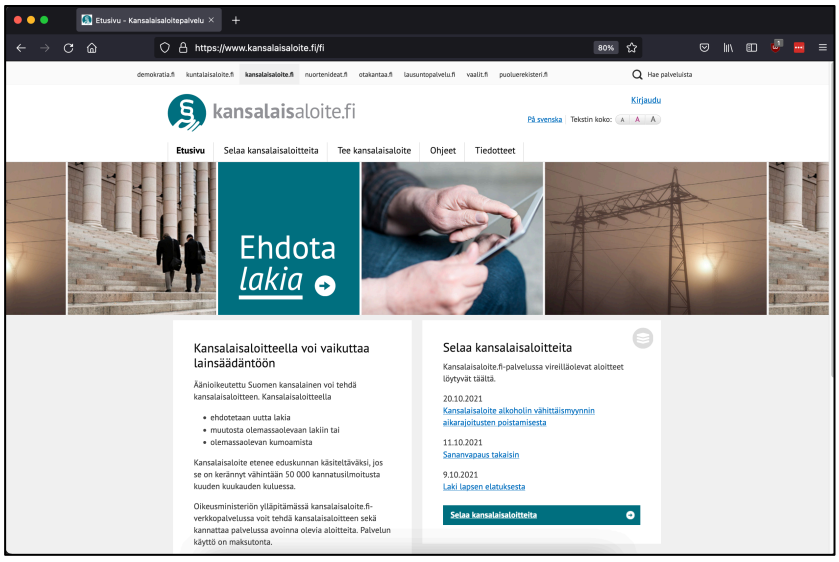
2. VERKKOSOVELLUSTEN TIETOTURVATESTAUS

Tässä luvussa käydään läpi tietoturvatestaamisen teoriaa. Luvussa syvennytään verkkosovellusten periaatteisiin ja määritelmiin. Luvussa esitellään verkkosovellusten toiminnallisuuksia ja niiden mahdollisia heikkouksia tietoturvan näkökulmasta. Luvussa paneudutaan myös tietoturvahaavoittuvuuksiin ja niiden vaikutuksiin verkkosovelluksissa.

Näiden lisäksi luvussa on kuvattuna teoriaa tietoturvatestauksista sekä yleisesti, että tutkimuksen kontekstissa. Verkkosovellusten tietoturvatestaaminen on mahdollista toteuttaa usealla eri keinolla ja tässä luvussa avataan, miten tutkimuksen aineiston tuottaneet tietoturvatestaukset on toteutettu ja miten ne pääsääntöisesti eroavat muista tietoturvatestausmetodeista.

2.1 Verkkosovellus

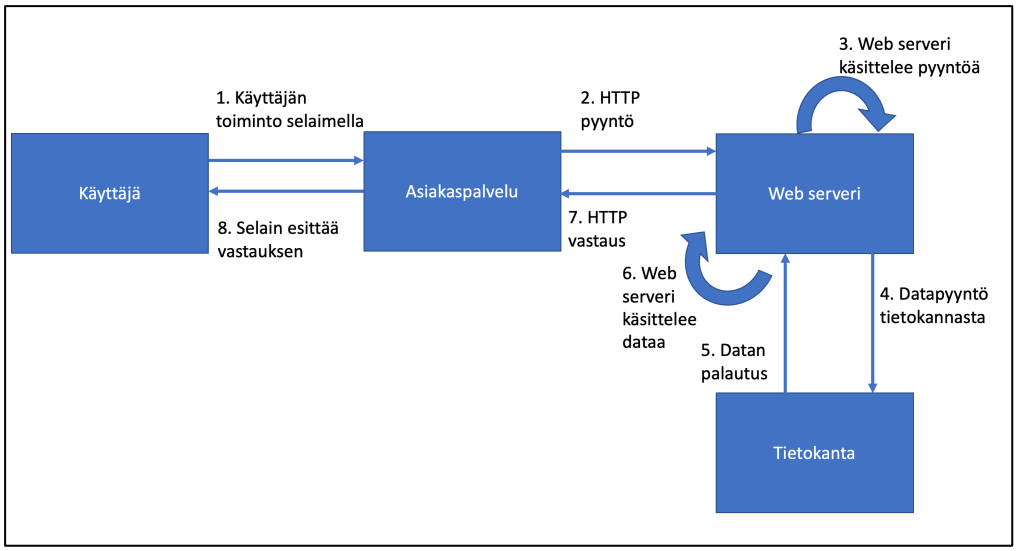
Verkkosovellus tai Web-sovellus on tietojärjestelmä, joka on Internetyhteyden avulla käytettävissä kohdeyleisölleen. Verkkosovelluksen tarkoituksena on palvella käyttäjiään, sidosryhmiään, omistajiaan tai muita sidosjärjestelmiä suunnitellun mukaisesti. Yksi yleisesti tunnetuista verkkosovellusmuodoista on verkkokauppa. Verkkokauppa mahdollistaa kaupassa asioimisen verkkoselaimen avulla. Toisin kuin tavallinen kauppa, verkkokauppoja eivät rajoita esimerkiksi sijainnit tai aukioloajat. Tämä tarjoaa asiakkaalle enemmän valinnanvapautta kuin perinteinen kauppa.



Kuva 1. *Kuvakaappaus kansalaisaloite.fi verkkosovelluksen etusivunäkymästä*

Verkkokaupan lisäksi muita verkkosovellusmuotoja ovat muun muassa organisaatioiden toiminnanohjausjärjestelmät, verkossa toimivat tiedostonhallintapalvelut ja organisaatioiden asiakkailleen tarjoamat asiointipalvelut. Kuvassa 1 on esitettynä kuvakaappaus esimerkkiverkkosovelluksen etusivusta. Yhteistä näille on, että käyttäjät hyödyntävät näitä palveluita verkkoselaimella tai palveluun tarkoitetulla sovelluksella. [1]

Leon Shaktar ja Rich Rosen kuvaavat Web-sovelluksia seuraavasti: Web-sovellus on asiakas-palvelija -mallin sovellus, joka käyttää verkkoselainta asiakasohjelmanaan ja web-palvelinta palvelijaohjelmanaan. Se toimittaa interaktiivisia palveluita Internettiin tai intranettiin hajautettujen web-palvelinten avulla. Web-palvelimesta käytetään toisinaan myös nimitystä taustapalvelin. Web-sivu periaatteessa vain tuo näytille tiedostoissa olevan sisällön. Verkkosovelluksia käytetään yleisesti joko henkilökohtaisella tietokoneella tai mobiililaitteella. [2]



Kuva 2. *Esimerkki dynaamisesta verkkosovelluksen käyttötilanteesta*

Kuvassa 2 on esitettynä hyvin yksinkertaistettu esimerkki tilanteesta, jossa käyttäjä käyttää dynaamista verkkosovellusta. Usein verkkosovellusta käyttää useampi kuin yksi käyttäjä ja web-palvelimia voi olla samalla sovelluksella useampia. Perinteisen tietokannan lisäksi web-palvelin voi käyttää muitakin palveluita, kuten ulkopuolisia autentikaatiopalveluita. Ulkopuolisia palveluita verkkosovellus hyödyntää rajapintojen avulla. Web-serverin tietokantojakin voi olla useampia, joista web-serveri noutaa ja vie tietoja.

Verkkosovelluksen eri komponentit kommunikoivat keskenään seuraamalla kommunikointisääntöjä, joita kutsutaan protokolliksi. Yleisiä protokollia on esimerkiksi asiakaspalvelun ja web-palvelimen väliseen kommunikointiin käytetty HTTP-protokolla, sekä tiedostonsiirtoprotokolla FTP. HTTP-protokollasta on myös käytössä turvallisempi HTTPS-protokolla, jossa protokollan kommunikaatio on salattua.

Verkkosovelluksen komponentit kommunikoivat keskenään siirtäen toisilleen dataa, joka sisältää tietoa siitä, mitä komponentti haluaa toisten komponenttien toteuttavan. Esimerkiksi selain voi lähettää HTTP-pyynnön web-palvelimelle, josta se haluaa avata nähtäväksi jonkin tietyn tiedoston. Tiedoston sijainti lähetetään tällöin osana HTTP-pyyntöä. Pyynnössä on tiedoston sijainnin lisäksi mukana muitakin tietoja, joita verkkosovellus vaatii. Esimerkiksi käyttäjän tunnistamista varten verkkosovellus saattaa vaatia validia istuntotunnistetta. Istuntotunnisteella verkkosovellus varmistaa, että käyttäjällä on oikeus avata kyseinen tiedosto.

```
GET / HTTP/1.0
Accept: text/plain
Accept: text/html
Session-Id: SID:ANON:w5.com:j45BeggZh/gD745LGeXlf-01:021
User-Agent: libwww/4.1
```

Ohjelma 1. *Esimerkki HTTP-pyynnöstä, jossa on mukana istuntotunniste (Session-Id).*

Verkkosovellukset voivat olla joko staattisia tai dynaamisia. Yleisimpiä staattisia verkkosovelluksia ovat tilattomat verkkosivut, joissa asiakaspalvelu pyytää web-palvelimelta esimerkiksi HTML-tiedoston. Palautunut HTML-tiedosto näytetään asiakaspalvelussa esimerkiksi selainohjelmassa. Tällöin käyttäjälle näytettävä sisältö selaimessa on staattista ja näkyy samanlaisena riippumatta siitä, milloin käyttäjä verkkosovelluksessa vierailee. Jos tietoja verkkosovelluksessa halutaan päivittää, tulee se tehdä manuaalisesti päivittämällä HTML-tiedostoa.

Dynaaminen verkkosovellus eroaa staattisesta siten, että se on ohjelmoitu päivittämään itse itseään. Dynaamisessa verkkosovelluksessa web-palvelin noutaa tietoa esimerkiksi tietokannasta, jota se sitten käsittelee. Esimerkiksi verkkosovellus, joka näyttää käyttäjälle reaaliajassa kellonajan, noutaa päivitettävän tiedon näytettäväksi käyttäjälle. Yleisempiä dynaamisia tiedostomuotoja ovat asp.net, php, jsp ja .cgi.

Verkkosovelluksia kehitetään vastaamaan olemassa olevaan tarpeeseen. Organisaatiot ovat ajan saatossa kehittäneet verkkosovelluksia pyrkien saamaan niiden hyödyntämisestä etua. Kachan listaa seitsemän eri syytä, miksi organisaatiot ovat siirtyneet tarjoamaan palveluitaan verkkosovelluksina [3]. Näitä ovat muun muassa: kasvanut tehokkuus, palveluntarjonnan mahdollisuus vuorokauden ympäri ja palvelun parempi skaalautuvuus. Verkkosovellukset ovat hänen mukaansa myös tärkeämmässä osassa organisaation liiketoimintaa.

Verkkosovelluksia kehitetään itse organisaatioiden toimesta tai ostamalla kehitystyö palveluna. Molemmissa tapauksissa verkkosovellusten kehitystyötä ohjaa lähtökohtaisesti organisaation tarpeet. Tarpeet on hyvä määrittää, jotta valmis verkkosovellus vastaisi niitä mahdollisimman hyvin. Esimerkiksi kun verkkosovellus on tarkoitettu käytettäväksi ainoastaan organisaation sisäisille käyttäjille, on määritettävä verkkosovelluksen ominaisuudet ja toiminnat juuri näitä käyttäjiä ajatellen.

Verkkosovelluksille ei ole olemassa yhtä ainoaa kategorisointia, mutta ne voidaan kategorisoida esimerkiksi sen perusteella, kuinka dynaamisia ne ovat. Verkkosovellus voi tällöin olla joko dynaaminen tai staattinen. [4]

Toisaalta verkkosovellukset voidaan kategorisoida myös esimerkiksi käyttäjäryhmien mukaan. Käyttäjäryhmiä ovat esimerkiksi yksittäiset käyttäjät, asiakasorganisaatiot, tai organisaation sisäiset käyttäjät.

Tässä tutkimuksessa käytettiin verkkosovellusten kategorisointimallia, joka perustuu verkkosovellusten kompleksisuuteen. Tällöin keskiössä oli, kuinka monimutkainen verkkosovellus on. Mitä enemmän se mahdollistaa tai vaatii käyttäjiltään erilaisia toimenpiteitä, sitä enemmän siinä on myös lähtökohtaisesti toiminnallisuuksia ja kompleksisuutta. [5] [6]

2.1.1 Verkkosovellusten toiminnallisuudet ja tietoturvatestaaminen

Verkkosovelluksen toiminnallisuuksia on monia ja niitä voi olla samassa sovelluksessa useita. Alla on esiteltynä muutamia toimintoja, joita verkkosovelluksissa esiintyy. Toiminnollisuuksista on myös kuvattuna tietoturvatestaukseen liittyviä esimerkkejä.

Rajatut käyttöoikeudet ovat toiminnallisuuksia, jossa sovelluksen käyttöoikeudet ovat sidottuna käyttäjätiliin. Tällöin puhutaan käyttäjän autorisoinnista tai valtuuttamisesta. [7]. Oleellista tietoturvatestauksen kannalta on varmistua, että valtuuttaminen toimii oikein verkkosovelluksessa, eikä esimerkiksi valtuuttamaton käyttäjätili pysty käyttämään verkkosovelluksen häneltä pois rajattuja toimintoja [8]. Verkkosovelluksen tietoturvatestauksessa pyritään testausprojektin aikana saamaan valtuuttamattomalla testikäyttäjätunnuksella aikaan toimintoja, joita testikäyttäjätunnuksella ei pitäisi olla mahdollista tehdä. Verkkosovellukset, joissa ei ole tarvetta käyttäjän valtuuksien rajoittamiselle, eivät välttämättä tarvitse valtuuttamisen testaamista. Verkkosovellukset, joissa valtuuttamisen toimivuus on varmentamisen arvoista, sisältävät esimerkiksi pääkäyttäjätilin (admin-tili). Pääkäyttäjän oikeuksia ei kenen tahansa pitäisi pystyä käyttämään.

Autentikaatio tai todentaminen on menettely, jolla varmennetaan kohteen todenmukaisuus, alkuperä tai oikeellisuus [9]. Näin ollen varmistetaan, että verkkosovelluksen käyttäjä on kuka hän väittää olevansa. Autentikaation toimivuutta voidaan testata muun muassa varmistamalla asianmukainen suojaus väsytyshyökkäyksiä (brute-forcing) vastaan. Sovelluksen kirjautumissivun tulisi rajoittaa käyttäjän kirjautumisyrityksien nopeutta ja estää näin mahdollinen väsytyshyökkäys. Autentikaation testaamista vaativia sovelluksia ovat muun muassa verkkopankit, verkkokaupat ja muut transaktionaaliset sovellukset.

Testauksen keskiössä on tyypillisesti käyttäjän tekemät toiminnot ja niiden vaikutukset verkkosovellukseen. Näitä ovat muun muassa tietojen muuttaminen käyttäjän toimesta, uusien tietojen tallentaminen, tietojen poistaminen sekä tietojen päivittäminen. Käyttäjän interaktiot eivät välttämättä rajoitu vain yhteen tietuekohtaan. Sovellus voi koostua useasta toisistaan sidoksissa olevasta komponentista, joihin interaktio voi vaikuttaa. Esimerkiksi teatterin istuinpaikan varaussovellus. Oikein toimiva sovellus varmistaa, että käyttäjän varausinteraktion jälkeen, samaa istuinpaikkaa ei enää pystytä varaamaan.

Iso osa verkkosovellusten käyttäjäntoiminnoista suoritetaan sovelluksen käyttöliittymän avulla. Käyttöliittymän suojaus käyttäjän haitallisilta tai virheellisiltä syötteiltä on syytä turvata asianmukaisesti. Tietoturvatestaamisessa käyttöliittymän käyttäjän syötteen testaamiseen riippuu sovelluksen syötekohdista. Esimerkiksi syötekohtia voivat olla käyttäjätietojen vapaasti muokattavat syötekohta, viestin tai palautteen kommentointikenttä ja erilaiset hakukentät.

Tiedostojen lataamisen ja tallentamisen toiminnot mahdollistavat verkkosovelluksen käyttäjille tiedostojen tallentamisen ja lataamisen verkkosovelluksissa, esimerkiksi tiedostonjakopalveluissa. Nämä toiminnot vaativat usein käyttäjän interaktiota käyttöliittymässä. Esimerkiksi sovelluksen käyttäjätilin henkilökuvan tallentaminen sovellukseen on tiedoston tallentamisen muoto. Sovellukset lataavat tiedostoja web-palvelimelleen ja suorittavat niiden ajon asiakasohjelmassaan. Tällöin tietoturvatestauksella on hyvä varmistua, että sovellukseen lähetettävä tiedosto on ennalta määritettyjen vaatimusten mukainen. Vaatimuksia voivat olla tietyt tiedostomuodot ja tallennettavan tiedoston koko. Esimerkiksi riskinä on, että tiedoston tallentaminen sovelluksen web-palvelimelle ajetaan sisältäen mukanaan haitallista asiakasohjelmassa suoritettavaa haittakoodia. [10]

Useamman käyttäjän välisissä yhteiskäyttötoiminnoissa verkkosovellus jakaa tietoja usean eri käyttäjän välillä. Chat-sovellukset ja erilaiset tiedostonjakopalvelut ovat esimerkkejä sovelluksista, joissa on toiminnallisuuksia käyttäjien välisen tiedon jaossa. Tällöin siirrettävän tiedon, esimerkiksi chat-viestin, tulee olla toisen käyttäjän saavutettavissa, kun viesti on lähetetty. Viestien välittämisen turvallisuutta ja toimivuutta varten on sovelluksessa hyvä olla olemassa turvalliset ja luotettavat tiedonsiirtotoiminnot.

Verkkosovelluksen käyttäjä tai käyttäjät eivät aina ole sovelluksen keskiössä. Näitä ovat esimerkiksi erilaiset mittaustietoja esittävät sovellukset. Mittaustietoja ovat esimerkiksi sovelluksen käyttäjämäärä tai ulkolämpötila. Tällöin on oleellista varmistua muun muassa siitä, että mittauspisteestä verkkosovellukseen siirrettävä data on turvassa ja oikeaa. Tietoturvatestauksessa tämänkaltaisissa sovelluksissa varmistetaan muun muassa, että käytetyt siirtoprotokollat ovat riittäviä suojaamaan tietoa sen paljastumiselta ja muuttumiselta. [11]

Usein verkkosovellus kerää taustalla myös tietoa itsestään. Näitä kutsutaan lokijärjestelmiksi. Lokijärjestelmät keräävät tietoa sovelluksen tapahtumista ja virheistä. Yksittäistä lokijärjestelmän ilmaisemaa tietoa kutsutaan lokitapahtumaksi ja se on sidoksissa sovellukseen liittyneeseen tapahtumaan. Lokitapahtumien avulla on mahdollista tapahtuman jälkeen tarkastaa, miten sovellusta on käytetty ja miten sovellus on käyttäytynyt. Sovelluksen väärinkäytön jälkeen lokitapahtumien tulisi avustaa väärinkäyttötapahtuman ja syyn selvittämisessä. Mikäli lokitapahtumat eivät kerää riittävästi tietoa sovelluksen toiminnoista, on mahdollista, että sovelluksen väärinkäyttöä ei pystytä myöhemmin selvittämään.

Oikein asetetut lokihälytykset auttavat verkkosovellusta sen tietoturvallisuudessa. Lokihälytykset kytketään sovelluksen väärinkäyttötilanteisiin. Hälytykset ilmoittavat sovelluksen väärinkäytöstä, yleensä ylläpitäjätaholle. Ilmoitukset mahdollistavat reagoinnin tapahtuman jälkeen. Tietoturvatestaamisessa lokien asianmukainen toiminta on hyvä varmistaa, selvittämällä että testatut väärinkäyttötapaukset muodostavat riittävän lokitapahtuman sekä mahdollisesti aiheuttavat riittävän lokihälytyksen. [12]

2.2 Verkkosovellusten tietoturvahaavoittuvuudet

OWASP-projektin määritelmän mukaan tietoturvahaavoittuvuus (eng. vulnerabilitity) on aukko verkkosovelluksen tietoturvassa. Tietoturva-aukko mahdollistaa muun muassa haitanteon pahantahtoisen toimijan toimesta sovelluksen sidosryhmille. Sidosryhmiä voivat olla joko verkkosovelluksen omistaja, tilaaja tai muut verkkosovelluksen käyttöön liittyvät osapuolet. [13]

Haavoittuvuus on varmennettu tietoturva-aukko sovelluksessa. Toisin kuin haavoittuvuus, tietoturvahavainto voi olla esimerkiksi tietoturvatestaajan työkalun havaitsema haavoittuvuus, joka varmentamisen jälkeen voi osoittautua vaarattomaksi vaikutukseltaan tai vääräksi-todeksi. Vasta varmennettu havainto on haavoittuvuus. [14]

Mikäli verkkosovellusta vastaan hyökkäävä taho haluaa hyödyntää haavoittuvuuksia, käyttää hän tähän jotain hyväksikäyttömenetelmää. Tällöin puhutaan englanninkielisestä termistä exploit. Sanastokeskus TSK käyttää suomenkielisiä käännöksiä haavoittuvuuden hyödyntäjä tai haavoittuvuutta hyödyntävä menetelmä [15]. Haavoittuvuutta hyödyntävillä menetelmillä voidaan saavuttaa riippuen verkkosovelluksen haavoittuvuudesta sekä menetelmästä riippuen esimerkiksi verkkohyökkäyksiä. Haavoittuvuuden hyödyntäminen (engl. exploitation) ei kuitenkaan ole kaikissa tilanteissa mahdollista. Alvesin mukaan näitä tilanteita ovat [16]:

1. Hyökkääjällä ei ole riittävästi tietoa järjestelmästä haavoittuvuuden hyödyntämiseksi.
2. Haavoittuvuuden hyödyntäminen vaatii tietyt käyttöoikeudet, jotka hyökkääjältä puuttuu.
3. Haavoittuvuuden hyödyntämistä vastaan implementoidut turvallisuuskontrollit estävät haavoittuvuuden hyödyntämisen.

2.2.1 Suunnittelu-, konfiguraatio ja ohjelmointihaavoittuvuudet

OWASP jakaa haavoittuvuudet kahteen eri kategoriaan, jotka ovat suunnittelupuute ja toteutuspuute [13]. Tässä tutkimuksessa toteutetut tietoturvatestauksien haavoittuvuudet ovat jaoteltu kolmeen luokitukseen:

1. suunnitteluhaavoittuvuuksiin,
2. konfiguraatiohaavoittuvuuksiin sekä
3. ohjelmointihaavoittuvuuksiin.

Suunnitteluhaavoittuvuudet ovat kuten OWASP:n suunnittelupuutteet, eli verkkosovelluksen suunnittelussa johdosta tapahtuneita haavoittuvuuksia. Suunnitteluhaavoittuvuudet voidaan nähdä haavoittuvuuksina, joissa huolimattomuus sovelluksen määrittelyssä

tai suunnitteluvaiheessa on johtanut mahdolliseen haavoittuvuuteen. Suunnitteluhaavoittuvuudet ovat usein monimutkaisempia korjata, johtaen mahdollisesti jopa sovelluksen arkkitehtuurin uudelleensuunnitteluun.

Esimerkkejä suunnitteluhaavoittuvuuksista ovat muun muassa username harvesting -haavoittuvuudet. Tällöin sovelluksen kirjautumissivulla on haavoittuvuus, joka mahdollistaa käyttäjätunnuksien tietojen keräämisen. Näin on tilanteissa, joissa verkkosovellus palauttaa verkkosovelluksen käyttäjänimen kyselyn jälkeen erilaisen virheviestin kuin oikean käyttäjätunnuksen kohdalla. Tällöin on hyökkääjän mahdollista kokeilla eri käyttäjänimiyhdistelmiä ja pyrkiä siten keräämään virheviestien perusteella sovelluksen käyttäjistä listaa. Suunnitteluhaavoittuvuuden esimerkistä tekee, ettei käyttäjänimien virheviestien palautusta ole suunniteltu tai määritelty oikein. [17]

Konfiguraatiohaavoittuvuudet ovat pääsääntöisesti toteutuspuutteita, ja niitä esiintyy runsaasti. Näissä haavoittuvuuksissa sovelluksen asetuksia ja niiden vipuja on asetettu väärin turvallisuuden kannalta. Yksi mahdollinen syy on, ettei turvallisten konfiguraatioiden asettamista ole muistettu toteuttaa oikein ennen testausta. Toinen syy konfiguraatiohaavoittuvuuksien esiintymiselle voi johtua sovelluksen käytöstä. Jotkin sovellukset voivat vaatia esimerkiksi vanhempien selainversioiden tukemista, jolloin voi ilmetä yhteensopivuusongelmia. Konfiguraatiohaavoittuvuudet on yleisesti helpompia korjata kuin suunnitteluhaavoittuvuudet, ja joissain tapauksissa jopa yksinkertaisesti yhden asetuksen päälle kytkeminen korjaa haavoittuvuuden.

Yksi tyypillisimmistä konfiguraatiohaavoittuvuuksista on puutteelliset TLS/SSL-asetukset. Verkkosovelluksen web-palvelin voi käyttää esimerkiksi vanhentunutta ja tuen piiristä poistunutta TLS 1.0 -protokollaa uudempien sijasta. Vanhentunutta kommunikaatiota on mahdollista häiritä tai jopa kaapata ja purkaa selkokielisiksi tilanteissa, joissa hyökkääjällä on riittävästi tietotaitoa ja resursseja. Tällöin haavoittuvuuden korjaaminen vaatii uudemman TLS-version käyttöönottoa. [18]

Ohjelmointihaavoittuvuudet ovat konfiguraatiohaavoittuvuuksien tavoin pääsääntöisesti toteutuspuutehaavoittuvuuksia. Tällöin verkkosovellukselle suunniteltu tietoturvakontrolli ei toimi riittävän hyvin ollakseen turvallinen. Ohjelmointihaavoittuvuudet eroavat suunnitteluhaavoittuvuuksista nimenomaan sillä, että turvallisuuskontrolli on suunniteltu, mutta sitä ei vain ole toteutettu turvallisesti. Ohjelmointihaavoittuvuuksien korjaamisen haastavuus riippuu haavoittuvuudesta. Esimerkiksi syötteen tarkastelun uudelleensuunnittelu niin, että syöte tarkastetaan samoin kaikissa verkkosovelluksen osissa, voi olla haastavaa. Sen sijaan, jos syötteen tarkasteluun täytyy lisätä yksi tarkastettava merkkijono, on

ominaisuuden korjaaminen lähes yhtä helppoa kuin yksinkertaisimpien konfiguraatiohaavoittuvuuksien korjaaminen.

Esimerkkinä ohjelmointihaavoittuvuudesta on Insecure Direct Object Referencing (IDOR) haavoittuvuus, jossa verkkosovellus ei tarkasta kirjautuneen käyttäjän oikeuksia riittävästi. Jos tiedostojenhallintaa toteuttavassa sovelluksessa selaimen käyttämän URL:n lopussa esiintyvä tiedostoparametri seuraa helposti arvattavaa logiikkaa, voi toinen verkkosovelluksen käyttäjä kokeilla muilla tiedostoparametriarvoilla pääsyä toisiin tiedostoihin. Tällöin mahdollinen hyökkääjä pääsee käsiksi toisten käyttäjien tietoihin ilman että hänen käyttäjäänsä sidottuja oikeuksia on kunnolla varmennettu. Tällöin esimerkin verkkosovelluksessa on suunniteltuna kontrolli, joka tarkastaa, että käyttäjä on oikeutettu käyttämään palvelua. Kuitenkin käyttöoikeuksien tarkastelu saattaa pettää, jos kirjautuneen käyttäjän siirtyessä sovelluksessa toisen käyttäjän tietoihin ei käyttöoikeuksia varmenneta uudestaan. Näin ollen kontrolli on ohjelmoitu puutteellisesti. [19]

2.2.2 Tietoturvahaavoittuvuuksien vakavuus ja niiden hyödyntäminen

Haavoittuvuus itsessään ei välttämättä ole vakava haavoittuvuus. Haavoittuvuuden vakavuuteen vaikuttaa, kuinka haavoittuvuuden mahdollistama uhka arvotetaan. Yleisesti tietoturva-alalla seurataan CVSS:n haavoittuvuuspisteytysjärjestelmää. CVSS-pisteytysjärjestelmää ylläpitää Yhdysvaltojen hallinnon alaisen NIST:n sivuosasto NVD. CVSS pisteytysjärjestelmä antaa haavoittuvuudelle pisteytyksen nollasta kymmeneen. Suurempi pistearvo kuvaa vaarallisempaa haavoittuvuutta verkkosovelluksen ja sen sidosryhmien turvallisuudelle, taulukon 1 mukaisesti. [20]

Taulukko 1. *CVSS pisteytysjärjestelmä tietoturvahaavoittuvuudelle*

Vakavuus	**CVSS 3.0 arvosana**
Olematon	0,0
Matala	0,1–3,9
Keskitasoinen	4,0–6,9
Korkea	7,0–8,9
Kriittinen	9,0–10.0

NVD ylläpitää myös CVSS-laskuria, jolla haavoittuvuuden vakavuutta voi itse arvioida. Haavoittuvuuden CVSS-arvosanaan vaikuttavat muun muassa, kuinka paljon osaamista

haavoittuvuuden havaitseminen vaatii tai, kuinka korkean käyttäjätason haavoittuvuuden hyödyntäminen vaatii. [21]

Haavoittuvuuden vakavuus on myös mahdollista arvioida haavoittuvuuden löytäjän toimesta. Tämän työn aineistossa haavoittuvuuksien vakavuuden ovat arvioineet ammattitaitoiset tietoturvatestaajat osana testausta. Tietoturvatestaajat ovat pystyneet hyödyntämään myös muiden testaajien ammatillista näkemystä tai CVSS-laskuria tukemaan arviointia.

Haavoittuvuuksien vakavuuteen voi vaikuttaa muut haavoittuvuudet verkkosovelluksessa. Kaikissa tapauksissa haavoittuvuus yksinään ei välttämättä ole vakava, mutta yhdistelmänä muiden haavoittuvuuksien kanssa sen vaikutus voi olla korkeampi. Tämä on huomioitu aineiston haavoittuvuuksien arvioinnissa.

2.2.3 Haavoittuvuuksien yhdistäminen

Verkkosovelluksen haavoittuvuudet mahdollistavat toisinaan tilanteita, joissa haavoittuvuuksien yhdistäminen mahdollistaa verkkosovelluksen hyödyntämisen hyökkäävän tahon toimesta. Kyseisiä tilanteita kutsutaan englanniksi termillä *chained exploit*. Tälle ei ole vastaavaa suomenkielistä nimeä, mutta kyseessä on haavoittuvuuksien hyödyntämisen ketjuttaminen. Vothin, Whitakerin ja Evansin mukaan chained exploit on kyberhyökkäys, joka sisältää useita haavoittuvuuksien hyödyntämisen keinoja ja hyökkäyksiä. [22].

Yhdistämisellä pystytään tekemään yksittäisen haavoittuvuuden vaikutuksesta verkkosovelluksen turvallisuudelle suurempi. Esimerkiksi open redirect ja reflected XSS-haavoittuvuuksien yhdistäminen voi tietyissä tilanteissa tehdä XSS haavoittuvuuden vaikutuksesta korkeamman. Tällainen tilanne on mahdollinen, mikäli verkkosovelluksen reflected XSS-haavoittuvuudella on mahdollista saada haitallista JavaScript-koodia ajettua käyttäjän toimesta kirjautumisnäkymässä, ja mikäli JavaScript-koodilla hyökkääjän ylläpitämälle palvelimelle siirtyy käyttäjän salasana ja käyttäjänimi.

Koska aikaisempi XSS-haavoittuvuus on peilautuva, on käyttäjän epätodennäköistä syöttää itselleen vahingossa haitallinen JavaScript osana HTTP-pyyntöä. Mikäli samassa sovelluksessa on myös tietyn kaltainen open redirect -haavoittuvuus, on hyökkääjän mahdollista yhdistää se XXS-haavoittuvuuteen. Tilanteessa, missä open redirect mahdollistaa sovelluksen uudelleenohjata HTTP-pyynnön toiseen sijaintiin ilman alkuperäisen sovelluksen validointia, on hyökkääjän mahdollista huijata käyttäjää haitallisen linkin avulla. Hyökkääjän luoma haitallinen linkki on käyttäjälle erityisesti uhkaava, jos se sisältää haitallista JavaScript-koodia ja vaikuttaa riittävän aidolta klikattavaksi. Tällöin

linkki toimii oikein sovelluksessa ja käyttäjä kirjautuu oikeaan verkkosovellukseen. Samalla käyttäjän kirjautumistiedot kuitenkin välitetään eteenpäin hyökkääjän palvelimelle. Linkki täytyy toki toimittaa erikseen käyttäjälle esimerkiksi osana säköpostia.[23] [24]

Esimerkki kuvaa tilannetta, missä on monia muuttujia. Vastaavia tilanteita ei kaikissa tapauksissa aina esiinny, sillä open redirect haavoittuvuus ja reflected XSS haavoittuvuuksien yhdistelmä vaatii myös haavoittuvuuden hyväksikäytön oikeassa kohtaa. Käyttäjän kirjautumistietojen päätyminen hyökkääjälle on vakavampi vaikutus kuin esimerkiksi kuin käyttäjän nimen.

2.3 Tietoturvahaavoittuvuuksien nimeämiskäytännöt

Tietoturvahaavoittuvuuksille on useita nimiä. Sama haavoittuvuus voidaan nimetä eri tavoin. Haavoittuvuuksien nimet perustuvat yleensä kyberturva-alan käytänteisiin. Tässä tutkimuksessa verkkosovellusten tietoturvatestauksien haavoittuvuuksien nimeämisessä on hyödynnetty yleisesti tunnettujen OWASP:n [13], CWE:n [25] ja CVE:n [26] käyttämiä haavoittuvuuksien nimiä. Tämän kaltaisia ovat esimerkiksi SQL-injektiohaavoittuvuudet sekä cross-site request forgery -haavoittuvuudet.

CWE ja CVE ovat kyberturvallisuusalalla yleisesti tunnettuja tietoturvahaavoittuvuuksien nimeämiseen ja kategoriointiin käytettyjä tietokantoja. Haavoittuvuudet sekä hyväksikäyttömenetelmät on ilmoitettu näihin tietokantoihin. Tietokannat ylläpitävät haavoittuvuuskirjastojaan ja niihin voi kuka tahansa ilmoittaa uuden tietoturvahaavoittuvuuden sellaisen löydettyään. Tietokantojen ylläpitävät tahot arvioivat haavoittuvuuksia ja antavat niille referenssinumeron, mikäli uusi haavoittuvuus sen ansaitsee. CVE:tä ja CWE:tä ylläpitää Yhdysvaltojen hallituksen rahoittama turvallisuusalan yhtiö MITRE Corporation. CVE:n tietokanta on yleinen haavoittuvuuksien tietokanta. CWE:n tietokanta eroaa CVE:stä sen keskittyessä sovellushaavoittuvuuksiin.

Tietyissä haavoittuvuuksissa on käytetty haavoittuvuutta tarkemmin kuvaavia termejä. Cross-site scripting haavoittuvuuksia on erityyppisiä, kuten stored XSS, reflected XSS ja DOM-based XSS. Tässä tapauksessa haavoittuvuudet eroavat riittävästi toisistaan, minkä vuoksi niiden eritteleminen on luontevaa.

Tutkimuksen aineistosta on tietyissä tapauksissa yhdistelty haavoittuvuuksia. Näissä tilanteissa aineistossa saman tyyppinen haavoittuvuus on nimetty usealla eri tavalla. Näin on toimittu esimerkiksi tilanteissa, joissa verkkosovellus vuotaa tai paljastaa tarpeetonta tietoa itsestään.

Myös esimerkiksi puutteellisesti asetettuja HTTP-otsakkeita on raporteissa nimetty eri tavoin. Näissäkin tilanteissa on vaihtelua riippuen haavoittuvuuden vaikutuksesta verkkosovellukseen. Esimerkiksi verkkosovelluksen käyttämän istuntotunnisteen arvon salakuuntelemisen estämiseksi, on korjaussuosituksissa suositeltu HTTP Secure -vivun asettamista verkkosovellukseen. Vaikka haavoittuvuus on ollut puutteellinen otsakkeen asetus, on haavoittuvuus nimetty token value access via eavesdropping. Yleisesti puutteelliset HTTP-otsakeasetukset on sisällytetty haavoittuvuuksiin insufficient client protection ja missing HTTP security headers. Tällöin testauksessa on havaittu yleensä useampi puutteellinen asetus. [27] Muita tilanteita, joissa haavoittuvuuden yhdistelyä on tehty ovat olleet esimerkiksi päivittämättömät ja haavoittuvuuksia sisältävät verkkosovelluksen palvelimen komponentit.

Tietyissä tapauksissa haavoittuvuuksien nimeämisissä on käytetty myös verkkosovelluksessa käytettyä tekniikkaa ja siihen liittyvää haavoittuvuutta. Esimerkiksi verkkosovellus, joka käyttää GraphQLää voi sisältää GraphQL schema enumeration haavoittuvuuden. GraphQL on avoimen lähdekoodin tietokantakyselykieli. [28]

2.3.1 OWASP top 10 ja yleisimmät tietoturvahaavoittuvuudet

OWASP-projekti ylläpitää listaa vaarallisimmista haavoittuvuuskategorioista, jota kutsutaan OWASP Top 10 listaksi. Tästä listauksesta on muodostunut eräänlainen de facto listaus, jota hyödynnetään sekä verkkosovellusten kehittämisessä, että niiden tietoturvatestaamisessa. Sovellusten kehitystyössä saatetaan yhtenä vaatimuksena pitää OWASP top 10 listan haavoittuvuuksien estämistä. Myös testausta suorittavaa tahoa saatetaan pyytää varmentamaan, ettei top 10 haavoittuvuuksia esiinny sovelluksessa. OWASP top 10 haavoittuvuuskategoriat 2021 ovat taulukossa 2.

OWASP top 10 listaa päivitetään neljän vuoden välein, ja viimeisin listaus on vuodelta 2021. [29] Lista kuvaa tietoturvatutkijoiden näkemystä uhkaavimmista haavoittuvuuskategorioista verkkosovelluksille. Tämä tarkoittaa, että listan järjestys ei kuvaa yleisimmin havaittavia haavoittuvuuksia. Kuitenkin Top 10 listaa tehdessä on huomioitu haavoittuvuuksien esiintymisen yleisyys ja vakavuus. Kategorioiden järjestämisessä on siis käytetty esimerkiksi CWE-tietokantaa sekä CVSS-arvosteluasteikkoa.

Yksittäinen listauksen kategoria saattaa sisältää useita eri tietoturvahaavoittuvuuksia. Esimerkiksi listauksen haavoittuvuuskategoria Injection pitää sisällään useita eri injektiohaavoittuvuuksia.

Taulukko 2. *OWASP top 10 haavoittuvuuskategoriat 2021*

Kategoria	**Kategorian suomennos**
A:01:2021 Broken Access Control	Rikkinäinen pääsynhallinta
A:02:2021 Cryptographic Failures	Kryptografiasi virheet
A:03:2021 Injection	Injektiot
A:04:2021 Insecure Design	Turvaton suunnittelu
A:05:2021 Security Misconfiguration	Turvallisuuden konfiguraatiovirheet
A:06:2021 Vulnerable and Outdated Components	Haavoittuvat ja vanhentuneet komponentit
A:07:2021 Identification and Authentication Failures	Identitfioinnin ja autentikoinnin puutteet
A:08:2021 Software and Data Integrity Failures	Sovelluksen ja Datan eheyden puutteet
A:09:2021 Security Logging and Monitoring Failures	Turvallisuuslokien ja monitoroinnin puutteet
A:10:2021 Server-Side Request Forgery	Väärennös palvelimen puolella

CVE ylläpitää tietokantaa vuodesta 1999 alkaen tietoturvahaavoittuvuuksista, jotka keskittyvät käyttöjärjestelmiin, sovelluksiin ja laitteistoihin. Haavoittuvuudet ovat ilmoitettuja julkisia haavoittuvuuksia, joita organisaatiot sekä yksittäiset tietoturvatutkijat ovat voineet ilmoittaa. Tietokannassa on jaoteltuna yli 160 000 haavoittuvuutta 14 kategoriaan.

Yleisempiä CVE-tietokannan haavoittuvuuksia ovat code executive kategorian haavoittuvuudet. Näitä on vuoteen 2021 mennessä yli 24 % kaikista haavoittuvuuksista. [30] Esimerkki code executive haavoittuvuudesta on unrestricted file upload haavoittuvuus, jossa sovellukseen pystytään tallentamaan tiedostoja vaarallisesti. [31] Tämän tutkimuksen aineistossa on haavoittuvuudesta käytetty nimeä malicious file upload.

Vuonna 2008 Web Aplication Security Consortium (WASC) julkaisi tutkimuksen, jossa julkaistiin kahdeksan eri kyberturvaorganisaation tietoturvatestauksista löydettyjä tietoturvahaavoittuvuuksia. Kohteena oli yli 12 000 verkkosovellusta, joista löydettiin lähes 100 000 haavoittuvuutta. Tietoturvatestauksia suoritettiin kolmella eri metodilla:

1. White-box testaus
2. Black-box testaus
3. Automaattinen haavoittuvuusskannaus

WASC:n haavoittuvuuksista yleisimpiä haavoittuvuuksia olivat cross-site scripting haavoittuvuudet, erilaiset tietovuotohaavoittuvuudet ja SQL-injektiohaavoittuvuudet. [32]

Vuonna 2007 yhdysvaltalainen WhiteHat Security tietoturvaorganisaatio julkaisi tutkimuksen toteuttamiensa verkkosovellusten tietoturvatestauksien haavoittuvuuksista. Haavoittuvuudet keskittyivät verkkosovelluksiin, jotka eivät olleet avoimen lähdekoodin tuotteita. Artikkelissa WhiteHat Securityn testaamista verkkosovelluksista oli julkaistu tieto, kuinka todennäköisesti tietynkaltainen haavoittuvuus löytyy verkkosovelluksen tietoturvatestauksesta. Todennäköisimmin esiintyviä haavoittuvuuksia olivat cross-site scriptin haavoittuvuudet, joita esiintyi 73 % todennäköisyydellä tietoturvatestauksessa. Tietovuotohaavoittuvuus esiintyi 53 % todennäköisyydellä ja content spoofing -haavoittuvuus 24 % todennäköisyydellä. Samassa testauksessa oli mahdollista esiintyä useita eri haavoittuvuuksia. [33]

2.4 Verkkosovelluksen tietoturvatestaus

Tässä luvussa käsitellään tietoturvatestausta, sen tavoitteita ja erilaisia testaustyylejä. Luvussa käsitellään erityisesti grey-box-tyylistä tietoturvatestausta. Lisäksi luvussa käydään läpi, mitä asioita testataan ja millä työkaluilla.

2.4.1 Tietoturvatestauksen tarkoitus ja tavoite

Pressmanin ja Maximin mukaan tietoturvatestauksen tarkoituksena on varmistaa tietoturvahyökkäyksien varalta tietojärjestelmän suojaus. [34] Tämä voidaan ajatella myös tavoitteeksi löytää verkkosovelluksesta tietoturvaa vaarantavia virheitä ja uhkia, mielellään ennen kuin virhe on hyväksikäytettävissä. Toisin sanoen testauksella luodaan tietoa. Tätä tietoa pystytään hyödyntämään esimerkiksi päätöksenteossa. Sovellusta hallinnoiva taho pystyy tiedon avulla tekemään päätöksiä. Päätöksiä ovat esimerkiksi verkkosovelluksen tietoturvan parantaminen korjaamalla sitä tai luomalla uusia tietoturvakontrolleja.

Verkkosovelluksen tietoturvatestauksella tavoitellaan myös tietoa sen hetkisiä tietoturvatilaa. Parhaissa tapauksissa tietoturvatestaus antaa tietoa muun muassa seuraaviin kysymyksiin:

- Onko verkkosovellus turvallinen käyttää vai onko verkkosovellus vaarallinen tai ongelmallinen käyttää?
- Onko verkkosovelluksen kautta mahdollista hyväksikäyttää jotain, joka on ongelmallinen verkkosovelluksen tai sen liitännäisen palvelun kannalta?

Tietoturvatestauksen tarkoituksena on myös saada tietoa mahdollisista ongelmista ja haavoittuvuuksista turvallisesti. Tällöin testaan niin, että testauksen kohteeseen tai sidoksessa oleviin sovelluksiin ei kohdistuisi tarpeetonta haittaa. Turvallisessa tietoturvatestauksessa pyritään testaamaan sovelluksen testiversiota, jolloin jo tuotantokäytössä tai kehitysvaiheessa oleva versio ei kokisi tarpeetonta haittaa. Näin ollen esimerkiksi injektiohaavoittuvuuden varmistaminen tai palvelinestohyökkäyksen toimimisen varmistaminen ei aiheuta aitoja ongelmia tuotantoversioon. Turvalliseen testaamiseen kuuluu myös turvallisen testidatan käyttö. Testidatan ei tulisi sisältää esimerkiksi oikeita henkilötietoja vaan anonyymisoitua tai generoitua tietoa.

Verkkosovelluksen tietoturvatestaus on hyvä erottaa verkkosovelluksen tietoturva-auditoinnista. Tietoturva-auditointi eroaa testauksesta sovelluksen vaatimus- tai tarkastusluettelolla. Auditoinnissa keskitytään varmistamaan, että ennalta määritetyt vaatimukset tai tarkastettavat kohteet täyttyvät määritelmien mukaisesti. Sen sijaan tietoturvatestauk-

sessa keskitytään tarkastelemaan verkkosovellusta ja havaitsemaan mahdollisia tietoturvapuutteita tai -ongelmia kokeilevasti. Tietoturvatestauksessa varmistettaan toisinaan myös tietoturva-auditoinneissa määriteltyjä vaatimuksia, mutta testauksessa ei keskitytä vain määriteltyjen vaatimusten tarkastamiseen. Tietoturvatestauksista käytetään toisinaan myös nimitystä penetraatiotestaus.

2.4.2 White-box, grey-box ja black-box tietoturvatestaus

Tietoturvatestaustyylejä on erilaisia. Tyylit jaotellaan usein perustuen testaajan ennakkotietoon verkkopalvelusta. Whitaker ja Newman määrittelevät hyökkäävän tietoturvatestauksen kolmeen luokkaan, jotka ovat black-box, grey-box tai white-box testaus. [35] Määritelmä perustuu kuvaukseen, kuinka paljon testauksen suorittajalla on tietoa testattavasta kohteesta. Black-box kuvaa mustaa tietämätöntä laatikkoa ja white-box taas kohdetta, josta tiedetään kaikki mahdollinen. Grey-box testaus hyödyntää molempien testauksien käytäntöjä. Alla olevassa taulukossa on jaoteltuna tarkemmin testauksen tyylit:

Taulukko 3. *Tietoturvatestauksen kategoriat*

Tietoturvatestaus tyyli	**Tyypillinen lähtötilanne ennen tietoturvatestausta**
White-box testaus	Testattavasta verkkosovelluksesta tiedetään käytännössä kaikki mahdollinen ennen testausta. Mahdollista lisätietoa annetaan testaajille testauksen aikana. Esimerkiksi testaajille mahdollistetaan tietoa verkkosovelluksen lähdekoodista, tietoa lokitapahtumista.
Grey-box testaus	Testattavasta verkkosovelluksesta jotain lähtötietoja. Näitä ovat esimerkiksi tieto verkkosovelluksen toimintalogiikasta, käyttäjätunnukset verkkosovellukseen.
Black-box testaus	Testauksen kohteesta lähtötietoina minimalistisesti tietoja. Näitä ovat esimerkiksi testattavan kohteen nimi tai URL-osoite.

Tietoturvatestauksien mallista riippuen, tulokset vaihtelevat. On oletettavaa, että white-box testauksella, jossa testaajalla on enemmän lähtötietoa, pystytään samassa testausajassa black-box testauksen suhteen saamaan enemmän mahdollisia haavoittuvuuksia. [36]

Tässä tutkimuksessa tarkastellut tietoturvatestausraportit ovat lähtökohtaisesti luonteeltaan grey-box tyyppisiä testauksia. Kyberturvallisuusyrityksen tietoturvatestaajat ovat saaneet yleensä joitain lähtötietoja ennen testausta sekä mahdollisesti lisää tietoa kesken testauksen. Testaajien kouluttaminen järjestelmän täydelliseen ymmärtämiseen vie aikaa. White-box testaus lisää myös tietoturvatestauksen kohteen testauspinta-alaa. Toi-

saalta taas täysin black-box tyylinen testaus vaatii aikaa, joka taas ei aina sovellu organisaatioiden tarpeisiin. Grey-box tyylinen on tästä syystä yleensä kustannustehokas malli soveltaa tietoturvatestauksia.

2.4.3 Tietoturvatestauksen toteuttaja ja ajankohta

Verkkosovellusten tietoturvatestauksen suorittaja voi vaihdella. Kun puhutaan sisäisestä tietoturvatestauksesta, sovelluksen testauksen suorittaa organisaatio. Tällöin testauksen tekijä voi olla muun muassa organisaation määrittämä testaaja tai sovelluksen suunnittelussa mukana olleet henkilöt. Yleensä testaus toteutetaan osana laadunvarmistusta. Tällöin on kuitenkin vaarana, että testauksen suorittanut henkilö ei välttämättä huomioi ulkopuolisen hyökkääjän kokemuksella uhkia.

Sisäisen tietoturvatestauksen voi myös suorittaa seuraamalla ohjattua testausmanuaalia. OWASP tarjoaa penetraatiotestaukseen manuaalin, jonka avulla pystytään testaamaan yleisiä tietoturvauhkia. [37]

Whitakerin ja Newmanin mukaan ulkopuolisen tekemä penetraatiotestaus on paras keino testata järjestelmää ja suojautua oikeilta hyökkääjiltä. Ulkopuolinen eettisesti toimiva hakkeri on toimeksiannolla toimiva taho, joka pyrkii vaarantamaan tietojärjestelmää turvallisesti. [35] Ulkopuolista tietoturvatestaajan toimintaa ohjaavat usein rajoitteet. Esimerkiksi ulkoinen testaaja ei saa suorittaa palvelunestohyökkäyksiä ilman verkkosovelluksen omistajan lupaa ja testauksen ajankohdasta on ilmoitettava. Jotkin organisaatiot myös ylläpitävät ulkopuolisille tietoturvatestaajille kohdistettuja bug bounty -ohjelmia. Niissä tarkoituksena on sallia tietyin ehdoin vapaa tietoturvatestaaminen, josta maksetaan palkkioita perustuen haavoittuvuuksien hyödynnettävyyteen ja vakavuuteen. [38]

Türpen mukaan tietoturvatestauksen ajankohta tietojärjestelmän vaihtelee riippuen tilanteesta. Testausta voidaan suorittaa muun muassa:

- tietojärjestelmän kehitys- ja laadunvarmistusvaiheessa,
- operointivaiheessa tietojärjestelmän käyttäjän toimesta,
- ulkoisten kolmansien osapuolien toimesta esimerkiksi hallinnollisten tahojen, tutkijoiden tai median toimesta, tai
- rikollisien toimesta.

Testauksen tarve vaihtelee myös testaajan näkökulmasta. Esimerkiksi rikollisen tarve voi olla taloudellisen hyödyn saavuttaminen, kun taas tietoturvatutkija on saatettu palkata

antamaan puolueeton arviointi esimerkiksi oikeustapausta varten. [39] Tietoturvatestaaminen on usein myös luvanvaraista eikä toisen omaisuuden tietoturvatestaaminen ilman oikeuksia ole sallittua.

Tässä tutkimuksessa testauksien suorittajana on toiminut ulkopuolinen kyberturvallisuusorganisaatio. Tietoturvatestaukset on toteutettu tilauksesta ja niihin on sisältynyt rajoitteita. Testauksia on suoritettu verkkosovelluksen eri vaiheissa.

2.4.4 Miten verkkosovelluksia testataan

Alla olevassa taulukossa on kuvattuna OWASP Web Application Security Testing manuaalin mukainen jaottelu verkkosovelluksen testauksen osakokonaisuuksia: [37]

Taulukko 4. *Tietoturvatestauksen jakautuminen*

Testauksen osakokonaisuus	**Esimerkkihaavoittuvuus**
Informaation keräys	HTTP-otsakkeen sormenjälki
Konfiguroinnin ja käyttöönoton testaus	Vaarallinen HTTP metodi käytössä
Identiteetin testaus	Käyttäjäroolilla liian paljon oikeuksia
Autentikaation testaus	Alustavat tunnukset yhä voimassa
Autorisaation testaus	Käyttöoikeuksien korotus
Istunnonhallinnan testaus	Cross-site reguest forgery
Syötteenkäsittelyn testaus	Cross-site scripting
Virheiden käsittelyn testaus	Teknologia pinon paljastuminen virheen johdosta
Kryptografian testaus	Puutteelliset TLS/SSL-asetukset
Liiketoimintalogiikan testaus	Liiketoimintalogiikan ohittaminen
Asiakasohjelman testaus	Cross-origin resource sharing
Rajapinnan testaus	GraphQL haavoittuvuus

Tietoturvatestauksen suorittajasta sekä rajoituksista riippuen määräytyy mitä osa-alueita tietoturvatestauksessa suoritetaan. Esimerkiksi rajapinnan testaamista ei suoriteta, mikäli sovelluksessa ei sellaista ole. Tai mikäli sovelluksen testaus rajataan vain kehitystiimin toimintoihin, jolloin kolmannen osapuolen toimintoja ei testata.

Osana informaation keräystä verkkosovelluksesta on tunnistaa, miten sovellus toimii ja miten se on toteutettu. Informaation keräyksen päätarkoituksena ei ole saavuttaa pääsyä esimerkiksi kriittiseen henkilötietoon vaan kerätä ymmärrystä sovelluksesta. Esimerkiksi

HTTP-sormenjälkien vuotamisen avulla kerätään tietoa teknologiapinosta. HTTP-sormenjäljen vuotaminen on oman kaltaisensa haavoittuvuus. Sen avulla voi paljastua mahdollisesti lisää haavoittuvuuksia, kuten vanhentuneet ja päivitystä tarvitsevat verkkosovelluskomponentit tai -kirjastot. Informaation keräystä suoritetaan kaikissa testauksissa.

Konfiguroinnin ja käyttöönoton testaus pitää sisällään muun muassa verkkosovelluksen käyttämän teknologiapinon konfiguraatioiden testaamisen. Tarkoituksena on varmistaa, että konfiguraatiot on asetettu oikein. Esimerkiksi konfiguroinnin ja käyttöönoton testauksessa saatetaan varmistaa, etteivät web-palvelimen tiedostojen käyttöoikeudet ole liian vapaasti hyödynnettäviä.

Autentikaation testaamisessa pyritään ohittamaan verkkosovelluksen autentikaatio tai aiheuttamaan haittaa verkkosovellukselle autentikaatiovaiheessa. Autentikaation ohittaminen voi tapahtua yksinkertaisimmillaan esimerkiksi verkkosovellukseen kehitysvaiheessa jääneiden oletuskäyttäjätunnusten avulla. Autentikaation testaamiseen kuuluu myös esimerkiksi ”unohditko salasanasi”-toiminnon testaaminen sekä heikkojen salasanapolitiikkojen käytön testaus.

Autorisaation testauksen tarkoituksena on varmistaa, että verkkosovellusta pystyy käyttämään vain asetettujen oikeuksien mukaisesti. Esimerkiksi autentikoinnin ohittaminen tai toisena käyttäjänä esiintymisen estäminen testataan osana pääsynhallinnan testausta. Autorisaation haavoittuvuuksia ovat myös muun muassa käyttäjäoikeuksien korottamisen haavoittuvuus privilege escalation. [40]

Istunnonhallinnan testauksella on tarkoitus varmistua, että verkkosovelluksen käyttäjän istunto toimii halutulla tavalla. Tähän kuuluu muun muassa vanhentuneen istunnon uudelleenkäytön testaaminen sekä cross-site request forgeryn testaaminen. Puutteellinen istunnonhallinta voi johtua esimerkiksi huonosta uloskirjautumistoiminnallisuudesta tai puutteellisista HSTS-otsakkeista (HTTP strict transport security).

Kryptografian testaus pitää sisällään varmistuksen, että verkkosovellus käyttää turvallisia salattuja menetelmiä ja protokollia. Esimerkiksi vanhentuneen TLS 1.0 -version käyttämistä tulisi välttää, ottamalla käyttöön TLS 1.2 -versio. Kryptografioilla ominaisuuksilla suojataan esimerkiksi salattavaa tietoa, kuten käyttäjänimiä sekä istuntotunniste-tokeneita. Salattava tieto sovelluksen HTTP-pyynnöissä tulisi olla riittävän hyvin kryptattuna.

Verkkosovelluksen liiketoimintalogiikan suojauksen testaamisen tarkoituksena on varmistaa, että sovelluksen logiikka toimii halutulla tavalla. Esimerkiksi on hyvä varmentaa, ettei verkkokaupan käyttäjä pysty lisäämään ostoskoriinsa lisää tuotteita tilauksen maksamisen jälkeen. Sovellus pitää sisällään tietyn toimintalogiikan, jota käyttäjä suorittaa

seuraamalla häntä varten suunniteltua käyttäjäpolkua. Haitallinen käyttäjä voi pyrkiä rikkomaan tätä logiikkaa ja muun muassa poikkeamaan halutulta käyttäjäpolulta.

Rajapinnan testauksella pyritään varmentamaan, että verkkosovellukseen liittyvät rajapinnat toimivat oikein ja turvallisesti. Rajapinnan testauksessa pyritään kokeilemaan erilaisia rajapinnan kutsujen väärinkäyttöä. Muun muassa istuntotunnisteiden varastamiselta suojaudutaan rajapinnan testauksella.

2.4.5 Tietoturvatestauksen työkalut, automaattiset ja manuaaliset menetelmät

Verkkosovellusten tietoturvatestauksissa kohdistetaan hyökkäyksiä sekä manuaalisin menetelmin että automatisoiduilla tekniikoilla. Automatisoiduilla menetelmillä tarkoitetaan yleisesti työkaluja, jotka pyrkivät itsenäisesti etsimään verkkopalvelusta heikkouksia. Esimerkiksi erilaiset skannaustyökalut, kuten Nessus ja Retina ovat automaattisia tietoturvatyökaluja. Näiden avulla skannataan kohdejärjestelmiä mahdollisten verkkohaavoittuvuuksien varalta. [41]

Täysin automaattisten testaustyökalujen lisäksi on olemassa myös testaustyökaluja, jotka sisältävät sekä automaattisia että manuaalisia toimintoja. Tämän kaltaisia työkaluja ovat muun muassa Caido, OWASP:n Zed Attack Proxy (ZAP) ja BurpSuite. Nämä työkalut tehostavat testausta käymällä läpi erilaisia testausmenetelmiä, joita voidaan automatisoida. [42] [43] [44]

Esimerkiksi tietoturvatestausmenetelmä fuzzauksella syötetään sovelluksen tiettyyn syötekohteeseen useita erilaisia syötetyyppejä yksikkötestauksen kaltaisesti. Sen sijaan, että testaaja manuaalisesti kokeilisi erilaisia syötetyyppejä, säästää automatisoitu kokeilu aikaa. Fuzzausta voidaan hyödyntää esimerkiksi tiedostopolkujen arvaamiseen. [45]

Automatisoiduilla tietoturvatestaustyökaluilla on mahdollista syöttää tietoturvatestausta varten tarkoitettuja HTTP-pyyntöjä, joilla työkalu etsii mahdollisia hyökkäyspaikkoja. Työkalut kertovat myös mitä erilaisia konfiguraatiopuutteita sovellus sisältää. Yksi esimerkki on puutteellisesti asetetut HTTP-otsakkeet. Näitä otsakkeita hyödynnetään suojaamaan sovellusta tietovuodoilta ja ulkoisista lähteistä tulevilta vaarallisilta HTTP-pyynnöiltä.

Automatisoidut tietoturvatestaustyökalut eivät kuitenkaan ole täysin itsenäisesti toimivia ja saattavat ilmoittaa havainnoista, jotka eivät kuitenkaan ole oikeita hyödynnettäviä tietoturvahaavoittuvuuksia. Näitä kutsutaan vääriksi positiivisiksi (false-positive) tietoturvahavainnoiksi. Automaattiset tietoturvatestaukset eivät ole täysin aukottomia eivätkä myöskään testaa kattavasti kaikkia mahdollisia syötekohtia.

Verkkosovelluksen liiketoimintalogiikka rajoittaa automaattisten työkalujen hyödyntämistä. Kaikkia sovelluksia vasten ei voida testata samoja syötteitä ja sama pätee automaattisissa testaustyökaluissa. Manuaalinen tietoturvatestaus pitää sisällään kokeilevaa ja verkkosovelluksen liiketoimintalogiikkaan perustuvaa tietoturvatestausta.

Automaattityökalut ovat kuitenkin testaajalle erittäin oleellisia myös manuaalisen testauksen kannalta. Esimerkiksi automaattityökalu saattaa huomauttaa cross-site scripting haavoittuvuudesta, jolloin testaajan tehtäväksi jää haavoittuvuuden varmentaminen. Tällöin on myös mahdollista, että testaaja havaitsee uusia hyökkäysmahdollisuuksia, joissa verkkosovellus ei suojaa käyttäjiään tai dataansa riittävästi.

2.4.6 Testausprojektin vaiheet

Tietoturvatestauksien vaiheet riippuvat sen käytännöistä ja suorittajasta. On kuitenkin yleisiä vaiheita, joita testauksessa toteutetaan:

- Testauksen laajuuden määritteleminen
- Testauksen suoritus
- Tuloksien raportointi

OWASP:n testausmanuaali suosittelee testaamista kaikissa verkkosovelluksen kehityksen vaiheissa sekä kun verkkosovellus siirtyy ylläpitovaiheeseen. On kuitenkin huomioitava, että tietoturvatestaamisen laajuus tällöin vaihtelee riippuen verkkosovelluksen kehitysvaiheesta. [37]

Laajuuden määrittäminen

Tietoturvatestaukset alkavat yleensä testauksen laajuuden määrittelyllä. Tässä vaiheessa pyritään saamaan vastauksia muun muassa seuraaviin kysymyksiin:

- mitä kohdetta tietoturvatestauksessa testataan,
- mitä kohteen tyypillisimmät toiminnot ovat,
- mitkä rajoituksia tietoturvatestaus sisältää?

Esitiedoilla pyritään luomaan arvioita myös kuinka monimutkainen itse testauksen toteuttaminen tulisi olemaan. Testauksen laajuuteen vaikuttaa myös kustannusarvio testauksesta. Laajuuden määrittelyllä on tärkeä tehtävä varmistaa, että tietoturvatestaus toteutetaan riittävän kattavasti ja turvallisesti. Testauksen laajuuden määrittämisen vaiheessa luodaan testauksesta testaussuunnitelma. Testaussuunnitelma voi edetä esimerkiksi OWASP:n testausohjeistusta seuraamalla, valitsemalla suunnitelmaan sovelluksen ominaisuuksien mukaisia testauksen osakokonaisuuksia.

Testauksen suoritus

Testausvaiheessa tietoturvatestaajat toteuttavat tietoturvatestausta. Testausta suoritetaan seuraamalla testaussuunnitelmaa. Testaussuunnitelma perustuu testattavan kohteen laajuuteen. Testausvaiheessa tietoturvatestaajat pyrkivät tutkimaan verkkosovellusta ja löytämään sekä varmentamaan verkkosovelluksen mahdollisia tietoturvahavaintoja haavoittuvuuksiksi. Testausvaiheessa hyödynnetään yleensä automaattisia ja manuaalisia menetelmiä. Tietoturvatestaajat varmistavat testausvaiheessa myös, etteivät haavoittuvuudet ole vääriä positiivisten haavoittuvuuksia.

Testauksen raportointi

Kun tietoturvatestaus on suoritettu seuraa raportoinnin vaihe. Tässä vaiheessa testaajat kirjaavat haavoittuvuudet esitettävään muotoon. Raportoinnin tarkoituksena on ymmärrettävästi esitellä testauksella löydetyt haavoittuvuudet ja korostaa niihin liittyvät riskit. Hyvä raportti on ymmärrettävissä sekä johtotasolla että verkkosovellusta kehittäneille teknisille osaajille. OWASP:n testausohjeissa suositellaan testausraportin sisältävän kolme osuutta. Lausunnon, jossa on sanallisesti kuvattuna mitä on tehty ja mitä on löydetty mielellään niin että sen ymmärtäminen ei vaadi teknistä osaamistaitoa. Toiseksi testauksen raportissa tulisi esiintyä testaukseen liittyvät parametrit. Näitä ovat esimerkiksi testauksen kohde ja tavoite, testauksen rajoitukset sekä testauksen aikataulu. Kolmas raportin osuus on löydökset testauksesta. Siinä kuvataan mitä haavoittuvuuksia on testauksessa löytynyt sekä mitä korjaustoimenpiteitä ne vaativat. Myös haavoittuvuuksien vaikutukset verkkosovellukseen on hyvä kuvata. [37]

2.4.7 Jatkotestaukset

Toisinaan tietoturvatestauksissa on tarvetta jatkotestaukselle. Tällä tarkoitetaan korjaustarkastuksia, uusintatestauksia ja vuosittaisia testauksia. Korjaustarkastuksen tavoitteena on varmentaa, että verkkosovelluksesta ensimmäisellä kerralla havaitut tietoturvahaavoittuvuudet on korjattu riittävän hyvin. Tällöin testauksen laajuudeksi määrittyy vain edellisen testauksen haavoittuvuudet.

Uusintatestaus on verkkosovelluksen testaaminen uudelleen. Vanhaa verkkosovellusta on saatettu uudistaa aikaisemmasta tietoturvatestauksesta sen verran paljon, että on käytännöllisempää suorittaa verkkosovellukselle täysin uusi testaus. Testauksen laajuus usein määritellään uudelleen uusintatestauksessa.

Vuosittaiset testaukset samaan verkkosovellukseen seuraavat samoja käytänteitä kuin uusintatestaukset. Vuosittaiset testaukset toteutetaan noin vuoden intervallilla. Testauksen laajuus yleensä määritellään uudelleen.

3. TUTKIMUKSEN TOTEUTUS

Tässä luvussa käydään läpi diplomityön tutkimusmenetelmää sekä avataan, miten tutkimus on toteutettu. Luvussa 3.1. syvennytään tutkimuksen aiheeseen, aiheen rajaukseen ja kirjallisuuskatsaukseen. Luvussa 3.2. käydään läpi tutkimuksen tavoitteet. Luvussa 3.3. kerrotaan, miten tutkimus on toteutettu, sekä miten aineistoa on kerätty ja käsitelty.

3.1 Tutkimuksen aiheen rajaaminen ja kirjallisuuden kartoitus

Tutkimuksen aiheena on verkkosovellusten tietoturvahaavoittuvuudet ja niiden esiintyvyyden arviointi. Tietoturvallisuus on nykyaikainen puheenaihe ajassa, jossa ihmiset käyttävät entistä enemmän verkkosovelluksia. Tässä kontekstissa verkkosovellusten tietoturvahaavoittuvuudet ovat kiinnostava aihe.

Aihetta rajattiin käsittelemään vain verkkosovellusten tietoturvatestauksia. Muita mahdollisia tietoturvatestauskohteita ovat esimerkiksi mobiilisovellukset tai verkkoon yhdistettävät laitteet. Tässä tutkimuksessa rajauksen syynä oli tutkittavan aineiston määrä. Verkkosovellusten tietoturvatestauksia oli aineiston keräämistä varten riittävästi, jolloin keskittyminen vain niihin oli luontaista.

Tutkimus on mielenkiintoinen ainutlaatuisuutensa vuoksi. Diplomityötä varten etsittiin tieteellisiä artikkeleita, jotka käsittelivät tietoturvatestaamista ja testauksessa havaittuja tietoturvahaavoittuvuuksia. Kirjallisuudesta haettiin erityisesti tietoa siitä, mitä tyypillisiä haavoittuvuuksia tietoturvatestauksissa paljastuu. Useimmat artikkelit olivat kuitenkin yksittäisiä verkkosovelluksen testauksen tuloksista kertovia artikkeleita.

Muutamia verkkosovellusten tietoturvatestauksien haavoittuvuuksien määriä esittäviä artikkeleita löytyi. Näitä on esiteltynä enemmän luvussa 2.3.1. Suomessa toteutettuja vastaavan kaltaisia tutkimuksia ei tullut vastaan lainkaan. On kuitenkin huomioitava, että kaikkea tietoa ei välttämättä ole käyty läpi ja on olemassa mahdollisuus vastaavanlaiseen tutkimukseen. Kirjallisuuskatsausta toteutettiin Google Scholar ja Tampereen yliopiston Andor kirjallisuushakujen avuilla.

3.2 Tutkimuksen tavoitteet

Tutkimuksen tavoitteena oli kerätä tietoturvahaavoittuvuuksia verkkosovellusten tietoturvatestauksista sekä verrata niiden esiintymistä tietyissä verkkosovelluskategorioissa. Tutkimuksen tarkoituksena oli vertailla sovelluksia kompleksisuuden avulla. Kompleksisempi sovellus sisältää mahdollisesti enemmän vakavia tietoturvahaavoittuvuuksia. Hypoteesina on, että monimutkaisemman sovelluksen toteuttaminen vaatii enemmän huomioitavia ominaisuuksia. Tähän voi vaikuttaa esimerkiksi sovelluksen laajuus. Isompi sovellus saattaa vaatia enemmän tekijöitä, ja suurempi määrä muuttujia ei aina takaa parasta lopputulosta.

Toisena tavoitteena oli saada tietoa siitä, millaisia tietoturvahaavoittuvuuksia suomalaisten organisaatioiden käyttämissä verkkosovelluksissa esiintyy, ja onko verkkosovelluksen kompleksisuudella merkitystä esiintyvien haavoittuvuuksien tyyppeihin. Hypoteesina on, että kaikissa sovelluksissa esiintyy tasaisesti konfiguraatiohaavoittuvuuksia, mutta suunnittelu- ja ohjelmointihaavoittuvuuksia esiintyy enemmän kompleksisemmissa sovelluksissa. Sovelluksissa on kompleksisuudesta riippumatta samoja ominaisuuksia, jotka johtavat konfiguraatiohaavoittuvuuksiin. Kompleksisimmissa sovelluksissa on kuitenkin enemmän ja monimutkaisempia ominaisuuksia, joiden johdosta ohjelmointi- ja suunnitteluhaavoittuvuuksia esiintyy näissä oletettavasti enemmän.

Tutkimuskysymykset tässä tutkimuksessa olivat seuraavia:

- Onko verkkosovelluksen kompleksisuudella vaikutusta merkittävien tietoturvahaavoittuvuuksien lukumäärään?
- Mikä on yleisin tietoturvatestauksessa esiintyvä haavoittuvuustyyppi?
- Onko verkkosovelluksen kompleksisuudella vaikutusta haavoittuvuustyyppien esiintyvyyteen?
    - Esiintyykö konfiguraatiohaavoittuvuuksia, ohjelmointihaavoittuvuuksia ja suunnitteluhaavoittuvuuksia kaikissa verkkosovelluskategorioissa yhtä paljon suhteessa kaikkiin kategorian haavoittuvuuksiin?

Kysymyksiin vastattiin keräämällä aineistoa kohdeyrityksen verkkosovellusten tietoturvatestauksien raporteista ja analysoimalla näistä kerättyä tietoa.

3.3 Tutkimusmenetelmä ja tutkimuksen vaiheet

Tutkimustyö on tehty kvantitatiivisena tutkimuksena, jossa on käytetty tietoturvatestauksia tapauksina. Kvantitatiivisessa tutkimuksessa keskeistä on tutkimuksen aiheen määrittelyn avulla kerätä aineistoa [46]. Lähtökohtana aineiston analyysissä on aineiston mitattavuus. Tässä tutkimuksessa aineiston keräämisestä on kuvattuna ja aineiston analysointia luvussa 3.4.

Itse tutkimus koostui seuraavista vaiheista:

1. Ongelman määrittely ja suunnitteluvaihe,
2. Aineiston kerääminen,
3. Aineiston jalostaminen,
4. Aineiston analysointi,
5. Tulosten raportointi ja arviointi

Tutkimuksen ongelman määrittely- ja suunnitteluvaiheessa määriteltiin tutkimuskysymykset ja tavoitteet, sekä toteutettiin kirjallisuuskatsausta. Aineiston keräämisvaiheessa kerättiin tutkimuksen aineistoa tutkimusongelman ratkaisemisen tueksi. Tämän jälkeen aineistoa jalostettiin rajaamalla ulos tutkimukseen kuulumattomia tapauksia sekä yhdistämällä esimerkiksi verkkosovellushaavoittuvuuksia. Jalostamisen jälkeen aineistoa analysoitiin. Analysoinnissa etsittiin vastauksia tutkimuksen kysymyksiin ja pyrittiin vastaamaan tutkimuksen tavoitteisiin. Lopuksi tulosten arviointi- ja raportointivaiheessa suoritettiin tulosten siirtämistä taulukoihin. Tuloksia analysoitiin verraten tutkimuskysymyksiin.

3.4 Tietoturvatestausraportit ja raporttien sisältämien tietojen kerääminen

Tutkimustyössä kerättiin aineistoa kyberturvallisuusyrityksen toteuttamien tietoturvatestauksien raporteista vuosilta 2012–2021. Tietoturvatestausraportit on kirjoitettu sen hetkisistä tietoturvatestauksista. Raportin on kirjoittanut tietoturvatestauksen toteuttanut tekninen asiantuntija.

Raporttien läpikäynti oli ajallisesti tutkimuksen suurin osuus. Testausraportit etsittiin yrityksen raporttipankista ja niistä kirjattiin tiedot Excel taulukko-ohjelmaan manuaalisesti. Aineiston keräämisessä ei ollut juurikaan automatisointimahdollisuuksia, tai niiden kehittäminen ja hyödyntäminen olisi vienyt enemmän aikaa.

Erilaisia tietoturvatestausraportteja oli useita, joista tämän tutkimuksen aineistoon rajattiin pois muut kuin verkkosovellusten tietoturvatestauksen raportit. Näitä oli yhteensä 447, joista 76 oli uusintatestauksia, korjaustestauksia tai vuosittaisia tietoturvatestauksia

samalle kohteelle. Tietoturvatestausraporteista kerättiin aineistoa varten erilaisia tietoa. Tärkeimpiä olivat tietoturvahaavoittuvuustiedot sekä testauksen kohdetiedot.

3.4.1 Tietoturvatestausraporttien tietojen kerääminen

Tietoturvatestausraporteista on kerätty erilaisia tietoja kohdeyrityksen suorittamista tietoturvatestauksista. Tietoja kerättiin Excel taulukko-ohjelmaan kahteen päätauluun. Ensimmäiseen kerättiin tietoja tietoturvatestauksesta ja toiseen testauksien haavoittuvuuksista.

Tietoturvaraportit-tauluun kerättiin tietoja testauksen kohteesta ja tuloksista. Kohdetietoja olivat testauksen kohteen verkkosovelluskategoria, oliko testattava kohde jatkotestaus vai ensimmäinen testaus, sekä mitä käyttäjäryhmää varten verkkosovelluksen oletettu käyttötarkoitus on. Kohdetiedot on kerätty sovelluksista tutkijan parhaimman ymmärryksen avulla perustuen testauksien raportteihin. Tulostiedot testauksesta sisälsivät haavoittuvuuksien lukumäärät tyypeittäin, kriittisten haavoittuvuuksien määrät, sekä testauksen arvosanan.

Tietoturvahavainnot-tauluun kerättiin tietoja raporttien tietoturvahaavoittuvuuksista. Jokainen raporteissa esiintynyt haavoittuvuus ja sen testauksen aikana arvioitu kriittisyys merkittiin ylös. Joitakin haavoittuvuustyyppejä yhdisteltiin yhdeksi kategoriaksi, sillä testaajat olivat nimenneet samankaltaisia haavoittuvuuksia eri tavoin. Tämä johtuu testaajien eri tavoista nimetä raporttien tietoturvahavainnot. Osa tietoturvahaavoittuvuuksista on nimetty eri tavalla, vaikka itse tietoturvahaavoittuvuus on saman tyyppinen.

Esimerkiksi haavoittuvuustyypit data leak, sensitive data disclosure ja improper error handling pitävät sisällään saman havainnon eli palvelu vuotaa tarpeetonta tietoa itsestään. Näiden lisäksi raportteihin oli nimetty tietoturvahavaintoja, jotka ovat täysin verkkopalvelutyyppikohtaisia. Esimerkiksi IBAN harvesting, joka vaatii verkkopalvelulta IBAN numeroiden käyttöä jossain palvelun vaiheessa. Näitäkin havaintoja yhdistettiin jälkikäteen.

Vaikeita ongelmia tietojen keräämisessä ei esiintynyt juurikaan. Yleisesti tietojen kerääminen oli manuaalisena työnä aikaa vievää. Myös yksittäiset haavoittuvuudet, jotka eivät suoraan sopineet jo olemassa oleviin haavoittuvuuksiin tuottivat pientä vaivaa. Kokonaisuudessaan haavoittuvuuksien ja raportin tietojen keräämistä helpotti yhdenmukaiset raportointipohjat. Keräämistyötä olisi helpottanut, mikäli tiedot haavoittuvuuksista olisi ollut etukäteen saatavissa helpommin esimerkiksi tietokannasta.

3.4.2 Tietoturvatestausraporttien kategorisointi

Tietoturvatestausraporttien kohteiden kategorisointityö on toteutettu manuaalisesti käymällä läpi tietoturvatestausraportteja. Raporttien tiedoista on päätelty mihin kategoriaan testauksen kohde on kuulunut. Tässä tutkimuksessa on seurattu kategorisointia, joka perustuu verkkosovellusten kompleksisuuteen [6]. Kategoriat esimerkkeineen on esitelty Taulukossa 5.

Taulukko 5. *Tutkimuksen verkkosovelluskategoriat*

Kategoria	**Esimerkkiverkkosovellus**
Informatiivinen	Staattinen verkkosivu, Organisaation www sivu
Interaktiivinen	Verkkolomakkeet, Verkkopelit, Laskintyökalut
Kaupallinen / vaihdollinen	Verkkokauppa, Verkkopankki, Varaussivusto
Työjonoon perustuva	Toiminnanohjausjärjestelmät (ERP), Asiakkuudenhallintajärjestelmät (CRM), Aikataulutus ja suunnittelusovellukset, Monitorointisovellukset
Yhteisöllinen	Chat-sovellukset, Sähköiset oppimispalvelut, Jaetut työkalut, Tiedostojen jakopalvelu
Muut	Sovelluksia mahdollistavat ominaisuudet, Usean sovelluksen API-rajapinnat, E-allekirjoitusominaisuus

Tutkimuksen kategorisoinniksi valittu malli soveltui aineistoa varten hyvin. Kategorisoinnin malli oli riittävän selkeä, jotta sen avulla verkkosovelluksia pystyttiin jakamaan riittävän selkeisiin kategorioihin. Kategoriointia on myös muokattu alkuperäisestä lisäämällä yksi uusi kategoria. Kategoriaan ”Muut” on sisällytetty lähinnä muita verkkosovelluksia mahdollistavia sovelluksia. Näihin kuuluivat muun muassa erilaiset API:t ja e-allekirjoitusominaisuudet.

Tutkimukseen valittu kategorisointi ei kuitenkaan ole täydellinen vaan enemmänkin paras yritys. Tämä johtuu muun muassa testattujen verkkosovellusten moniulotteisuudesta. Verkkosovellukset voivat sisältää useita yllä olevan kategorisoinnin mukaisia ominaisuuksia. Kategorisoinnissa huomioitiin verkkosovelluksen testauksessa painotettuja

ominaisuuksia sekä määriteltiin verkkosovellus sen mukaan, mitä se tutkijan parhaan arvion perusteella eniten edustaa.

Esimerkiksi kuvitteellinen verkkosovellus, joka on organisaation sisäiseen käyttöön tarkoitettu web-portaali, voi sisältää organisaation uutissivuston. Uutissivuston johdosta kategoriaksi voisi valikoitua informatiivisen verkkosovellus. Tässä tutkimuksessa se olisi kuitenkin luokiteltu yhteisöllisiin verkkosovelluksiin. Mikäli tietoturvatestaus olisi keskittynyt portaalissa vain uutissivustoon, olisi se kategorisoitu informatiiviseksi verkkosovellukseksi.

Kategorisointi on toteutettu perustuen verkkosovelluksen tietoturvatestausraporttiin, jossa tietoturvatestaaja on kuvannut testattavaa kohdetta sanallisesti. Mahdollisuuksien mukaan kategorisoinnissa on pyritty selvittämään Internet-hauilla mahdollisen kohteen käyttötarkoitusta etenkin tilanteissa, joissa tietoturvatestausraportti ei ole selkeästi kuvannut verkkosovelluksen käyttötarkoitusta.

Verkkosovelluskategorioiden kompleksisuuden järjestys on määritelty seuraavasti.

1. Informatiiviset ja Muut-kategoria
2. Interaktiiviset
3. Transaktionaaliset
4. Yhteisölliset ja Työjonoon perustuvat

Informatiiviset verkkosovellukset ja Muut-kategorian verkkosovellukset ovat vähiten kompleksisia. Näissä verkkosovelluksissa toiminnallisuuksia on melko vähän. Esimerkiksi organisaatioiden www-sivut eivät välttämättä sisällä muuta kuin staattisia ominaisuuksia. Seuraavaksi vähiten monimutkaisempia ovat interaktiiviset verkkosovellukset. Interaktiiviset verkkosovellukset sisältävät informatiivisia enemmän toiminnallisuuksia, mutta kuitenkin rajoitetusti.

Interaktiivisia verkkosovelluksia monimutkaisempia ovat transaktionaaliset verkkosovellukset. Transaktionaalisissa sovelluksissa on tyypillisesti useita toiminnallisuuksia liittyen käyttäjään ja hänen tietoihinsa. Esimerkiksi verkkovaraussivusto kerää ja tallentaa sekä käyttäjätietoja, että tilatietoja varattavista kohteista. Kompleksisimpia sovelluksia ovat yhteisölliset ja työjonoon perustuvat sovellukset. Näissä sovelluksissa on useita toiminnallisuuksia, jotka ovat riippuvaisia toisistaan. Hyvä esimerkki on toiminnanohjausjärjestelmä. Siinä monet sovelluksen toiminnallisuudet ovat sidoksissa keskenään ja riippuvia toisten toiminnollisuuksien tiloista ja arvoista.

4. TUTKIMUSTULOKSET

Tässä luvussa käydään läpi tutkimuksen tuloksia. Tutkimuksen tuloksissa on nostettuna esille tietoturvatestauksien raporttien sisältöä ja mitä johtopäätöksiä niiden perusteella voidaan tehdä. Luvussa pohditaan myös tutkimustuloksia ja niiden syitä.

4.1 Tutkimustuloksia

Tutkimuksen tarkoituksena oli selvittää kuinka erilaiset verkkosovelluskategoriat vertautuvat keskenään tietoturvatestauksista paljastuneiden haavoittuvuuksien kautta. Tutkimuksessa selvitettiin mitä haavoittuvuuksia testausten aikana löytyi ja mitkä haavoittuvuudet olivat tyypillisimpiä erilaisissa sovelluksissa. Tutkimuksen tuloksia varten aineistoa kerättiin ja analysointiin manuaalisesti. Liitteessä 1 on tutkimuksen haavoittuvuudet, haavoittuvuuksien määrät sekä kriittisten ja korkean vaikutuksen haavoittuvuuksien määrät.

4.1.1 Tutkimuksen taulukot ja yleistä tietoa tutkimuksesta

Tässä tutkimuksessa tarkasteltujen tietoturvatestausten kokonaismäärä oli 447. Tietoturvatestausraportteja on kerätty vuosilta 2012–2021, ja niitä oli yksi kappale jokaista testausta vasten. Tietoturvatestausten raporteista kerättiin testauksen kohteeseen ja testaustuloksiin liittyviä tietoja.

Kerättyjä kohdetietoja testauksen kohteesta olivat seuraavat:

- Tietoturvatestauksen tyyppi
- Testatun kohteen käyttäjäryhmä
- Kohteen verkkosovelluskategoria

Tietoturvatestauksen tyyppi kuvaa sitä, millainen tietoturvatestaus oli kyseessä. Testaus oli tyypiltään joko tietoturvatestaus tai jatkotestaus. Jatkotestaukset jaettiin edelleen kolmeen tyyppiin, jotka olivat korjaustarkastus, uusintatestaus, ja vuosittainen testaus. Tietoturvatestauksen kohteen käyttäjäryhmät jaoteltiin tutkimuksessa kolmeen ryhmään. Nämä olivat sisäiset käyttäjät, yritysasiakkaat ja henkilöasiakkaat. Verkkosovelluskategorioihin jaottelu on kuvattuna luvussa 3.4.2.

Tietoturvatestauksista kerättyjä tuloksiin liittyviä tietoja olivat seuraavat:

- Kohteen arvosana perustuen tietoturvatestaukseen
- Kriittisen ja korkean vaikutuksen haavoittuvuuksien määrät
- Konfiguraatiohaavoittuvuudet
- Ohjelmointihaavoittuvuudet
- Suunnitteluhaavoittuvuudet
- Haavoittuvuudet yhteensä

Tietoturvatestauksen arvosana perustuu viisiportaiseen asteikkoon (1–5). Korkein arvosana (5) kuvaa turvalliseksi arvioitua verkkosovellusta ja matalin arvosana (1) kuvaa haavoittuvaista ja epäturvallista verkkosovellusta. Arvosanan testauksesta on määritellyt testauksen suorittanut tietoturvatestaaja testauksen ajankohtana. Arvosanaan ovat vaikuttaneet testauksen haavoittuvuuksien määrät sekä haavoittuvuuksien vaikutukset sovelluksen tietoturvaan. Mahdollisessa tilanteessa, jossa testauksessa oli suuri määrä matalan vaikutuksen haavoittuvuuksia ei välttämättä johtanut erittäin haavoittuvaisen sovelluksen arvosanaan.

Verkkosovelluksen kriittisen ja korkean vaikutuksen haavoittuvuuksien määrä kuvaa sitä, kuinka monta haavoittuvuutta testaaja on määritellyt verkkosovellukseen kriittiseksi tai korkean vaikutuksen havainnoksi. Konfiguraatio-, ohjelmointi- ja suunnitteluhaavoittuvuuksien määrät kuvaavat sitä, kuinka monta kuhunkin kategoriaan luokiteltua haavoittuvuutta raporttiin on kirjattu. Haavoittuvuuksien yhteismäärä kuvaa koko tietoturvatestauksessa raportoitujen haavoittuvuuksien määrän. Kaikki haavoittuvuudet olivat jaoteltuna joko konfiguraatio-, ohjelmointi- tai suunnitteluhaavoittuvuudeksi.

Tietoturvatestauksen kohteista kerättyjen tietojen lisäksi kerättiin tietoa tietoturvahaavoittuvuuksista. Tietoturvahaavoittuvuuksista kerättyjä tietoja olivat tietoturvatestauksen haavoittuvuuden nimi, sekä oliko haavoittuvuus kriittinen tai korkean vaikutuksen omaava.

4.1.2 Yleisiä tutkimustuloksia

Tutkimuksen otanta oli yhteensä 447 verkkosovelluksen tietoturvatestausta. Tietoturvatestauksen tyyppien lukumäärät ja prosenttiosuus kaikista raporteista on esitetty taulukossa 6. Suurin osa testauksista oli ensimmäisen kerran tietoturvatestauksia. Jatkotestauksien osuus aineistosta on 17,7 %.

Taulukko 6. *Tietoturvaraporttien määrät ja keskiarvot*

Yksikkö	Lukumäärä	%-osuus
1. testauskertoja	368	82,3 %
Jatkotestauksia	79	17,7 %
Tietoturvatestausraportteja	447	100 %

Kaikista tietoturvatestauksista jatkotestauksia oli 79 kappaletta ja niiden jakautuminen korjaustarkastuksiin, uusintatestauksiin, ja vuosittaisiin testauksiin on esitetty taulukossa 7. Valtaosa jatkotestauksista oli korjaustarkastuksia.

Taulukko 7. *Jatkotestauksien määrät ja keskiarvot*

Yksikkö	Lukumäärä	%-osuus kaikista testauksista
Korjaustarkastuksia	59	13,2 %
Uusintatestauksia	13	2,9 %
Vuosittaisia testauksia	7	1,6 %

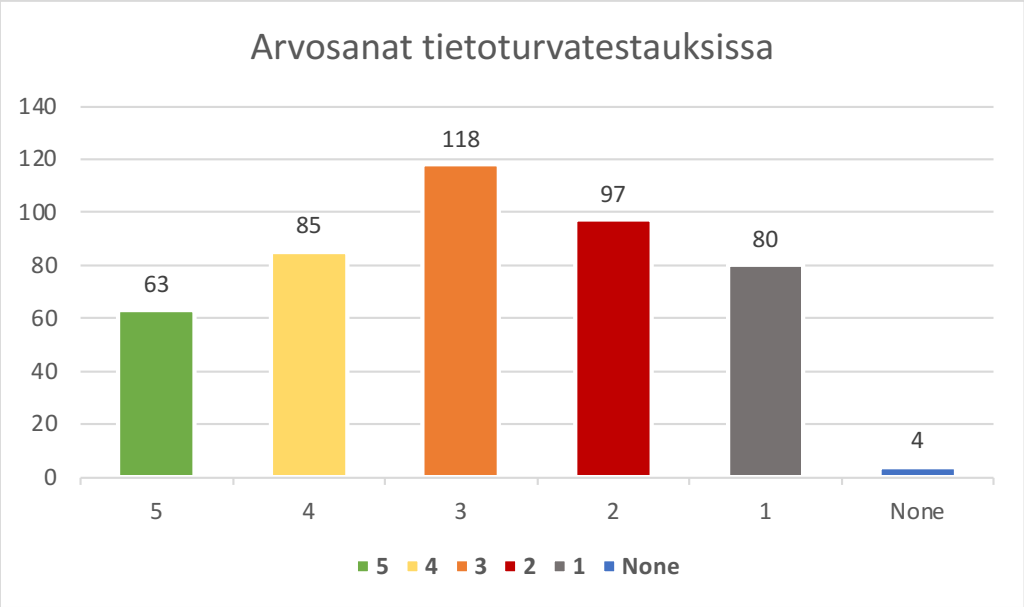
Tietoturvatestausten kohteena olleita verkkosovelluksia luokiteltiin myös niiden kohdekäyttäjäryhmien mukaan. Käyttäjäryhmiksi määritettiin henkilöasiakkaat, yritysasiakkaat sekä organisaation sisäiset käyttäjät. Raporttien määrät käyttäjäryhmittäin on esitetty taulukossa 8.

Taulukko 8. *Tietoturvaraporttien määrät verkkosovelluksen käyttäjäryhmittäin*

Verkkosovelluksen käyttäjäryhmä	Lukumäärä	%-osuus
Henkilöasiakkaat	164	36,7 %
Yritysasiakkaat	159	35,6 %
Sisäinen	124	27,8 %

Suurin osa tietoturvatestauksista oli henkilöasiakkaiden käyttöön kehitettyjä verkkosovelluksia. Testaukset käyttäjäryhmien välillä jakautuivat tuloksissa kuitenkin melko tasaisesti kaikkien kolmen kesken. Tilaajaorganisaatioiden sisäiseen käyttöön tarkoitettuja verkkosovelluksia oli testissä vähiten.

Verkkosovellusten tietoturvatestauksien arvosanat seurasivat viisiportaista arvosana-asteikkoa. Korkein arvosana (5) kuvasi turvallista verkkosovellusta ja matalin arvosana (1) vähinten turvallisinta verkkosovellusta. Arvosanojen jakautuminen on esitetty kuvassa 3.



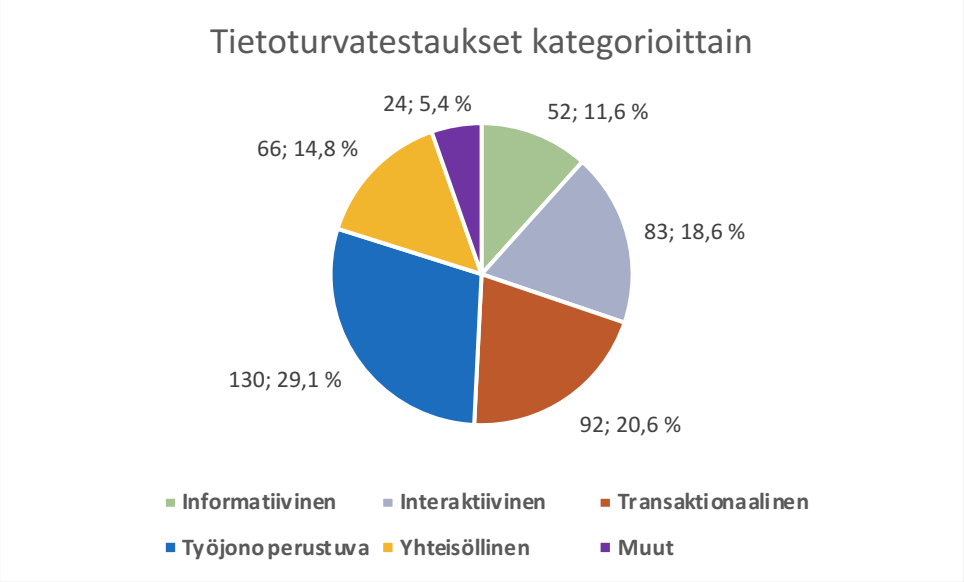
Kuva 3. *Verkkosovellusten tietoturvatestauksien arvosanat*

Aineiston keskiarvoinen verkkosovelluksen arvosana on 2,89. Arvosanat jakautuivat epätasaisesti. Arvosanan kolme saaneita oli eniten ja arvosanan 2 seuraavaksi eniten. Tämä tarkoittaa, että tietoturvatestauksen kohteiden turvallisuuden taso oli testaajien näkökulmasta keskimääräisesti melko heikko.

Neljä tietoturvatestausta ei sisältänyt tietoturvatestauksen kokonaisarvosanaa lainkaan. On myös huomioitava, että kaikissa tietoturvatestauksissa ei ole ollut kyse koko verkkosovelluksen tietoturvatestaamisesta vaan tietoturvatestaus on voinut keskittyä tiettyyn verkkosovelluksen osaan. Esimerkiksi verkkosovelluksen rajapinnan testaamisen tapauksissa ei ole huomioitu koko verkkosovellusta vaan pelkästään rajapinnan toteutus.

4.1.3 Tutkimustuloksia verkkosovelluskategorioittain

Tutkimuksessa käytettiin kuutta verkkosovelluskategoriaa. Tietoturvatestaukset verkkosovelluskategorioittain jakautuivat kuvan 4 mukaisesti.



Kuva 4. *Kaavio tietoturvatestauksien jakautumisesta verkkosovelluskategorioittain*

Työnjonoon perustuvia tietoturvatestauksia oli 130 kappaletta ja myös eniten kaikista verkkosovelluskategorioista. Yleisesti ottaen verkkosovellusten testausten määriä oli hyvä edustus, pois lukien Muut-kategorian testaukset. Haavoittuvuudet sovelluskategorioittain jakautuivat taulukon 9 mukaisesti.

Taulukko 9. *Tietoturvatestauksien haavoittuvuudet verkkosovelluskategorioittain*

Kategoria	Haavoittuvuuksien määrä	Testauksien määrä	Haavoittuvuudet / Testaus
Informatiivinen	264	52	5,1
Interaktiivinen	555	83	6,7
Transaktionaalinen	598	92	6,5
Työjonoon perustuva	868	130	6,7
Yhteisöllinen	390	66	5,9
Muut	127	24	5,3
Yhteensä	2802	447	6,3

Suurin osa aineiston haavoittuvuuksista löytyi työjonoon perustuvista verkkosovelluksista. Haavoittuvuuksien jakautuminen tietoturvatestausta kohden oli melko yhteneväinen. Eroa oli 1,6 haavoittuvuutta suurimman ja pienimmän verkkosovelluskategorian välillä. Haavoittuvuuksia esiintyi keskimääräisesti vähinten informatiivisissa (5,1 kpl) ja eniten työjonoon perustuvissa (6,7 kpl) verkkosovelluksissa.

Jokaisessa verkkosovelluskategoriassa esiintyi kriittisen ja korkean vaikutuksen tietoturvahaavoittuvuuksia. Verkkosovelluskategorioittain tietoturvatestauksien kriittisten ja korkean vaikutuksen haavoittuvuudet jakautuivat taulukon 10 mukaisesti.

Taulukko 10. *Tietoturvatestausten kriittiset ja korkean vaikutuksen haavoittuvuudet verkkosovelluskategorioittain*

Kategoria	**Lukumäärä**	**Testauksia**	**Kriittiset ja korkean vaikutuksen havainnot / testaus**
Informatiivinen	23	52	0,4
Interaktiivinen	79	83	1,0
Transaktionaalinen	68	92	0,7
Työjonoon perustuva	138	130	1,1
Yhteisöllinen	70	66	1,1
Muut	11	24	0,5
Yhteensä	389	447	0,9

Suurin osa kriittisistä haavoittuvuuksista oli työjonoon perustuvissa sovelluksissa, joissa niitä esiintyi 138 kappaletta. Myös havaintojen suhde testauksiin oli suurinta työjonoon suuntautuvissa verkkosovelluksissa sekä yhteisöllisissä sovelluksissa. Molemmissa esiintyi 1,1 kriittistä havaintoa yhtä testauskertaa kohden. Informatiivisissa ja muiden kategorioiden sovelluksissa esiintyi kriittisen vaikutuksen haavoittuvuuksia vain noin joka toisessa testauksessa.

Verkkosovelluskategorioittain kriittisiä haavoittuvuuksia sisältäviä testaustapauksia esiintyi eniten työjonoon perustuvissa sovelluksissa. Yhteensä 61 kappaletta kaikista 183 testauksesta, jotka sisälsivät edes yhden kriittisen haavoittuvuuden, kuului työjonoon perustuvaan verkkosovelluskategoriaan. Taulukossa 11 on esitettynä kriittisiä ja korkeita haavoittuvuuksia sisältäneiden testauksien jakautumisen sovelluskategorioittain.

Taulukko 11. *Kriittisiä ja korkean vaikutuksen haavoittuvuuksia sisältävät testaukset verkkosovelluskategorioittain*

Kategoria	**Kriittisiä haavoittuvuuksia sisältävät testaukset**	**Raportteja yhteensä**	**%-osuus kategoriassa**
Informatiivinen	17	52	32,7 %
Interaktiivinen	29	83	34,9 %
Transaktionaalinen	39	92	42,4 %
Työjonoon perustuva	61	130	46,9 %
Yhteisöllinen	29	66	43,9 %
Muut	8	24	33,3 %
Yhteensä	183	447	40,9 %

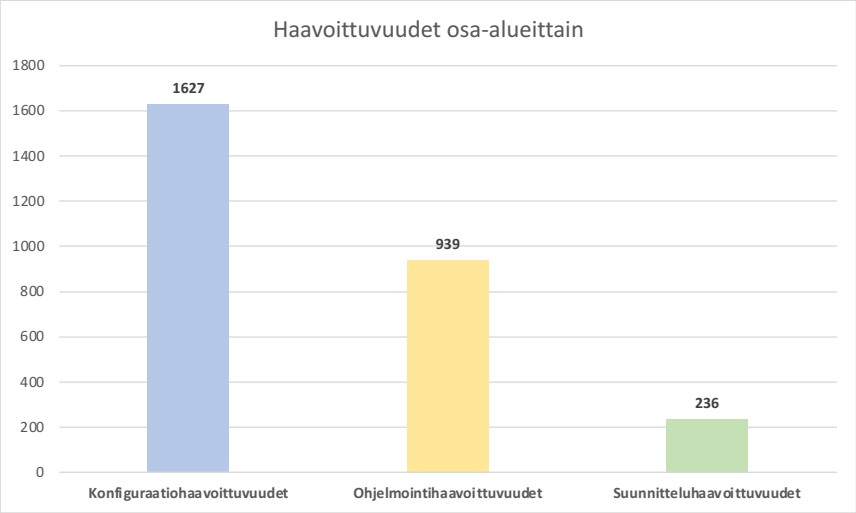
Kun vertaillaan prosenttiosuuksia kategorioittain työjonoon perustuvien ja yhteisöllisien verkkosovellusten testauksissa esiintyi kriittisiä tai korkean vaikutuksen haavoittuvuuksia keskiarvoon verrattuna enemmän. Keskiarvo kaikista testauksista oli 40,9 % kun taas työjonoon perustuvien ja yhteisöllisten sovellusten kriittisten ja vakavienhaavoittuvuuksien osuudet olivat 46,9 % ja 43,9 %. Vähemmän kompleksisissa verkkosovelluskategorioissa lukemat olivat pienemmät. Informatiivisten verkkosovellusten testaukset sisälsivät korkean ja kriittisen vaikutuksen haavoittuvuuksia 32,7 % testaustapauksissa ja Muut-kategoriassa vastaava luku oli 33,3 %.

4.1.4 Tietoturvahaavoittuvuudet osa-alueittain

Kaikkinensa tietoturvatestauksissa esiintyi yhteensä 2802 tietoturvahaavoittuvuutta. Nämä olivat tutkimuksessa jaoteltuna kolmeen yläluokkaan:

- konfiguraatiohaavoittuvuudet,
- ohjelmointihaavoittuvuudet ja
- suunnitteluhaavoittuvuudet.

Haavoittuvuuksien tiedot on kerätty tietoturvaraporteista niiden nimien perusteella ja yhdistelty muutamissa tapauksissa. Samaa haavoittuvuutta on voinut esiintyä useassa eri kohtaa verkkosovellusta, jolloin haavoittuvuuden vertailu lukumäärällisesti ei ole täysin samanarvoista. Luvussa 2.2.1 avataan haavoittuvuuksien luokittelusta enemmän.



Kuva 5. *Kaavio haavoittuvuuksien jakautumisesta osa-alueittain*

Kuvassa 5 on esitettynä haavoittuvuuksien jakautuminen osa-alueittain. Suurin osa aineiston haavoittuvuuksista oli konfiguraatiohaavoittuvuuksia. Konfiguraatiohaavoittuvuuksia oli yli puolet kaikista aineiston haavoittuvuuksista. Suunnitteluhaavoittuvuuksia esiintyi 236 kappaletta ja ohjelmointihaavoittuvuuksia 939 kappaletta.

Kun vertaillaan haavoittuvuuksien osa-alueiden esiintymistä verkkosovelluskategorioittain seuraavat tulokset yleistä linjaa. Kaikista verkkosovelluskategorioista konfiguraatiohaavoittuvuudet olivat yleisimpiä haavoittuvuuksia.

Taulukko 12 esittää prosenttiluvut haavoittuvuustyylien jakautumisesta verkkosovelluskategorioittain, kun huomioidaan kaikki haavoittuvuudet. Aineistossa haavoittuvuuksien määrät suhteutettuna kaikkiin haavoittuvuuksiin kuvaa kuinka konfiguraatiohaavoittuvuudet esiintyivät myös eniten suhteessa ohjelmointi- ja suunnitteluhaavoittuvuuksiin.

Taulukko 12. *Haavoittuvuustyylien jakauma verkkosovelluskategorioittain*

Kategoria	Konfiguraatio-haavoittuvuudet	Ohjelmointihaa-voittuvuudet	Suunnitteluhaavoittu-vuudet	Haavoittuvuuksia yhteensä
Informatiivinen	63 %	29 %	8 %	264
Interaktiivinen	54 %	35 %	10 %	555
Transaktionaalinen	59 %	34 %	7 %	598
Työjono suuntautuva	57 %	34 %	9 %	868
Yhteisöllinen	56 %	36 %	8 %	390
Muut	70 %	24 %	6 %	127
Yhteensä	58,1 %	33,5 %	8,4 %	100,0 %

Kuitenkin suhdeluvuissa on eroavaisuuksia. Informatiivisissa ja Muut-kategorioissa suunnittelu- ja ohjelmointihaavoittuvuuksien esiintyminen on pienempää kuin muissa kategorioissa. Suunnittelu- ja ohjelmointihaavoittuvuuksia esiintyy yhteenlaskettuna eniten interaktiivisissa ja yhteisöllisissä verkkosovelluskategorioissa.

Taulukossa 13 on sovelluskategorioittain määrät, kuinka eri osa-alueittain haavoittuvuuksia esiintyi keskimääräisesti yhtä testausta kohden.

Taulukko 13. *Haavoittuvuuksien esiintymisen keskimäärät testausta kohden verkkosovelluskategorioittain*

Kategoria	Konfiguraatio-haavoittuvuudet	Ohjelmointihaa-voittuvuudet	Suunnitteluhaavoittu-vuudet
Informatiivinen	3,2	1,5	0,4
Interaktiivinen	3,6	2,4	0,7
Transaktionaalinen	3,8	2,2	0,5
Työjonoon perustuva	3,8	2,2	0,6
Yhteisöllinen	3,3	2,1	0,5
Muut	3,7	1,3	0,3
Yhteensä	3,6	2,1	0,5

Informatiivisen verkkosovelluskategorian tietoturvatestauksessa esiintyy keskimäärin 1,5 kappaletta ohjelmointihaavoittuvuuksia ja 0,4 kappaletta suunnitteluhaavoittuvuuksia.

Taulukosta 13 on nähtävissä, että konfiguraatiohaavoittuvuuksia esiintyy melko tasaisesti kaikissa verkkosovelluskategorioissa, mutta tietyissä kategorioissa ohjelmointi- ja suunnitteluhaavoittuvuuksien määrät kasvavat. Tuloksista erikoista on, että toisin kuin oletettua, interaktiivisissa sovelluksissa esiintyy keskimääräisesti enemmän ohjelmointi- ja suunnittelu haavoittuvuuksia kuin kompleksisimmissa verkkosovelluksissa.

4.1.5 Yleisimmät tietoturvahaavoittuvuudet

Tutkimuksen aineiston haavoittuvuuksista yleisin liittyi datan paljastumiseen. Datan paljastumisen haavoittuvuuksia, joko suoraan tai virheiden kautta, esiintyi yhteensä 236 kertaa. Datan paljastumisen jälkeen yleisimpiä haavoittuvuuksia olivat SSL & TLS-haavoittuvuudet sekä puutteelliset HTTP turvallisuusotsakkeet. Taulukossa 14 on aineiston kymmenen yleisintä haavoittuvuutta.

Taulukko 14. *Tietoturvatestauksien kymmenen yleisintä haavoittuvuutta*

Sija	Haavoittuvuus	Haavoittuvuuksien lukumäärä	Tietoturvatestausta kohden
1.	Virheidenkäsittelyongelmat / Sensitiivisen datan paljastuminen / Datavuoto	236	0,53
2.	Riittämätön kuljetustason suojaus / SSL & TLS-haavoittuvuudet	197	0,44
3.	Puutteelliset HTTP turvallisuusotsakkeet / Riittämätön Asiakasohjelmiston suojaus	171	0,38
4.	Stored XSS	124	0,28
5.	HTTP otsakkeen sormenjäljen vuotaminen	122	0,27
6.	Reflected XSS	117	0,26
7.	Tokenin arvo salakuuntelun avulla / Puuttuva HTTP secure vipu	104,5	0,23
8.	Haavoittuvuuksia sisältävät komponentit / Vanhentuneet komponentit	102	0,23
9.	Username enumeration / harvesting	99	0,22
10.	Cross-Site Request Forgery	89	0,20

Tuloksista on nähtävissä, että yleisimmät haavoittuvuudet olivat tyypillisesti konfiguraatiohaavoittuvuuksia. Tyypillisesti ohjelmointihaavoittuvuuksiksi määritellyistä haavoittuvuuksista yleisimpiä olivat cross-site scripting (XSS) haavoittuvuudet stored XSS (124 kpl) ja reflected XSS (117 kpl). Myös cross-site request forgery (89 kpl) haavoittuvuudet olivat yleisiä.

4.1.6 Kriittiset ja korkean vaikutuksen haavoittuvuudet

Kriittisiä ja korkean vaikutuksen haavoittuvuuksia oli aineistossa 389 kappaletta. Kaikista haavoittuvuuksista nämä ovat 13,9 %. Näitä haavoittuvuuksia löytyi 183 eri tietoturvatestauksesta.

Kriittisiä ja korkean vaikutuksen haavoittuvuuksia oli aineistossa eniten Työjonoon perustuvien verkkosovellusten tietoturvatestauksissa. Tietoturvatestausta kohden sama verkkosovelluskategoria sisälsi myös eniten kriittisiä ja korkean vaikutuksen haavoittuvuuksia.

Kriittisen ja korkean vaikutuksen haavoittuvuuksista suurin osa oli cross-site scripting haavoittuvuuksia. Stored XSS haavoittuvuuksia oli 77 kappaletta ja reflected XSS haavoittuvuuksia 54 kappaletta. Stored XSS ja reflected XSS haavoittuvuuksien jälkeen seuraavaksi eniten oli SQL injektio haavoittuvuuksia.

Taulukko 15. *Tietoturvatestauksien yksitoista yleisintä kriittistä haavoittuvuutta*

#	Haavoittuvuus	Haavoittuvuuksien lukumäärä
1.	Stored XSS	77
2.	Reflected XSS	54
3.	SQL injektio	43
4.	Cross-Site Request Forgery	37
5.	Direct Object Reference	29
6.	Käyttöoikeuksien laajentaminen	20
7.	Haitallisen tiedoston lisääminen	14
8.	Suora URL pääsy	12
=9.	Kirjautumisen ohitus	9
=9.	Command injection	9
=9.	Path Traversal	9

Kaikkia stored XSS haavoittuvuuksia ei ollut luokiteltu testaajien toimesta kriittisiksi tai korkean vaikutuksen haavoittuvuuksiksi. Keskimäärin noin joka kolmas stored XSS haavoittuvuus ei ollut kriittinen. Haavoittuvuuksien kriittisyys ei ole suoraan verrannollinen haavoittuvuuteen. Tämä johtuu tietoturvatestaajan näkemyksestä haavoittuvuuden vaikutuksesta itse verkkosovelluksen turvallisuuteen.

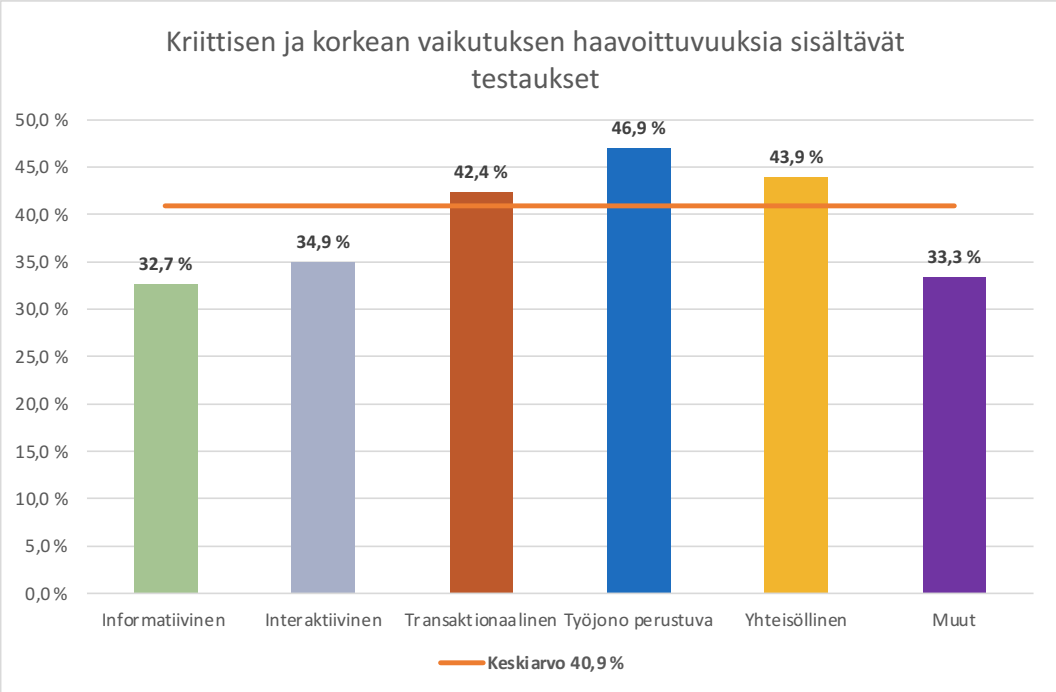
4.2 Päätelmiä tutkimustuloksista

Tässä luvussa käsitellään tutkimuksen tuloksia sekä pohditaan laajemmin mistä tulokset ovat voineet johtua. Tutkimuksen tuloksia tarkastellessa on huomioitava, että jokainen verkkosovellus on omanlaisensa. Tuloksista ei voida tehdä täydellistä johtopäätöksiä, sillä tutkimuksen kohteet ovat niin erilaisia. Testauksiin vaikuttivat muun muassa testauksen tilaajan määrittelemä laajuus, testaajan oma ammattitaito ja verkkosovelluksen tilanne testaushetkellä. Kaikki testauksen kohteet eivät olleet täysin valmiita verkkosovelluksia, vaan osa verkkosovelluksista testattiin osissa. Kuitenkin testauksien määrien johdosta pystytään arvioimaan joitain johtopäätöksiä.

4.2.1 Tuloksista verkkosovelluskategorioittain

Tutkimuksen tuloksista on nähtävissä, että työjonoon perustuvien ja transaktionaalisten kategorioiden verkkosovelluksissa on todennäköisemmin enemmän haavoittuvuuksia kuin muissa kategorioissa. Vastaavasti taas informatiiviset ja Muut-kategorian sovellukset sisälsivät vähiten haavoittuvuuksia.

Vastaavia tuloksia on nähtävissä, kun tarkastellaan kriittisen ja korkean vaikutuksen haavoittuvuuksia. Työjonoon perustuvien sovellusten tietoturvatestauksista löytyi 46,9 % todennäköisyydellä kriittinen tai korkean vaikutuksen haavoittuvuus. Nämä ovat kuvattuina kuvassa 6.



Kuva 6. *Kaavio kriittisten ja korkean vaikutuksen tietoturvahaavoittuvuuksien jakautumisesta verkkosovelluskategorioittain*

Se miksi sekä korkean vaikutuksen haavoittuvuuksia, että haavoittuvuuksia ylipäätään esiintyy todennäköisemmin nimenomaan työjonoon perustuvissa verkkosovelluksissa, selittynee osittain verkkosovelluksen kompleksisuudella. Enemmän ominaisuuksia sovelluksessa tarjoaa kyberhyökkääjille laajemman hyökkäyspinta-alan. Laajempi-hyökkäyspinta-ala vaikuttaa suoraan myös haavoittuvuuksien esiintymisen mahdollisuuksiin. Vastaavasti, suppeammat sovellukset, kuten yksinkertaiset verkkosivut, eivät sisällä itsessään niin paljon hyökkäyspinta-alaa.

Monimutkaiset verkkosovellukset laajuudessaan vaativat myös enemmän kehitystyötä. Kehittämiseen voi tällöin osallistua myös enemmän tekijöitä, jolloin riski virheisiin kasvaa. Todennäköisyys, että yksi henkilö tekee virheen sovelluksen kehitystyövaiheessa, on pienempi kuin useamman henkilön tapauksessa. Virheet voivat johtaa tietoturvahaavoittuvuuksiin, mikäli niitä ei osata tarkistaa ennen sovelluksen lopullista käyttöönottoa.

Se miksi Muut-kategorian verkkosovellukset esiintyivät paremmin esimerkiksi yhteisöllisiin sovelluksiin nähden, voidaan selittää kategorian sovellusten testauksista. Yleisesti Muut-kategorian sovellukset olivat osakokonaisuuksia sovelluksista. Esimerkiksi sähköisen allekirjoitusominaisuuden testaus, joka laskettiin Muut-kategoriaan, ei sisällä itsessään niin paljon hyökkäyspinta-alaa kuin esimerkiksi yhteisölliset sovellukset.

Täysin mahdotonta ei myöskään ole se, että tietyn tyyppisissä verkkosovelluksissa on valmiiksi kovempia tietoturvavaatimuksia. Kovemmat tietoturvavaatimukset oletettavasti suojaavat sovelluksia useammilta haavoittuvuuksista. Esimerkiksi verkkokauppojen sovelluksissa oletus on, että rahan siirtämisen ominaisuudet aiheuttavat kovempia tietoturvavaatimuksia. Maksukortilla maksamiseen verkkokaupoissa on yleensä käytössä PCI DSS tietoturvan vaatimuskehikko. Verkkokaupat käyttävät myös ulkoisia palveluntarjoajia. Nämä palveluntarjoajat pyrkivät huolehtimaan, että maksukorttivaatimukset toteutuvat ja että maksutapahtumat tapahtuvat tietoturvallisesti. [47]

Tulokset eivät kuitenkaan suoraan seuraa tätä oletusta. Transaktionaalisissa verkkosovelluksissa esiintyi usein kriittisen ja korkean vaikutuksen haavoittuvuuksia verrattuna vähemmän kompleksisiin sovelluskategorioihin.

Jos tutkitaan minkä tyylisiä haavoittuvuuksia transaktionaalisissa verkkosovelluksissa esiintyi, oli kyseessä yleisimmin konfiguraatiohaavoittuvuus. Kuitenkin suhteessa muihin sovelluskategorioihin transaktionaalisissa sovelluksissa ei esiintynyt yhtä todennäköisesti ohjelmointi- tai suunnitteluhaavoittuvuutta.

Myös interaktiivisissa verkkosovelluksissa oli kompleksisuuden oletuksiin nähden eroavaisuuksia. Interaktiivisissa sovelluksissa esiintyi tutkimuksessa eniten ohjelmointi- ja

suunnitteluhaavoittuvuuksia verrattuna muihin kompleksisempiin kategorioihin. Interaktiivisinten sovellusten todennäköisyys sisältää ohjelmointihaavoittuvuuksia oli 2,4 kappaletta ja suunnitteluhaavoittuvuuksia 0,7 kappaletta testausta kohden. Nämä olivat keskiarvolukemien 2,1 ja 0,5 yläpuolella. Tähän tulokseen ei ole antaa yksinkertaista selitystä. Sovellusten kompleksisuus ei vaikuta olevan sidoksissa sovellusten haavoittuvuuksien määriin haavoittuvuusosa-alueittain.

4.2.2 Konfiguraatiohaavoittuvuuksien yleisyys

Yleisesti verkkosovelluksista löytyneistä tietoturvahaavoittuvuuksista eniten esiintyi verkkosovelluksen konfiguraatiohaavoittuvuuksia. Niitä esiintyi eniten riippumatta verkkosovelluskategoriasta. Keskimäärin konfiguraatiohaavoittuvuuksia oli 3,6 kappaletta testausta kohden.

Yleisesti konfiguraatiohaavoittuvuudeksi luokiteltava sovelluksen datavuotohaavoittuvuus oli haavoittuvuuksista yleisin. Tämä johtunee siitä, että datavuodoksi arvioitiin testaajien toimesta esimerkiksi stack-trace virheviestit, joista paljastuu muun muassa webserverin tyyppi. Testauksen kohteet eivät aina olleet täysin käyttövalmiita versioita, jolloin esimerkiksi kehitysympäristössä tarvitut virheviestit ovat voineet jäädä poistamatta testattavasta versiosta.

Toiseksi yleisimpiä haavoittuvuuksia olivat puutteelliset SSL/TLS-protokollat ja algoritmit. Vanhentuneet SSL/TLS-asetukset ovat omiaan mahdollistamaan salatun tietoliikenteen kaappaamisen ja jopa salauksen purun. On huomioitava, että suuri määrä SSL/TLS-haavoittuvuuksista johtunee verkkosovellusten vaatimuksista toimia vanhemmissakin selainversioissa. Oletuksena on, että sovelluksia halutaan käytettävän mahdollisimman useassa verkkoselainversiossa. Myös sovelluksen kehitysvaiheessa on saatettu seurata ohjeistusta käyttää tuttuja SSL/TLS-konfiguraatioasetuksia.

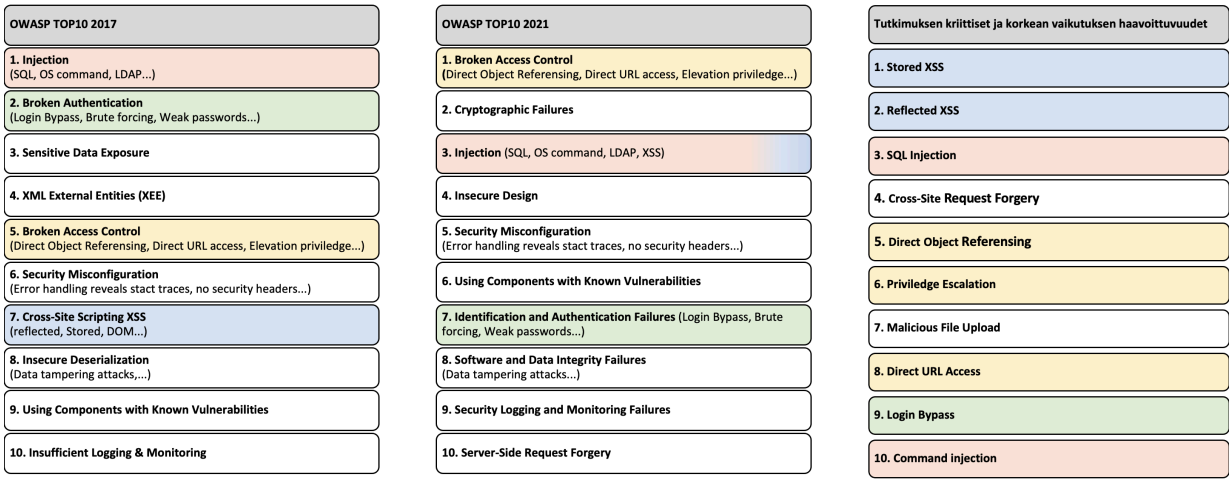
Puutteelliset HTTP-otsakeasetukset olivat kolmanneksi yleisimpiä haavoittuvuuksia. Näiden korkea esiintyvyys voinee johtua olittain haavoittuvuuden tyypistä. Puutteellinen HTTP-otsake-haavoittuvuus oli raportoitu, mikäli jokin sovelluksen turvallisuuteen vaikuttavista HTTP-otsakkeista oli puutteellisesti asennettu. Näitä otsakkeita ovat muun muassa X-XSS-Protection, X-Content-Type-Options, Content-Security-Policy ja X-Frame-Options. Puutteelliset otsakeasetukset mahdollistavat muita haavoittuvuuksia, esimerkiksi sovelluksen käyttämien resurssien epäturvallisen latautumisen. [48]

4.2.3 Kriittiset ja korkean vaikutuksen haavoittuvuudet

Kriittisistä ja korkean vaikutuksen haavoittuvuuksista yleisimpiä olivat XSS-haavoittuvuudet stored XSS ja reflected XSS. Nämä olivat yleisiä haavoittuvuuksia myös aikaisemmissa vastaavissa tutkimuksissa ja julkaisuissa. XSS-haavoittuvuudet olivat yleisimpiä WhiteHat Securityn [33] ja WASC:n [32] julkaisuissa. Nämä julkaisut eroavat taulukon 15 tuloksista kuitenkin sillä, että julkaisuissa ei ollut huomioitu erikseen vain korkean vaikutuksen haavoittuvuuksia.

Se miksi XSS-haavoittuvuuksia esiintyy paljon verkkosovelluksissa, johtunee haavoittuvuudesta itsessään. XSS-haavoittuvuudet voivat esiintyä useita kertoja samassa sovelluksessa. XSS-haavoittuvuus voikin johtua huolimattomuudesta, osaamattomuudesta tai esimerkiksi puutteellisesta suunnittelusta. XSS on myös tietoturva-alalla yleisesti yksi tunnetuimmista haavoittuvuustyypeistä, jolloin tietoturvatestaajat osaavat niitä myös yleisesti etsiä. Myös automatisoidut työkalut havaitsevat mahdollisia XSS-haavoittuvuuksia.

Kun verrataan yleisimpiä kriittisiä ja korkean vaikutuksen tietoturvahaavoittuvuuksia OWASP:n ylläpitämään top 10 listaukseen, on saman kaltaisia havaintoja huomioitavissa. Kuvasta 7 on nähtävissä OWASP:n Top 10 listaukset vuodelta 2017 [49] sekä vuodelta 2021 [29] ja tämän tutkimuksen kymmenen yleisintä kriittistä ja korkean vaikutuksen haavoittuvuutta.



Kuva 7. *Kaavio OWASP TOP 10 2017 & 2021 vertautumisesta aineiston kriittisiin haavoittuvuuksiin*

Toisin kuin tässä tutkimuksessa, OWASP kerää tietoturvatutkijoilta näkemyksiä haavoittuvuuksien trendeistä. OWASP käyttää myös haavoittuvuuskategorioita, jotka eroavat yksittäisistä haavoittuvuuksien nimistä. Muun muassa näistä syistä sijoitukset OWASP:n Top10 listauksissa eivät ole suoraan verrannollisia haavoittuvuuksien esiintymismääriin. Kuitenkin samoja haavoittuvuuksia, joita OWASP top10 listauksen haavoittuvuuskategorioissa on käytetty, esiintyi tuloksissa. Näin ollen tuloksien on nähtävissä seuraavan

OWASP:n tietoturvatutkijoiden näkemyksiä. Kuvassa seitsemän on väritettynä samaan OWASP:n haavoittuvuuskategoriaan kuuluvat haavoittuvuudet. Ehkä merkittävin eroavaisuus tuloksien ja OWASP top10 listauksien kanssa on cross-site request forgery haavoittuvuuden puuttuminen OWASP top10 listauksesta.

5. YHTEENVETO

Tässä luvussa käsitellään yhteenvetoa tutkimuksen tuloksista ja kuinka tuloksiin päästiin. Tämän lisäksi luvussa arvioidaan kriittisesti itse tutkimusta. Luvussa käsitellään myös tutkimuksesta kehiteltäviä mahdollisia jatkotutkimuksia.

5.1 Tulosten yhteenveto

Tässä tutkimuksessa tavoitteena oli vertailla tietoturvahaavoittuvuuksia sekä niiden esiintymistä erilaisissa verkkosovelluksissa. Sovelluksen kompleksisuudella oli tutkimuksen tuloksissa havaittavissa merkitystä sovelluksen tietoturvatestauksen haavoittuvuuksien määriin. Kompleksisempien sovellukset sisälsivät enemmän haavoittuvuuksia kuin vähemmän kompleksiset sovellukset. Kompleksisemmat sovellukset sisälsivät myös enemmän korkeamman vaikuttavuuden haavoittuvuuksia. Myös tietoturvatestauksien kokonaisarvosanat olivat keskiarvoisesti heikompia kompleksisimmissa sovelluksissa. Nämä tulokset vastasivat odotuksia.

Tutkimuksessa vertailtiin myös haavoittuvuuksien luokkia perustuen verkkosovelluskategorioihin. Konfiguraatiohaavoittuvuuksia esiintyi kaikissa sovelluskategorioissa yhtä paljon. Kompleksisuus ei kuitenkaan vaikuttanut ohjelmointi- ja suunnitteluhaavoittuvuuksien esiintymiseen. Vähemmän kompleksisen interaktiivistensovellusten kategorian testaukset sisälsivät kompleksisimpia sovelluksia enemmän ohjelmointi- ja suunnitteluhaavoittuvuuksia. Nämä tulokset eivät vastanneet tutkimuksen odotuksia.

Yleisimmät haavoittuvuudet tutkimuksessa liittyivät verkkosovelluksen tietojen vuotamiseen. Kuitenkin on huomioitava, että vuotanut tieto liittyi ennen kaikkea verkkosovelluksen hyödyntämän teknologiapinon tietojen paljastumiseen sekä tarpeettomiin virheviesteihin. Korkean vaikutuksen haavoittuvuuksista yleisimmiksi raportoituja olivat cross-site scripting haavoittuvuudet.

Tietoturvahaavoittuvuuksien vertailua varten kerättiin verkkosovellusten tietoturvaraporteista haavoittuvuuksien tietoja. Haavoittuvuuksista kerättiin niiden nimet ja tieto haavoittuvuuksien vaikuttavuudesta verkkosovelluksen turvallisuudelle. Haavoittuvuuksien lisäksi kerättiin tietoturvatestauksista kohdetietoja ja tulostietoja.

5.2 Tutkimuksen arviointia

Tutkimus on mielenkiintoinen aiheensa puolesta. Verkkosovellusten haavoittuvuuksien määriin perustuvaa vertailua ei ole tutkittu paljon, eikä vastaavan kaltaisia suomalaisia tutkimuksia ei löytynyt kirjallisuuskatselmuksen aikana lainkaan. Tutkimuksen puolesta puhuu kuitenkin aineiston määrä. Otoskoko tutkimuksessa oli melko kattava. Tietoturvatestattujen verkkosovelluskertojen määrä oli 447. Kategorioittainkin vertailua varten tietoturvatestauksia oli riittävästi. Myös tutkimuksen haavoittuvuudet ja niiden esiintyminen olivat samankaltaisia kuin vastaavissa ulkomaisissa tutkimuksissa. Tutkimuksen puolesta puhuu myös puolueettomuus, sillä tutkija itse ei ole osallistunut testattujen sovellusten tietoturvan arviointiin ja tuloksia on mahdollista vertailla lukumääriin perustuen.

Tutkimuksessa on kuitenkin tekijöitä, jotka vaikuttavat tutkimuksen tulosten täydelliseen vertailuun. Ensinnäkin tietoturvatestaukset oli suoritettu kyberturvallisuusorganisaation toimesta verkkosovelluksen ylläpitäjien tai omistajien tilauksesta verkkosovelluksiin. Tällöin tutkittavien sovellusten joukko on rajattu. sovellusten testaus ei ole jokaisessa testauksessa ulottunut koko sovellukseen, vaan testaus on voitu kohdistaa osaan sovellusta. Testauksen kohde on tällöin ollut rajoitetumpi, ja on mahdollista, että kaikkia haavoittuvuuksia ei ole siksi havaittu testauksen aikana.

Myös verkkosovelluksen testaukseen on olemassa tekijöitä, jotka vaikuttavat tutkimuksen tuloksiin. Tietoturvatestaukseen varattu testausaika vaihtelee testauskohteesta riippuen. Tietoturvatestaamista tiettyyn sovellukseen on mahdollista suorittaa ajallisesti hyvinkin pitkän ajanjakson ajan. Tässä tutkimuksessa käytettyjen tietoturvatestauksien aika on ollut rajattu. On mahdollistaa, että enemmän aikaa testauksiin käyttämällä, olisi haavoittuvuuksia löytynyt enemmän. Sovelluksia testasi myös eri henkilöt. Testaajat eroavat osaamistaidoiltaan haavoittuvuuksien paikallistamisessa ja raportoinnissa.

Myös tutkimuksen aikana on ollut mahdollista tapahtua virheitä. Tutkimuksen tietojen keräämistä ja kirjaamista toteutettiin manuaalisesti aineistosta johtuen. Virheitä on voinut tapahtua esimerkiksi tiedon syöttämisessä väärään sarakkeeseen. Myös verkkosovellusten kategorisoinnissa on kategorisointi tehty perustuen tutkijan parhaimpaan arvioon. Arvion perusteena on ollut tietoturvatestauksen raportin lausunnon kuvaus kohteesta.

Myös tietoturvahaavoittuvuuksien esiintyminen vaihtelee. Jotkin haavoittuvuudet esiintyvät tietyssä ajanjaksossa useammin kuin toiset. Tämän tutkimuksen haavoittuvuudet ovatkin yhden aikakauden otos.

5.3 Tulevaisuuden tutkimuskohteet ja tutkimuksen hyödyt

Mahdollisia jatkotutkimuskohteita on ainakin muutamia. Yleisesti se vaatii kuitenkin tutkimusaineiston laventamista. Nyt kerätyistä tietoturvatestauksen raporttien tiedoista hyödynnettiin tietoturvatestauksen kohdetta, testauksen arvosanaa, haavoittuvuuksien nimiä ja niiden vakavuutta. Esimerkiksi tietoturvatestauksien raporteissa yleensä on tapana ilmaista miten ja mistä jokin haavoittuvuus löytyy. Näistä on mahdollista tutkia esimerkiksi, missä syötekohdissa puutteet verkkosovelluksen suojauksessa tapahtuu. Tarkempi tieto siitä, mitkä syötteet ovat useimmiten turvattomampia auttaisi kohdistamaan verkkosovellusten tietoturvatestaamista näihin kohtiin.

Myös eri verkkosovellusten teknologiavalintojen suhdetta haavoittuvuuksien esiintymisiin olisi mielenkiintoista tutkia. Esimerkiksi tämän avulla saataisiin lisää tietoa, kuinka jokin tietty teknologiavalinta vaikuttaisi sovelluksen turvallisuuteen. Tutkimus olisi mielenkiintoinen, jos tuloksista olisi mahdollista tehdä johtopäätöksiä teknologiavalintojen turvallisuuteen. Useamman sovelluksia kehittävä tahon uskoisi olevan tämän kaltaisesta tiedosta kiinnostunut.

Yritys, jonka tietoturvatestausraportteja tutkimuksessa hyödynnettiin, hyötynee tutkimuksen aineiston koostamisesta. Aineistoa on mahdollista hyödyntää tietoturvatestauksien suunnittelussa sekä tietoturvatestauksien tulosten arvioinnissa. Tietoturvatestauksien tuloksista on mahdollista tehdä johtopäätöksiä tietoturvatestauksien haavoittuvuuksien löytämiseen. Myös verkkosovelluskategorioittain tyypillisesti esiintyvien haavoittuvuuksien hyödyntäminen on mahdollista ottaa käyttöön. Kuitenkin sen sijaan, että tietoturvatestaajat etsisivät vain määrällisesti eniten löytyviä haavoittuvuuksia, on kannattavampaa keskittyä löytämään haavoittuvuuksia kattavasti sovelluksista. Haavoittuvuuksien vaikutuksilla on merkitystä verkkosovelluksen turvallisuuden kannalta, eikä niiden määrissä.

LÄHTEET

[1] Wentrup, R. (2016). The online-offline balance: Internationalization for swedish online service providers. *Journal of International Entrepreneurship, 14*(4), 562-594. doi Saatavissa: http://dx.doi.org.libproxy.tuni.fi/10.1007/s10843-016-0171-2

[2] Shklar, Leon, and Rich Rosen. Web Application Architecture : Principles, Protocols and Practices. 2nd ed. Place of publication not identified: Wiley, 2009. Print.

[3] Kachan D (2020) 7 Reasons Why Businesses Are Migrating to Web Applications. https://dev.to/danakachan/7-reasons-why-businesses-are-migrating-to-web-applications-epg

[4] Types of web applications? Guru Technolabs blog. Saatavilla: https://www.guru-technolabs.com/types-of-web-applications/

[5] Syed Mohsin Saif et al., International Journal of Software and Web Sciences, 12(1), March-May, 2015, pp. 83-91 Saatavissa: http://iasir.net/IJSWSpapers/IJSWS15-261.pdf

[6] Murugesan, S., Ginige, A., “Web Engineering: Introduction and Perspectives.”, Web Engineering: Principles and Techniques,
Woojong, S. (ed.), Idea Group Publishing, 2005

[7] Kent, Stephen T., and Lynette I. Millett. Who Goes There? Authentication through the Lens of Privacy . Washington, D.C: National Academies Press, 2003. Print.

[8] Autentikaatio ja autorisaatio. Web-palvelinohjelmointi Java 2019 verkkokurssi, Helsingin yliopisto. 2019. Saatavilla: https://web-palvelinohjelmointi-19.mooc.fi/osa-5/2- autentikaatio-ja-autorisaatio

[9] Kyberturvallisuuden sanasto – Turvallisuuskomitea, 2018. Saatavissa: https://turvallisuuskomitea.fi/wp-content/uploads/2018/06/Kyberturvallisuuden-sanasto.pdf

[10] Unrestricted File Upload – OWASP. Saatavissa: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

[11] What is Transport Layer Security (TLS)? – Cloudflare. Saatavissa: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

[12] Logging Cheat Sheet – OWASP Cheat Sheet Series. Saatavissa: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

[13] Vulnerabilities – OWASP. Saatavissa: https://owasp.org/www-community/vulnerabilities/

[14] Watts S (2020). IT Security Vulnerability vs Threat vs Risk: What are the Differences? – BMC Blogs Saatavissa: https://www.bmc.com/blogs/security-vulnerability-vs-threat-vs-risk-whats-difference/

[15] TEPA-termipankki haku Exploit – Sanastokeskus TSK. Saatavissa: https://termipankki.fi/tepa/fi/haku/exploit

[16] Alves. A. Vulnerable vs. Exploitable: Why These are Different & Why it Matters – Threatstack blog. Saatavissa: https://www.threatstack.com/blog/vulnerable-vs-exploitable-why-these-are-different-why-it-matters

[17] Username Enumeration Vulnerabilities – Gnucitizen, 2007. Saatavissa: https://www.gnucitizen.org/blog/username-enumeration-vulnerabilities/

[18] TLS 1.0 and TLS 1.1 Are No Longer Secure – Packetlabs, 2019. Saatavissa: https://www.packetlabs.net/tls-1-1-no-longer-secure/

[19] Insecure Direct Object Reference – PortSwigger. Saatavissa: https://portswigger.net/web-security/access-control/idor

[20] Common Vulnerability Scoring System – NVD / NIST. Saatavissa: https://nvd.nist.gov/vuln-metrics/cvss

[21] Common Vulnerability Scoring System Calculator – NVD / NIST. Saatavissa: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

[22] Voth, Jack, Andrew Whitaker, and Keatron Evans. *Chained Exploits: Advanced Hacking Attacks from Start to Finish*. Addison-Wesley Professional, 2009. Print.

[23] What is reflected cross-site scripting? – PortSwigger. Saatavissa: https://portswigger.net/web-security/cross-site-scripting/reflected

[24] Nidecki, What are Open Redirects? – Acunetix blog, 2020. Saatavissa: https://www.acunetix.com/blog/web-security-zone/what-are-open-redirects/

[25] Common Weakness Enumeration – CWE. Saatavissa: https://cwe.mitre.org/index.html

[26] Common Vulnerabilities and Exposures – CVE. Saatavissa: https://www.cve.org/

[27] OWASP Security Headers Project – OWASP. Saatavissa: https://owasp.org/www-project-secure-headers/

[28] GraphQL Cheat Sheet – OWASP Cheat Sheet Series. Saatavissa: https://cheatsheetseries.owasp.org/cheatsheets/GraphQL_Cheat_Sheet.html

[29] OWASP Top 10 2021 – OWASP tietoturvaprojekti. Saatavissa: https://owasp.org/Top10/

[30] Vulnerabilities by Type – CVE. Saatavissa: https://www.cvedetails.com/vulnerabilities-by-types.php

[31] Unrestricted File Upload – OWASP. Saatavissa: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

[32] Web Application Security Statistics – WASC, 2007. Saatavissa: http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics#Methodology

[33] WhiteHat Website Security Statistics Report – WhiteHat Security 2007. Saatavissa: http://hhs.janlo.nl/articles/Whitehatstat.pdf

[34] Pressmann, S. & B. Maxim (2012). Software Engineering a Practioner's Approach. 8th edition. McGraw Hill education. ISBN 978-0-07-802212-8

[35] Daniel P. Newman, and Andrew Whitaker. *Penetration Testing and Network Defense*. Cisco Press, 2005. Print.

[36] Khan M.E. ja Khan F. (2012) A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. Saatavissa: https://www.researchgate.net/publication/270554162_A_Comparative_Study_of_White_Box_Black_Box_and_Grey_Box_Testing_Techniques

[37] OWASP Web Security Testing Guide – OWASP. Saatavissa: https://owasp.org/www-project-web-security-testing-guide/

[38] The Beginners' Guide to Bug Bounty Programs - Hackerone. Saatavissa: https://www.hackerone.com/beginners-guide-bug-bounty-programs

[39] S. Türpe, "Security Testing: Turning Practice into Theory," *2008 IEEE International Conference on Software Testing Verification and Validation Workshop*, 2008, pp. 294-302, doi: 10.1109/ICSTW.2008.38.

[40] Testing for Priviledge Escalation – OWASP Web Security Testing Guide. Saatavissa: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/03-Testing_for_Privilege_Escalation

[41] Pohjola (2019) Tietoturvaraporttien yhteenmuotoilu. Saatavissa: https://www.theseus.fi/handle/10024/168165

[42] Caido - A lightweight web security auditing toolkit. Saatavissa https://caido.io/

[43] Zap - OWASP Zed Attact Proxy. Saatavissa: https://www.zaproxy.org/

[44] BurpSuite Penetration testing tool. PortSwigger. Saatavissa: https://portswigger.net/burp

[45] Fuzzing – OWASP. Saatavissa: https://owasp.org/www-community/Fuzzing

[46] P. Ritala. Johdatus tutkimusmetodologiaan – Kauppatieteellinen tiedekunta, LUT. Saatavissa https://developmentcentre.lut.fi/digi/Moodle_pohjat/Ritala_Johdatus%20tutkimusmetodologiaan%202013.pdf -sivu 47

[47] PCI Security Standards Council, LLC. Saatavissa: https://www.pcisecuritystandards.org/

[48] Ickler Kent (2017) How To Fix a Missing Content-Security-Policy on a Website – Black Hills Information Security. Saatavissa: https://www.blackhillsinfosec.com/fix-missing-content-security-policy-website/

[49] OWASP Top 10 2017 tietoturvaprojekti. Saatavissa: https://owasp.org/www-project-top-ten/

LIITE A: HAAVOITTUVUUKSIEN MÄÄRÄT

Sija	Haavoittuvuus	Haavoittuvuuksien lukumäärä	Kriittisiä haavoittuvuuksia
1.	Sensitive Data Disclosure / Data Leak / Deliberate Error Conditions /Improper Error Handling	236	4
2.	Insufficient Transport Layer Protection	197	0
3.	Insufficient Client Protection / Missing HTTP (security) Headers	171	0
4.	Stored XSS	124	77
5.	HTTP Header Fingerprinting	122	0
6.	Reflective XSS	117	54
7.	Token Value Access via Eavesdropping (missing secure flag)	104,5	0
8.	Components with known vulnerabilities	102	2
9.	Username Harvesting	99	2
10.	Cross-Site Request Forgery	89	37
11.	Token Value Access via Code (missing httpOnly Flag)	88,5	5
=12.	Unencrypted Communications (HSTS Header misisng)	85	2
=12.	Brute-forcing Login Form / Voucher codes	85	1
14.	Missing security updates	78	5
15.	Direct URL Access	58	12
16.	Cross-Origin Resource Sharing	53	7
17.	Insecure Direct Object Referencing	51	29
18.	Malicious File Upload	50	14
=19.	User Interface Redressing / Clickjacking	47	0
=19.	Denial of Service	47	8
=21.	Default services and content	46	1
=21.	SQL injection	46	43

23.	Parameter Tampering / Log Tampering	43	2
24.	Weak Cryptographic Ciphers	42	0
25.	Open Redirect	39	2
26.	Session Termination Prevention	36	0
27.	Using vulnerable services	32	4
28.	Privilege Escalation / Account takeover / Role manipulation	29	20
29.	Improper / Missing input validation / Client-side input validation	25	0
30.	Session Fixation	24	0
31.	Command Injection	22	9
=32.	Admin Interface Disclosure	21	0
=32.	Login Bypass / Authentication Bypass / CAP-TCHA Bypass	21	9
=34.	Directory Browsing	19	0
=34.	Dangerous HTTP Methods	19	0
=36.	Server-Side Request Forgery	18	5
=36.	Password Recovery Abuse	18	4
=38.	Version Infromation Disclosure	16	0
=38.	DOM-based XSS	16	4
=38.	Lack of Rate limiting / Brute-forcing / Resource exhaustion	16	0
41.	Sensitive Services Accessible from Internet	14	0
=42.	Path Traversal	12	9
=42.	Circumvention of Workflow / Abuse of function-ality	12	0
=42.	Insufficient Security Controls	12	0
45.	Weak Authorization Token	11	2
46.	Vulnerable SSL/TLS Implementation	10	0
=47.	Certificate Related Problems	9	1
=47.	XML injection	9	3
=47.	Input returned in response	9	0

=50.	Sensitive Data In Session Tokens	8	0
=50.	HTTP Parameter Pollution	8	2
=50.	Weak logout / Catchable HTTPS Response	8	0
=53.	Access of Privileged Service	7	0
=53.	Sensitive Data in Client-side code	7	0
55.	Session Token Value Guessing	6	0
=56.	Unnecessary Services and Default Files	5	0
=56.	Abuse of Functionality	5	0
=56.	GraphQL introspection and deprecated fields	5	0
=59.	Service Fingerprinting	4	0
=59.	Weak password requirements / default password configuration	4	0
=59.	Login Form autocomplete / Password autocomplete	4	1
=59.	Flash Cross-Domain Policy	4	0
=63.	Confidential Data Access in Clear-Text	3	1
=63.	Access Cached Credentials	3	0
=63.	Host Header injection	3	0
=63.	Content enumeration	3	0
=63.	Eavesdropping data in transit	3	0
=68.	HTTP Response Splitting	2	0
=68.	Cross-Site WebSocket Hijacking	2	0
=68.	Eavesdropping Credential Submission	2	0
=68.	Arbitary Password Selection	2	0
=68.	Insecure Cryptographic Storage	2	0
=68.	HTTP-Request Smugling	2	0
=68.	Default User Credentials	2	0
=68.	Insufficient Credential Protection	2	0
=68.	Insecure Content-Type Validation	2	0
=68.	Remote code execution	2	0

=68.	Simultaneous Session Logons	2	0
=68.	Port Scanning / Open Ports	2	0
=68.	Request URL Override	2	0
=68.	Interger overflow, Memory Leak	2	0
=68.	GraphQL Information Disclosure or Deprecated Fields	2	0
=68.	GraphQL schema Enumeration	2	0
=68.	Monitoring bypass	2	0
=85.	2-step Authentication Bypass	1	0
=85.	LDAP	1	0
=85.	Replay attack	1	0
=85.	Lack of Physical security controls	1	1
=85.	Cross domain Scripting	1	0
=85.	ASP.NET ViewState without MAC Enabled	1	0
=85.	Open Information Harvesting	1	0
=85.	Insecure File Sharing	1	0
=85.	Credit card information manipulation	1	1
=85.	Unnecessary arbitrary string reflection	1	0
=85.	No evidence of unauthenticated access	1	0
=85.	Static API key	1	0
=85.	Illegal State Transition	1	0
=85.	Sandboxin bypass	1	1
=85.	Incorrect content type	1	0
=85.	Spoofing SMS sender	1	0
=85.	Root OS Privileges for Application	1	0
=85.	External Service Interaction	1	0
=85.	Insufficient Content-Security-Policy	1	0
=85.	Inconsistencies in UI for different roles	1	0

=85.	Protocol Downgrade Protection	1	0
=85.	Malicious URL inclusion	1	1
=85.	Quality aspects of application	1	0
=85.	HTTP Cache Poisoning	1	0
=85.	Cross-Site Tracing (XST)	1	0
=85.	API Authentication	1	0
=85.	Undocumented Endpoints	1	0
=85.	Lack of Expiration on Registration Link	1	0