Haaga-Helia
ammattikorkeakoulu Oy

Laajasalon opiston lainausjärjestelmän tietoturva ohjelmointinäkökulmasta

Jyri Lindroos

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2015

Haaga-Helia
ammattikorkeakoulu Oy
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyö

9.6.2015

Tekijä tai tekijät Jyri Lindroos	**Ryhmätunnus tai aloitusvuosi** 2010
Raportin nimi Laajasalon opiston lainausjärjestelmän tietoturva ohjelmointinäkökulmasta	**Sivu- ja liitesivumäärä** 47 + 13
Opettajat tai ohjaajat Irene Vilpponen	

Tämä opinnäytetyö tarkastelee verkko-ohjelmoinnin tietoturvanäkökulmia. Työn taustalla on tarve kehittää pienehkölle opistolle uusi lainausjärjestelmäsovellus. Tämä siksi, että aiemmassa sovelluksessa esiintyi turvallisuuteen liittyviä riskejä.

Opinnäytetyön teoriaosuudessa kuvataan tietoturvaa eri näkökulmista. Tavoitteena selvittää ja dokumentoida tietoturvaan liittyviä asioita ohjelmoitaessa sovellusta. Samoin opinnäytetyössä kuvataan menetelmiä tietoturvan saavuttamiseksi ja ylläpitämiseksi.

Tässä työssä tarkastelu on rajattu ohjelmointikieliin, jotka ovat siirrettäviä alustalta toiselle. Näitä ovat Java, JavaEE, Java Server Page, JavaScript, MariaDB, HTML5 ja CSS3. Näitä ohjelmia käytetään ohjelmiston ohjelmoimiseen.

Työssä esitellään kyseiset ohjelmointikielet ja tarkastellaan jokaisen tietoturvaa ohjelmointikielen näkökulmasta.

Opinnäytetyön suunnittelu aloitettiin syksyllä 2014. Tällöin tehtiin alustavat kartoitukset ja kerättiin asiakaskokemukset. Varsinainen ohjelmointi ajoittui keväälle 2015. Samoin teoriaosuuden kirjoitus.

Opinnäytetyön pohdinnassa kuvataan asiakkaalle tulevaa lisäarvoa, jonka uusi sovellus tuo vanhaan verrattuna. Työn konkreettisena tuloksena on lainausjärjestelmäohjelma, jossa tietoturvaa on huomioitu eri näkökulmista. Pohdinnassa kuvataan myös teknologiaa, palvelimen ominaisuuksia, joita ratkaisu edellyttää.

Asiasanat
Ohjelmointi, tietoturva, internet, palvelimet

HAAGA-HELIA
University of Applied Sciences

Degree programme in Information Technology

Thesis

09.06.2015

Authors Jyri Lindroos	**Group or year of entry** 2010
The title of thesis THE SECURITY ANGLE IN PROGRAMMING OF BORROWING SYSTEM IN LAAJASALO'S OPISTO	**Number of pages and appendices** 53 + 14
Supervisor(s) Irene Vilpponen	

This thesis examines programming information security. It arises from a need to develop a new borrowing system application for a small college as their earlier application appears to have security-related risks.

The theoretical section of the thesis describes the information security from different angles. The aim is to clarify and document security issues related to programming the application. The thesis also describes methods of achieving and maintaining information security.

In this work, the review is limited to programming languages, which are movable from one platform to another: Java, JavaEE, Java Server Page, JavaScript, MariaDB, HTML5 and CSS3.

The work presents these programming languages and examines the security of each of them from the point of view of programming language. These programs are also used to program the actual software.

The thesis was started in the autumn of 2014 when the preliminary surveys were made and customer experiences were collected. The programming took place in the spring of 2015 and the theoretical part was also written then.

The discussion of the thesis describes the future value to the customer which the new application will bring compared to the old one. A concrete result is a borrowing system application in which security is taken into account from different perspectives. The discussion also describes the technology and the server features that the application requires.

Key words
Programming, information security, internet, servers

Sisällys

1 Johdanto

Tietoturvanäkökulma on yhä tärkeämpi ohjelmoinnissa ja ohjelmistoissa nykypäivänä. Pienemmissä organisaatioissa tai opistoissa, kuten tässä opinnäytetyössä on kyseessä, tietoturvahaasteet liittyvät muun muassa useiden eri muuttuvien käyttäjien kautta ohjelmistoturvallisuuteen.

Tässä työssä tarkastellaankin ohjelmoinnin tietoturvaa. Turvauhat voivat tulla vallitsevasta fyysisestä luonnonympäristöstä ja sen katastrofeista kuten tulvista, tornadoista, luonnon paloista, maanjäristyksistä ja tulivuorenpurkauksista. Muita uhkia voi aiheutua henkilöstöuhista, laitteistovioista, sähkönsaantiin liittyvistä pulmista, tuhopoltoista, sabotaaseista, lakoista, terrorismista tai sodista. Ohjelmapuolen uhat tulevat puolestaan suoraan ohjelmista, joita käytetään kyseisen ohjelman ohjelmoimiseen tai suorittamiseen. Laatiessaan katastrofitoipumissuunnitelmaa yrityksen tulee huomioida riskit laajemmin jo valmisteluvaiheessa. Tämä opinnäytetyö voisi olla osa yrityksen katastrofitoipumissuunnitelmaa.

Ajatus opinnäytetyöstä sai alkunsa Laajasalon opiston laitelainausjärjestelmän puutteista. Ohjelmoija ei enää päivittänyt ohjelmaa, joten sitä kautta ei apua ollut saatavissa. Koska järjestelmä oli tehty pääosin PHP:llä, olisi sitä voitu kuitenkin korjata. Dokumentaation puutteiden vuoksi, nopeampi sekä halvempi keino oli lähteä kuitenkin suunnittelemaan ja toteuttamaan uutta ohjelmistoa lainausjärjestelmään. Alussa myös etsittiin valmista ohjelmaa, jonka olisi voinut räätälöidä opiston tarpeisiin. Tätä ei kuitenkaan löytynyt. Näistä syistä päädyttiin ratkaisuun ohjelmoida sellainen.

Laajasalon opisto on Suomen vanhin kristillinen kansalaisopisto. Se perustettiin viime vuosisadan alussa Helsingin Sörnäisiin ja Laajasaloon se muutti viime vuosisadan puolessa välissä. Laajasalon opistossa opiskelee vuosittain n. 300 opiskelijaa, joista n. 250 vuoden kestävillä linjoilla ja loput puolivuotta kestävillä kielilinjoilla.

Laajasalon opiston opetus on mediapainotteista ja tämän vuoksi lainattavaa laitteistoa on paljon. Tämä on yksi syy, miksi tarvetta uudelle lainausjärjestelmälle oli olemassa. Paine vahtimestarin puolella kasvoi, kun tietojen syöttötyö lainaamisten yhteydessä oli runsas.

Projektin aihetta esiteltiin työyhteisössä työtovereille ja heiltä tuli arvokkaita kehitysehdotuksia. Opinnäytetyön näkökulmasta ehdotuksia rajattiin ja osa ehdotuksista

rajattiin ulkopuolelle. Tämä siksi, ettei opinnäytetyö olisi kasvanut sisällöltään liian laajaksi.

Uuden järjestelmän avulla vahtimestari pystyy vähentämään kirjoitustyön minimiin, kun opiskelijat ja ohjelma tekevät sen jo etukäteen ja hänelle jää vain tavaroiden luovutus ja vastaanotto.

Aiheeseen perehtymisen jälkeen syntyi oivallus laajemmin hyödynnettävästä sovelluksesta, joka ei ole sidottu ainoastaan yhteen opistoon, vaan olisi hyvin pienin muutoksin käytettävissä kenelle tahansa vastaavanlaista lainajärjestelmää tarvitsevalle.

Aikataulu työlle muodostui siten, että syksyn 2014 aikana kerättiin ohjelmaa varten taustatietoa, mitä parannusehdotuksia ja mahdollisia muutoksia entiseen ohjelmaan haluttiin ja kevään 2015 aikana suoritettiin itse koodaus ja testaus. Opinnäytetyö eli tämä dokumentti aloitettiin kevään 2015 alussa ja tarkoitus oli saada tämä valmiiksi toukokuuhun mennessä.

Opinnäytetyön aihe rajautui vain niihin ohjelmointikieliin, joita käytettiin varsinaisessa ohjelmointityössä. Lopussa on pieni katsaus myös muihin ohjelmointikieliin, lähinnä siltä kannalta, jos ratkaisu olisi ollut toinen.

2 Ohjelmointikielten esittely ja niiden tietoturva

Varsinaiseen työhön ohjelmoinnissa käytetään seuraavia kieliä: Java, JavaEE, Java Server Pages, MariaDB, HTML5, CSS3 sekä JavaScript, joita kuvataan seuraavissa kappaleissa. Kyseisten kielten tärkein valintakriteeri oli niiden siirrettävyys alustalta toiselle. Haluttiin käyttää myös uusinta mahdollista tekniikkaa, joka tarjoaa turvaratkaisuiltaan tuoreinta näkemystä tietoturvaongelmiin.

Tässä luvussa kuvataan myös ohjelmointikielten tietoturvaa. Tämän lisäksi luvuissa 4. ja 5. syvennytään tietokannan tietoturvaan ensin yleisellä tasolla ja sitten käyttäjäkohtaisella tasolla. Lisäksi liitteessä 2. on komentoja, joilla käyttäjäkohtaista tietoturvaa voidaan hallita.

Ohjelmoitaessa käytettiin kaikkia yllä olevia ohjelmistokieliä hyväksi seuraavasti: tieto tallennettiin MariaDB-tietokantaan ja Java toimi varsinaisena ohjelmointikielenä. Jotta ohjelmistosta saatiin riittävän tietoturvallinen, lisättiin mukaan JavaEE-ominaisuuksia, joka on selitetty luvussa 2.4. sekä JavaServer Page –ominaisuuksia, joiden tietoturva on selitetty luvussa 2.6. Käyttöliittymä tehtiin käyttäen HTML5-kieltä sekä CSS3:n ulkoasuominaisuuksia. JavaScriptiä käytettiin tarkastamaan HTML5-lomakkeiden syötteet, että niissä oleva tieto oli oikeamuotoista ja varmistamaan, ettei lomakkeiden kautta päässyt tekemään esim. XSS-hyökkäyksiä.

2.1 Java ja sen tietoturva

Java on ohjelmointikieli, joka on ensisijaisesti suunniteltu käytettäväksi internetin hajautetussa ympäristössä. Sen tarkoituksena on näyttää ja tuntua C++ -kieleltä, mutta olla yksinkertaisempi ja se käyttää olio-ohjelmointimallia. Javaa voidaan käyttää luomaan ohjelmia, joita voidaan ajaa joko yhdessä tietokoneessa, jakaa palvelimilla tai asiakkaiden verkoissa. Sillä voidaan rakentaa pieni sovellusmoduuli tai applikaatio, jota käytetään websivun osana. Applettien tuovat interaktiivisuuden mukaan, eli käyttäjät voivat esimerkiksi antaa ohjelman tarvitsemia tietoja. Tämä on yksi tärkeimmistä syistä siihen, miksi kyseinen kieli valittiin käyttökieleksi. (Rouse 2007.)

Sun Microsystems esitteli Javan vuonna 1995 ja loi heti uudet interaktiiviset mahdollisuudet webiin. Molempiin suuriin selaimiin tuli Java -tuki. Lähes kaikki suurten käyttöjärjestelmien kehittäjät lisäsivät Java -kääntäjät osaksi tuotetarjontaansa. (Rouse 2007.)

Tärkeimpiä Javan ominaisuuksia ovat siirrettävyys, vahva koodi sekä oliopohjaisuus. Ohjelmat ovat siirrettäviä verkossa. Lähdekoodi on käännetty tavukoodiksi (Javan termi), jota voidaan ajaa missä tahansa verkon palvelimella tai asiakaskoneessa, jossa on Javan virtuaalikone. Javan virtuaalikone tulkitsee (kääntää) tavukoodin koodiksi, jonka se ajaa todellisessa tietokonelaitteistossa. Tämä tarkoittaa sitä, että tietokoneista riippuvat erot voidaan tunnistaa ja sovittaa paikallisesti samanaikaisesti, kun ohjelmaa suoritetaan. Tämä johtaa siihen, ettei sovellusalustakohtaisia ohjelmaversioita enää tarvita. (Rouse 2007.)

Koodi on vahva, joka tarkoittaa sitä, että Java objektit eivät sisällä viittauksia ulkopuolisista tiedoista itseensä tai muihin tunnettuihin objekteihin. Toisin kuin ohjelmissa, jotka on kirjoitettu C++:lla ja mahdollisesti muilla ohjelmointikielillä. Näin varmistetaan, että käsky ei voi sisältää tietovaraston osoitetta toisesta ohjelmasta tai käyttöjärjestelmästä, joka voisi aiheuttaa ohjelman tai jopa käyttöjärjestelmän kaatumisen. Java-virtuaalikone tekee useita tarkastuksia jokaiseen kohteeseen varmistaakseen tiedon eheyden. (Rouse 2007.)

Java on olio-pohjainen ohjelmointikieli, joka tarkoittaa muun muassa sitä, että kohde voi hyödyntää osaa luokan kohteista ja periä koodia, joka on yhteistä koko luokalle. Metodi voidaan käsittää kohteen kyvyksi tai käytökseksi. Sen lisäksi, että Java toteutetaan asiakkaan koneella palvelimen sijasta, Java-appletilla on muita ominaisuuksia jotka on suunniteltu tekemään sen nopeammaksi. (Rouse 2007.)

Java-virtuaalikone sisältää valinnaisen just-in-time (JIT) kääntäjän, joka dynaamisesti kokoaa tavukoodin osaksi suoritettavaa koodia vaihtoehdoksi yhden tavukoodin tulkintaa kerrallaan. Monissa tapauksissa dynaaminen JIT kääntäminen on nopeampi kuin virtuaalikoneen tulkinta. (Rouse 2007.)

JavaScriptiä ei pidä sekoittaa Javaan. Se sai alkunsa Netscape-selaimesta, sitä tulkitaan korkeammalla tasolla, mutta siitä puuttuu Javan tavukoodin siirrettävyys ja nopeus. (Rouse 2007.)

Java on yksinkertainen oppia ja se tarjoaa turvallisen sovellusalustan. Se on ihanteellinen verkkosovelluksille ja on siirrettävissä alustalta toiselle (Linux, Mac, Windows). Se ei ole alusta riippuvainen, vaan on olio-pohjainen ja vahvasti tyypitetty kieli. (eduCBA Academy for IT Training 2014.)

Javan turvallisuusteknologia sisältää ison joukon rajapintoja, työkaluja ja yleisesti käytettyjä sovelluksia kuten turvallisuusalgoritmeja, mekanismeja ja protokollia. Javan turvallisuusrajapinnat ulottuvat laajoille alueille, kuten kryptografia, yleisen avaimen infrastuktuuri, turvallinen kommunikointi, autentikointi ja pääsynvalvonta. (Oracle Technology Network 2014.)

Javan turvallisuusteknologia tarjoaa käyttäjille kattavat turvallisuuspuitteet kirjoittaa sovelluksia ja se tarjoaa ohjelmistokehittäjille tai ylläpitäjille joukon työkaluja turvallisesti hallita ohjelmia. Javan turvallisuusmalli perustuu muokattavaan ”hiekkalaatikkoon”, jossa Java ohjelmia voidaan ajaa turvallisesti ilman mahdollisia systeemi- tai käyttäjäriskejä. (Oracle Technology Network 2014.)

Oracle ohjeistaa eritasoisia käyttäjiä omilla sivuillaan. Oraclen mukaan **ohjelmistokehittäjien** on tutustuttava ”Java 8 turvallisuuden kohokohdat” -sivustoon sekä Java 7:n ja Java 8:n välisiin turvallisuuseroihin. Samoin JDK 8:n uusi ”Kirjoituskommentti”-dokumentti on integroitava ohjelmiin. Tämä helpottaa estämään, löytämään ja korjaamaan virheitä. Lisäksi puolustusohjelmointistrategioiden oppimista pidetään tärkeänä, jotta voidaan oikein vähentää ohjelman heikkouksia ja estää haavoittuvuuksia. (Oracle Technology Network 2014.)

”Tietoturvaparannukset JDK 8:ssa” –sivusto pitää sisältää monia uusia kryptografisia algoritmeja, parannetun satunnaistamisen sekä protokollapäivityksiä. Appletteja ja Web-sovelluksia varten on käytävä läpi ”RIA turvallisuustarkastuslista” –sivusto ja ymmärrettävä myös ”koodiallekirjoituksen laajennettu rooli” kun tarkastaan ja todennetaan loppukäyttäjien oikeuksia. (Oracle Technology Network 2014.)

Java SE:n turvallisuus -kohdassa kuvataan rajapintoja, määritelmiä ja ohjelmistokehittäjiin liittyviä käyttöönottotietoja. Näitä ovat esimerkiksi koodiallekirjoitus ja aikaleima. Myös ”Kryptografisten algoritmien nimet” –sivustoon perehtyminen on suositeltavaa, sillä se pitää sisällään Javan Kryptografisen arkkitehtuurin. Jotta saadaan parempi ymmärrys Javan turvallisuudesta, kannattaa osallistua yhteisöjen kuten AvoinJDK turvallisuusryhmän keskusteluihin. Oraclen Java tuotehallintablogia seuraamalla ja tilaamalla aiheeseen liittyviä RSS -linkkejä saa arvokasta tietoa. (Oracle Technology Network 2014.)

Oracle ohjeistaa **systeemin ylläpitäjiä** taas seuraavasti: systeemin ylläpitäjät vastaavat siitä, että he ajavat Java sovelluksia tietoturvallisella tavalla seuraten ’vähimmän

etuoikeuden' -periaatetta. Tällä tarkoitetaan vain sellaisten oikeuksien antamista käyttäjälle, mitä hän tarvitsee. (Rouse 2008.)

Tietoturvasta vastaavien täytyy asentaa kriittinen paikkaustiedosto ”Java 8u31” tai ”7u75”, joka poistaa käytöstä SSLv3 POODLE:n oletuksena tai poistaa SSLv3 käytöstä vanhemmista versioista itse käsin. Tällä turvataan tarvittavat vanhat versiot seuraamalla käyttöä, vähennetään hyökkäyspintaa ja ylläpidetään yhteensopivuutta. (Oracle Technology Network 2014.)

”Java 8 turvallisuuden kohokohdat” –sivuston sisältöön on perehdyttävä, joka pitää sisällään yhteenvedon muutoksista Java 7:n ja Java 8:n välillä. Ylläpitäjän on kontrolloitava TLS:n (katso liite 2.) yhteensopivuutta palvelimilla ja työasemilla ja valita kaikkein turvallisin ja yhteensopivin vaihtoehto. (Oracle Technology Network 2014.)

”Tietoturvaparannukset JDK 8:ssa” –sivustoon on paneuduttava. Se sisältää uusia kyptografisia algoritmeja, parannetun satunnaistamisen sekä protokollapäivitykset. Ajan tasalla pysyminen mahdollistuu muun muassa tilaamalla sähköpostiin kriittisten korjauspäivitysten tiedotukset sekä lukemalla niiden yleiset informaatiosivut. Tarvittaessa voidaan hallita useita Java versioita staattisten asennusten ja käyttöönottosääntömääritysten avulla, jotta myös vanhat versiot toimisivat yhteensopivasti. Samoin Applet ja Webpohjaisten ohjelmien ”whitelist” –sivustolta saa tietoa. Siellä kerrotaan ristiin hallinnoitujen tietokoneiden käyttöönottosäännöistä. (Oracle Technology Network 2014.)

”JRE Server” –palvelinjärjestelmien käyttöä on harkittava. Näitä ovat esimerkiksi ohjelmapalvelimet ja muut pitkään jatkuvasti taustalla ajettavat prosessit. JRE Server on muutoin sama kuin JRE paitsi, että JRE Server ei sisällä web-selain liitännäistä. Luotettujen aikaleimojen käyttö, kun allekirjoitetaan ja tarkistetaan allekirjoitettuja JAR-tiedostoja (kts. liite 2.), estää tuotteen vanhentumista liian aikaisin. Samoin ylläpitäjän on tutustuttava dokumenttiin ”ominaisuudet, joita ei voi konfiguroida” Java asennusten sisällä. ”Java SE:n turvallisuus yleisesti”-dokumentti sisältää listan rajapinnoista, määritelmistä ja ohjelmistokehittäjiin liittyvistä käyttöönottotiedoista (kuten koodiallekirjoitus ja aikaleima). (Oracle Technology Network 2014.)

Loppukäyttäjät, jotka käyttävät Javaa koneissaan, joutuvat ottamaan vain muutaman askeleen varmistaakseen ja ymmärtääkseen Javan tietoturvan heidän laitteissaan. Heidän kannattaa lukea dokumentti ”Kuinka lisätä poikkeussivu”. Toiseksi heidän kannattaa aina käyttää uusinta Java versiota koneissa. Heidän on huolehdittava siitä, että Java versio

ladataan vain hyväksytyistä kohteista. Jos Javalle ei ole tarvetta, se kannattaa kytkeä pois web-selaimesta. Dokumentista ”mitä muita toimia” loppukäyttäjä saa vinkkejä siitä, miten voi lisätä Javan turvallisuutta. (Oracle Technology Network 2014.)

Turvallisuusammattilaiset, jotka suorittavat järjestelmän tarkistuksia, uhkamallinnuksia, arkkitehtuurikuvauksia tai käyvät läpi Java ohjelmien koodia on tutustuttava Javan turvallisuusarkkitehtuuriin ja rajapintadokumentointiin. Heidän on arvioitava ”kehittynyttä hallintakonsolia” turvatakseen tarvittavat vanhat versiot seuraamalla käyttöä, vähentämällä hyökkäyspintaa ja ylläpitämällä yhteensopivuutta. Heidän on varmistettava, että kaikki systeemit ovat ajan tasalla ja viimeisimmät turvallisuuspaikkaustiedostot on asennettu. Heidän tulee myös tutustua dokumenttiin ”Kuinka tarkistaa, onko SSLv3 päällä Java SE:ssä (ja kuinka ottaa se pois päältä)”. (Oracle Technology Network 2014.)

Uusimmat turvallisuusesitykset on luettava sekä perehdyttävä ”Java SE:n turvallisuus yleisesti”-dokumenttiin, joka pitää sisällään listan rajapinnoista, määritelmistä ja kehittäjiin liittyvistä käyttöönottotiedoista (kuten koodiallekirjoituksesta ja aikaleimasta). Lisäksi heidän on opittava puolustusohjelmointistrategiat, jotta he voivat oikein vähentää ohjelman heikkouksia ja estää haavoittuvuuksia. Samoin turvallisuusasiantuntijan on tutustuttava turvallisuusmäärityksiin, kuten dokumentteihin ”kryptografisten algoritmien nimiin”, ”Javan kryptografiseen arkkitehtuuriin” sekä ”hiekkalaatikkomääritykseen”. Heidän on tarkistettava, että kaikki allekirjoitettu koodi on oikein aikaleimattu, jotta tuotteen ennenaikainen vanheneminen estetään. (Oracle Technology Network 2014.)

2.2 Java2EE ja sen tietoturva

J2EE (Java 2 Platform, Enterprise Edition) on Javan alusta, joka on suunniteltu keskustietokone kokoluokan laskentaan suurille yrityksille. Sun Microsystems yhdessä alan yhteistyökumppanien kanssa (kuten IBM), suunnittelivat J2EE:n yksinkertaistamaan ohjelmakehitystä ”thin client” (kevytpääte) –tason ympäristössä. (Rouse 2005d.)

J2EE yksinkertaistaa ohjelmistokehitystä ja vähentää ohjelmoinnin tarvetta ja ohjelmoijien kouluttamisen tarvetta luomalla normit, uudelleen käyttämällä modulaarisia komponentteja ja mahdollistamalla monien ohjelmoinnin näkökohtien käyttämisen automaattisesti. (Rouse 2005d.)

J2EE sisältää Java 2 alustan vakiopainoksen (standard edition) (J2SE) sekä monia komponentteja ja malleja. J2EE arkkitehtuuri koostuu neljästä pääelementistä: (Rouse 2005d.)

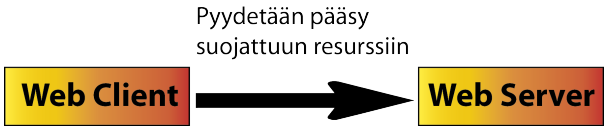
J2EE-Ohjelmointimalli on standardi malli, jota käytetään helpottamaan kevytpäätteiden monitahoista kehitystä. **J2EE-alusta** sisältää tarvittavat linjaukset ja rajapinnat, kuten Java servletit ja Java viestipalvelun (JMS**). J2EE:n-yhteensopivuustestauspaketti** varmistaa, että J2EE tuotteet ovat yhteensopivia alustastandardien kanssa. **J2EE:n-viitetoteutus** selittää J2EE mahdollisuuksia ja tarjoaa sen operatiivisen määritelmän. (Rouse 2005d.)

Yritystason ja web-tason ohjelmat koostuvat komponenteista, jotka ovat levitetty erilaisiin säiliöihin. Nämä komponentit ovat yhdistetty ja niistä on rakennettu monitasoinen yritysohjelma. Komponenttien turvallisuudesta vastaa niiden säiliöt. Säiliö tarjoaa kahdenlaista turvallisuutta: selittävää ja ohjelmallista. Selittävä turvallisuus voidaan jakaa kahteen alakategoriaan (käyttöönottokuvaukset ja huomautukset) riippuen siitä, kumpaa se käyttää. **Käyttöönottokuvaus** on XML-tiedosto, joka on ohjelman ulkopuolinen tiedosto ja jossa ilmaistaan ohjelman turvallisuusrakenne, mukaan lukien turvallisuusroolit, pääsyn valvonta sekä todennusvaatimukset. **Huomautukset**, joita kutsutaan myös metadataksi, käytetään määrittelemään turvallisuustietoa luokkatiedoston sisällä. Kun ohjelma on levitetty, tämä tieto voidaan ottaa käytettäväksi tai se voidaan ylikirjoittaa ohjelman levitysavainsanalla. Huomautuksia käyttämällä ei tarvitse kirjoittaa käyttöönottotietoja XML-avainsanoihin. Sen sijaan, voit laittaa avainsanat koodiin ja haluttu informaatio luodaan sen perusteella. (Oracle 2013, 692.)

Ohjelmallinen turvallisuus on sisällytetty ohjelmaan ja sitä käytetään tekemään turvallisuusratkaisuja. Ohjelmallinen turvallisuus on käyttökelpoinen sellaisessa tilanteessa, jolloin selittävä turvallisuus ei yksinään ole riittävä ilmaisemaan ohjelman turvallisuusmallia. (Oracle 2013, 692.)

Java EE ympäristön turvallisuuskäyttäytymisen voi ymmärtää ehkä paremmin tarkastelemalla, mitä tapahtuu yksinkertaiselle ohjelmalle, jossa on web-asiakas, käyttöliittymä ja ”enterprise bean” (katso liite 2) -yrityslogiikka. Seuraavissa esimerkeissä web-asiakas luottaa, että web-palvelin toimii sen todennusvälipalvelimena keräten käyttäjän todennustiedot asiakkaalta ja käyttäen sitä avatakseen todennetun istunnon. (Oracle 2013, 692.)

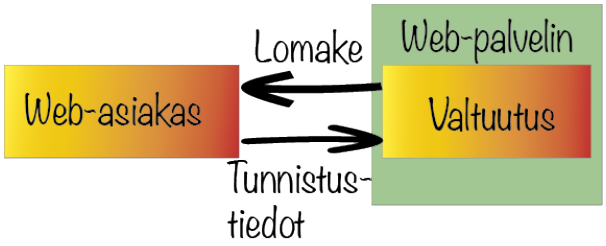
Alustavassa pyynnössä web-asiakas pyytää pääohjelmalta URL-osoitetta. Tämä on kuvattu alla olevassa kuvassa (kuva 1). (Oracle 2013, 692.)



Kuva 1. Pyydetään pääohjelmalta URL-osoitetta (Oracle 2013, 692).

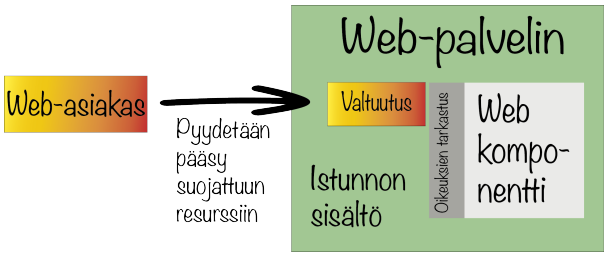
Koska asiakas ei ole vielä tunnistautunut ohjelmaympäristölle palvelin, joka on vastuussa ohjelman web-osuuden toimittamisesta, huomaa tämän ja herättää tarkoituksenmukaisen tunnistusmekanismin tälle resurssille. (Oracle 2013, 693.)

Alustavassa tunnistuksessa web-palvelin palauttaa lomakkeen, jota web-asiakas käyttää keräämään tunnistustietoja, kuten käyttäjänimeä ja salasanaa, käyttäjältä. Web-asiakas ohjaa tunnistustiedot web-palvelimelle, jossa web-palvelin tarkistaa niiden oikeellisuuden. Oikeellisuustunnistusmekanismi voi olla paikallisella palvelimella tai siinä voidaan hyödyntää taustalla olevia turvallisia palveluita. Oikeellisuustunnistuksen perusteella web-palvelin asettaa valtuutuksen käyttäjälle (kuva 2). (Oracle 2013, 693.)



Kuva 2. Alustava tunnistus (Oracle 2013, 693).

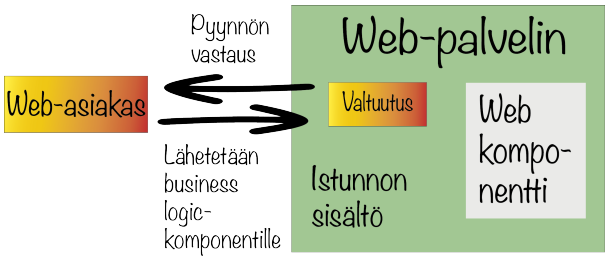
Valtuutuksia käytetään **URL-tunnistuksesta** lähtien määrittämään, onko käyttäjällä pääsyä rajoitettuihin resursseihin, joita hän saattaa pyytää. Web-palvelin konsultoi webresurssiin liitettyä turvallisuuspolitiikkaa selvittääkseen ne turvallisuusroolit, joilla on pääsy tähän resurssiin. Turvallisuuspolitiikka johdetaan joko huomautuksesta tai käyttöönottokuvauksesta. Tämän jälkeen ohjelma testaa käyttäjän valtuutusta jokaista roolia vastaan määrittääkseen, pystyykö se sijoittamaan käyttäjän rooliin (kuva 3). (Oracle 2013, 693.)



Kuva 3. URL-tunnistus (Oracle 2013, 693).

Web-palvelimen arviointi loppuu tulokseen ”on oikeutettu”, eli kun web-palvelin onnistuu sijoittamaan käyttäjän rooliin. ”Ei ole oikeutettu” tulos tulee, mikäli web-palvelin ei kykene sijoittamaan käyttäjää mihinkään sallittuun rooliin. (Oracle 2013, 693.)

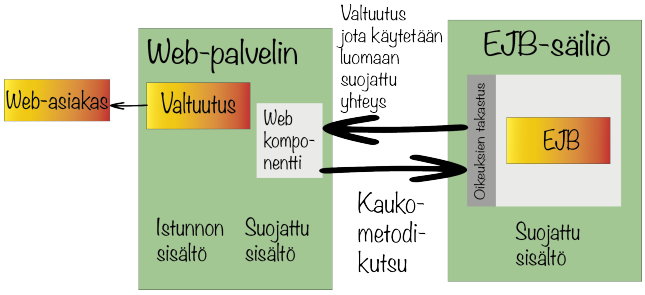
Jos käyttäjä on tunnistettu, web-palvelin palauttaa alkuperäisen URL-pyynnön tuloksen (kuva 4) (Oracle 2013, 693.)



Kuva 4. Alkuperäisen pyynnön täyttö (Oracle 2013, 693).

Yllä olevassa esimerkissä (kuva 4), pyydetty URL-osoite lähetetään vastaukseksi, jolloin käyttäjä voi syöttää tietoa, jota ohjelman ”business logic” –komponentti tarvitsee käsiteltäväkseen. (Oracle 2013, 693.)

Web-sivu suorittaa kaukometodikutsun enterprise beanille käyttäen käyttäjän valtuutusta luodakseen turvallisen yhteyden web-sivun ja enterprise beanin välille. Yhteys on toteutettu kuten kaksi suojattua sisältöä: toinen on web-palvelin ja toinen on EJB-säiliö (kuva 5): (Oracle 2013, 693.)



Kuva 5, Enterprise Bean Business -metodien kutsuminen (Oracle 2013, 693).

2.3 Java Server Page ja sen tietoturva

Java Server Page (JSP) on ohjelmointikieli, jolla kontrolloidaan Web-sivujen sisältöä tai ulkoasujen käyttöä servlettien avulla. Servletit ovat pieniä määrättyjä ohjelmia, jotka ajetaan Web-palvelimella sivujen muokkaamiseksi ennen web-sivun lähettämistä käyttäjälle, joka sitä pyysi. (Rouse 2005e.)

Sun Microsystem, joka kehitti Javan, viittaa JSP-teknologialla Servlet-ohjelman käyttöliittymään (API). JSP on yhteensopiva Microsoftin kehittämään Active Server Page (ASP) -teknologiaan. Siinä, missä Java Server Page kutsuu Java-ohjelmaa, joka suoritetaan Web-palvelimella, ASP sisältää komentosarjan, joka tulkitaan komentosarjan tulkitsijalla (kuten VBScript tai JScript) ennen kuin sivu lähetetään käyttäjälle. Joskus HTML-sivu, joka sisältää linkin Java-Servletiin, nimetään päätteellä .JSP. (Rouse 2005e.)

Java Server Page –tekniikassa, samoin kuin servleteissä, on useita mahdollisia mekanismeja web-kehittäjille luoda turvallisia ohjelmia. Resurssit on suojattu kuvaavasti tunnistamalla ne ohjelman käyttöönottoavainsanalla ja määräämällä niille roolit. (Java Server Pages 2014.)

Käytössä on useita mahdollisia tunnistustasoja, ulottuen perustunnistuksesta, jossa käytetään tunnusta ja salasanaa aina hienostuneeseen tunnistukseen, jossa käytetään varmenteita. (Java Server Pages 2014.)

Servletien määrityksessä käytettävä todentamismekanismitekniikkaa kutsutaan **rooliperusteikseksi** tunnistukseksi. Sen sijaan, että se rajoittaisi resursseja käyttäjätasolla, luodaan rooleja ja rajoitetaan resursseja roolitasolla. Voidaan määritellä erilaisia rooleja tomcat-users.xml-tiedostoon, joka sijaitsee Tomcatin kotihakemistossa olevassa conf-hakemistossa (kuva 6). (Java Server Pages 2014.)

```xml
<?xml version='1.0' encoding='utf-8' ?>
<tomcat-users>
    <role rolename="tomcat"/>
    <role rolename="role1"/>
    <role rolename="manager"/>
    <role rolename="admin"/>

    <user username="tomcat" password="tomcat" roles="tomcat" />
    <user username="role1" password="tomcat" roles="role1" />
    <user username="both" password="tomcat" roles="tomcat,role1" />
    <user username="admin" password="secret" roles="admin,manager" />
</tomcat-users>
```

Kuva 6. Rooliperusteinen jako, tomcat-users.xml (Java Server Pages, 2014).

Yllä olevassa kuvassa on yksinkertaisen kartoituksen nimi, salasana ja rooli. Käyttäjällä saattaa olla useampi rooli, kuten käyttäjälle ”both”, jolla on sekä ”tomcat”, että ”role1” – roolit. (Java Server Pages 2014.)

Tunnistamisen ja määrittämisen jälkeen tietyt roolit, rooliperusteiset turvallisuusrajoitukset voidaan asettaa eri web-ohjelmien resursseille käyttäen **<security-constraint>** -elementtiä web.xml-tiedostossa, joka löytyy WEB-INF-hakemistosta (kuva 7): (Java Server Pages 2014.)

```xml
<web-app>
...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>
                TurvattuKirjaSivusto
            </web-resource-name>
            <url-pattern>/secured/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <description>
                Vain managerit voivat käyttää tätä ohjelmaa
            </description>
            <role-name>manager</role-name>
        </auth-constraint>
    </security-constraint>
    <security-role>
        <role-name>manager</role-name>
    </security-role>
    <login-config>
        <auth-method>BASIC</auth-method>
    </login-config>
...
</web-app>
```

Kuva 7. Security-constraint (Java Server Pages 2014).

Edellisellä sivulla olevasta kuvasta nähdään, että sekä GET, että POST –pyynnöt osoitteeseen /secured/* turvataan automaattisesti. Henkilö, jolla on manager-rooli, on oikeus päästä näihin turvattuihin resursseihin. Lopuksi, login-config elementtiä käytetään kuvaamaan BASIC-nimistä lomaketunnistusta. (Java Server Pages 2014.)

Jos käyttäjä yrittää avata mitä tahansa web-sivua **/security**-polusta, hän saa keskusteluikkunan, jossa häneltä pyydetään käyttäjätunnusta ja salasanaa. Jos hän syöttää käyttäjän ”admin” ja salasanan ”secrer” (ovat tietokannassa), niin hän pääsee **/secured/*** -URL-osoitteisiin, koska edellisellä sivulla olevassa kuvassa määriteltiin, että käyttäjä admin, jolla on manager-rooli, on pääsy tähän resurssiin. (Java Server Pages 2014.)

Kun käyttää **lomaketunnistusmetodia**, täytyy käyttäjälle tarjota kirjautumislomake, jossa kysytään käyttäjänimeä ja salasanaa. Alla on yksinkertainen esimerkki login.jsp-koodista, joka tekee juuri tämän (kuva 8): (Java Server Pages 2014.)

```
<html>
<body bgcolor="#ffffff">
    <form method="POST" action="j_security_check">
        <table border="0">
            <tr>
                <td>Käyttäjätunnus</td>
                <td><input type="text" name="j_username"></td>
            </tr>
            <tr>
                <td>Salasana</td>
                <td><input type="text" name="j_password"></td>
            </tr>
        </table>
        <input type="submit" value="Kirjaudu!">
    </form>
</body>
</html>
```

Kuva 8. Login.jsp –kirjautumislomake (Java Server Pages 2014).

Koodin kirjoittajan tulee varmistaa, että kirjautumislomake sisältää elementit **j_username** ja **j_password** sekä action metodi sisältää elementin **j_security_check** (nimet voivat olla muitakin, kirjoittajan huomautus.). Lomakkeen metodina täytyy käyttää POST-metodia. Samalla koodin kirjoittajan tulee muokata **<login-config>**-tagia määrittääkseen FORM:lle auth-method:in (kuva 9): (Java Server Pages 2014.)

```
<web-app>
...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>
                TurvattuKirjaSivusto
            </web-resource-name>
            <url-pattern>/secured/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <description>
                Vain managerit voivat käyttää tätä ohjelmaa
            </description>
            <role-name>manager</role-name>
        </auth-constraint>
    </security-constraint>
    <security-role>
        <role-name>manager</role-name>
    </security-role>
    <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/login.jsp</form-login-page>
            <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
    </login-config>
...
</web-app>
```

Kuva 9. Auth-metodi (Java Server Pages 2014).

Kun käyttäjä yrittää päästä mihin tahansa sivustoon **/secured/***-hakemistossa, hän saa eteensä lomakkeen, jossa häneltä kysytään käyttäjätunnusta ja salasanaa. Kun ohjelma huomaa **j_secure_check** –toiminnon, se käyttää sisäistä mekanismiä käyttäjän tunnistamiseen. (Java Server Pages 2014.)

Jos kirjautuminen onnistuu ja käyttäjä tunnistetaan käyttämään turvattuja resursseja, ohjelma käyttää istunto-kohtaista tunnistetta (session-id) tunnistamaan käyttäjän kirjautuminen tästä hetkestä lähtien. Ohjelma ylläpitää istuntokohtaista kirjautumista keksin avulla, joka sisältää em. istunto-kohtaisen tunnisteen. Palvelin lähettää keksin asiakastietokoneelle ja niin kauan, kun käyttäjä esittää tämän keksin peräkkäisillä kutsuilla, ohjelma tietää, kenestä on kysymys. Mikäli kirjautuminen epäonnistuu, palvelin lähettää käyttäjän error.jsp –sivulle. (Java Server Pages 2014.)

Tässä **j_security_check** on toiminta, jota ohjelma käyttää määrittämään lomakeperusteisen kirjautumisen kirjautumislomakkeelle. Samalle lomakkeelle ohjelmoijan on syytä sisällyttää tekstin syöttökontrolli **j_username** sekä salasanan syöttökontrolli **j_password**. Tämä tarkoittaa, että tiedot lähetetään palvelimelle, joka tarkistaa tiedot. Miten palvelin sen tekee, on palvelinkohtaista. (Java Server Pages 2014.)

Java Servlet Page tarjoaa **ohjelmallista turvallisuutta** esimerkiksi seuraavasti: HttpServletRequest –objekti tarjoaa seuraavia metodeja, joita voidaan käyttää ajonaikaisen turvallisuuden kaivamiseen. (String) **getAuthType()** –metodi palauttaa merkkijono-olion, joka edustaa tunnistamismallin nimeä, jota käytetään suojaamaan servletiä. (Boolean) **isUserInRole** (java.lang.String role) –metodi palauttaa totuusarvomuuttuja-arvon: tosi, jos käyttäjä kuuluu annettuun roolijoukkoon, epätosi, jos ei kuulu (kuva 10). String **getProtocol()** – metodi palauttaa merkkijono-olion, joka edustaa protokollaa, jota käytetään lähettämään pyyntö (request). Tätä arvoa voidaan käyttää tarkistamaan, käytetäänkö turvallista protokollaa. boolean **isSecure()** –palauttaa totuusarvomuuttuja-arvon, joka tarkistaa, käytettiinkö pyyntöön HTTPS-protokollaa. Arvo on tosi, jos yhteys on suojattu, epätosi, mikäli ei ole. Principle **getUserPrinciple()** – palauttaa java.security.Principle-olion, joka sisältää tämän hetkisen tunnistetun käyttäjän nimen. (Java Server Pages 2014.)

```
<% if (request.isUserInRose("manager")) { %>
    <a href="managers/mgrreport.jsp">Manageri Raportti</a>
    <a href="managers/personnel.jsp">Henkilökuntatiedot</a>
</%>
```

Kuva 10. Esimerkki, jossa Java Server Pages linkitetään manager-rooliin (Java Server Pages 2014).

Kun käyttäjän rooli tarkistetaan JSP:ssä tai servletissä, voidaan muokata Web-sivut näyttämään käyttäjälle ainoastaan ne osiot, joihin hänellä on pääsy. Jos tähän tarvitaan käyttäjänimeä siinä muodossa, kun se syötettiin lomakkeelle, se saadaan getRemoteUser-metodilla. (Java Server Pages 2014.)

2.4 JavaScript ja sen tietoturva

JavaScript on ohjelmointikieli, jota käytetään, jotta web-sivuista saataisiin interaktiivisia. JavaScipt ajetaan käyttäjän tietokoneella. Yleensä sitä käytetään, kun luodaan kyselyitä ja mielipidemittauksia. JavaScriptiä ei pidä sekoittaa Javaan, joka on nimestään huolimatta eri ohjelmointikieli. (Chapman 2015.)

JavaScript-tuki on rakennettu kaikkiin isoimpiin selaimiin, mukaan lukien Internet Explorer, Firefox, Crome ja Safari. Sen lisäksi, että käyttäjillä pitää olla käytössään JavaScriptiä

tukeva selain, pitää tuen oletuksena olla päällä, jotta käyttäjät voivat ajaa JavaScriptiä selaimissaan. (Chapman 2015.)
JavaScriptiin on valmiiksi kirjoitettuja koodeja. Näitä löytyy netistä tuhansittain ja käyttäjä voi poimia haluamansa ja hyödyntää omalla sivullaan. Toki yleinen ymmärrys koodinkirjoittamisesta on hyvä olla, jotta ymmärtää koodin luonteen ja merkityksen sekä osaa hyödyntää sitä. (Chapman 2015.)

JavaScript on tulkittu koodi. Mitään erityistä kääntäjää ei tarvita, jotta saataisiin toimiva koodi. Sitä voi kirjoittaa millä tahansa tekstieditorilla, joskin suositellaan käyttämään ”puhdasta” tekstieditoria. Tämä siksi, että se ei lisää omia koodejaan tekstin joukkoon, kuten esimerkiksi Word tekee. Toki editorit, jotka värittävät erityyppiset tekstit (komennot, muuttujat) eri värillä, helpottavat kirjoitustyötä entisestään. Hyviä tällaisia ovat muun muassa Gvim ja NotePad+. (Chapman 2015.)

JavaScriptin voi kirjoittaa joko HTML-tekstin joukkoon tai omaksi tiedostokseen. Mikäli samaa koodia on tarve käyttää useammassa paikassa, on suositeltavaa kirjoittaa koodi omaksi tiedostokseen. Tähän tiedostoon voi viitata HTML-koodissa. (Chapman 2015.)

JavaScriptin ilmestyttyä, on ollut useita JavaScriptin turvallisuusongelmia, jotka ovat saavuttaneet laajamittaista huomiota. Eräs tällainen on se tapa, jolla JavaScript keskustelee dokumenttioliomallin kanssa. Se muodostaa loppukäyttäjille riskin, joka sallii ilkeämielisten toimijoiden jakaa komentosarjoja internetin yli ja ajaa niitä käyttäjän koneella. (Glynn 2015.)

On kuitenkin kaksi keinoa, jolla pystytään estämään tämän kaltaisia JavaScript turvallisuusriskejä. Ensimmäinen on ajaa ohjelmia ”hiekkalaatikossa” tai ajaa komentosarjoja erikseen siten, että niillä on pääsy vain tiettyihin resursseihin ja että ne suorittavat vain tietyn suoritteen. Toinen tapa on toteuttaa ’saman alkuperän’ -politiikka, joka estää komentosarjojen pääsyn yhdestä sijainnista tietoon, jota käyttää toiset komentosarjat käyttävät toisissa sijainneissa. Monet JavaScript tietoturvaongelmat johtuvat siitä, ettei selain ohjelmoijat ota edellä mainittuja toimia huomioon ohjelmointityössään, kun miettivät DOM-pohjaisia JavaScript turvallisuusriskejä. (Glynn 2015.)

Yksi yleisimmistä JavaScript-turvallisuusriskeistä on poikittaissivukoodaus. Poikittaissivukoodaus riskit mahdollistavat hyökkääjien muokata nettisivustoa siten, että se palauttaa ilkeämielistä komentosarjaa käyttäjille. Nämä ilkeämieliset komentosarjat antavat tämän jälkeen hyökkääjälle pääsyn käyttäjän koneelle. Poikittaissivukoodaus

turvallisuusongelmat voivat esiintyä silloin, kun selain tai ohjelman kirjoittajat eivät toteuta saman alkuperän politiikkaa. Jos XSS riskejä ei oikaista, ne saattavat aiheuttaa käyttäjän tietojen varastamisen, käyttäjän tilin peukaloimisen, haittaohjelmien leviämisen sekä käyttäjän koneen kauko-ohjaamisen selaimen avulla. (Glynn 2015.)

Toinen yleinen JavaScript-turvallisuusriski on poikittaissivupyyntöväärennys (CSRF). Poikittaissivuväärennys riski sallii hyökkääjän muokata uhrinsa selainta ottamaan ei-tarkoitettuja toimia muilta websivustoilta. Tämä on mahdollista, kun kohdesivuston tunnistuspyynnöt riippuu yksinomaan kekseistä ja hyökkääjä onnistuu lähettämään pyynnön, jossa on mukana käyttäjän keksi. Tämä JavaScript-turvallisuusongelma voi johtaa tilin peukalointiin, tietojen varastamiseen, petokseen, ja moneen muuhun. Sekä XSS että CSRF sijaitsevat ohjelmakerroksessa ja vaativat oikein kehitettyjen tekniikoiden seuraamista, jotta ne voidaan välttää. (Glynn 2015.)

On joukko muita yleisiä JavaScript-turvallisuusongelmia, jotka saattavat lisätä käyttäjien riskejä. Näitä ovat muun muassa väärä asiakaspalvelin-luottosuhde, ongelmat selaimessa ja selaimen liitännäiskoodissa, hiekkalaatikon tai saman alkuperän politiikan vääränlainen käyttö. Ainut tapa organisaatioille välttyä tällaisilta JavaScript-turvallisuusriskeiltä on kehittää ja hankkia ohjelmia, jotka ovat vapaita JavaScript-turvallisuusriskeistä. Monet organisaatiot käyttäjät JavaScript-turvallisuusanalysaattoreita testatakseen ja oikaistakseen näitä ongelmia. (Glynn 2015.)

JavaScript turvallisuusanalysaattorit ovat JavaScript turvallisuustyökaluja, jotka suorittavat ohjelmakoodianalyysia asiakaspuolen ohjelmissa. Nämä analysaattorit voivat tyypillisesti testata JavaScript turvallisuusongelmia, toimeenpano-ongelmia, asetusvikoja ja muita riskejä, joita hyökkääjät voivat käyttää hyväkseen. (Glynn 2015.)

2.5 MariaDB ja sen tietoturva

MariaDB pyrkii olemaan looginen vaihtoehto tietokanta—ammattilaisille, jotka etsivät vahvaa, skaalautuvaa ja luotettavaa SQL-palvelinta. Tämän toteuttamiseksi, MariaDB-säätiö tekee tiivistä yhteistyötä käyttäjien ja kehittäjien laajan yhteisön kanssa vapaan ja avoimen lähdekoodin ohjelmistojen hengessä. MariaDB:n julkaisutapa tasapainottaa ennustettavuuden luotettavuuden kanssa. (MariaDB Foundation 2015.)

MariaDB on parannettu, drop-in korvaaja MySQL:lle ja on käytettävissä GPL v2 – lisenssillä. Sen on kehittänyt MariaDB-yhteisö MariaDB-säätiön toimiessa pääasiallisena rahoittajana. (MariaDB Foundation 2015.)

MariaDB:n kehittäjille turvallisuus on erityinen painopiste. Projekti ylläpitää sen omia turvallisuuspaikkaustiedostoja MySQL:n paikkaustiedostojen päälle. Jokaiseen MariaDB julkaisuun kehittäjät liittävät MySQL turvallisuuspaikkaustiedostot ja kehittävät niitä, mikäli sille on tarvetta. Kun kriittisiä turvallisuusongelmia löydetään, kehittäjät valmistelevat ja jakavat välittömästi uuden MariaDB julkaisun saadakseen korjauksen jakoon niin pian kuin mahdollista. (MariaDB 2010.)

Monet turvallisuusongelmat, jotka ovat löytyneet MySQL:stä ja MariaDB:stä, ovat löytyneet ja raportoitu MariaDB tiimin toimesta. MariaDB tiimi toimii läheisesti CVE:n (katso liite 2) pitääkseen huolen siitä, että kaikki turvallisuusongelmat tulevat asianmukaisesti raportoitua ja selitettyä riittävän yksityiskohtaisesti. Nämä yksityiskohdat julkaistaan yleensä vasta sen jälkeen kun korjaustiedostot MariaDB:lle ja MySQL:lle on julkaistu. (MariaDB 2010.)

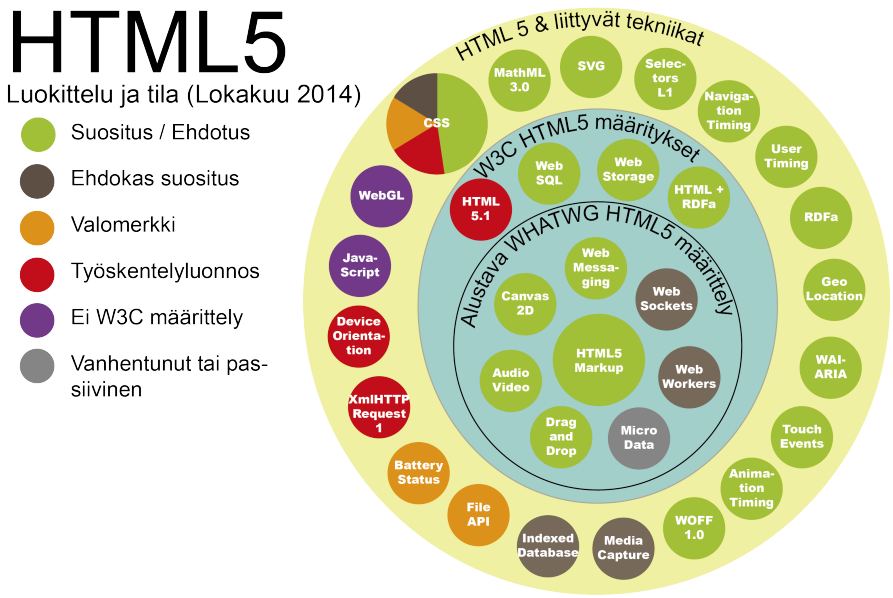
2.6 HTML5 ja sen tietoturva

HTML 5 on uudistettu versio Hypertext Markup Language (HTML) standardi ohjelmointikielestä, joka kuvaa verkkosivujen sisältöä ja ulkoasua. HTML5 on kehitetty ratkaisemaan yhteensopivuusongelmia, jotka vaikuttavat nykyiseen standardiin, HTML4:een. (Rouse 2014a.)

Yksi suurimmista eroista HTML5:n ja aikaisempien versioiden välillä on, että vanhemmat versiot vaativat omat pluginsa ja rajapintansa. Tämän vuoksi websivu, joka on rakennettu ja testattu yhdellä selaimella, ei lataudu oikein toisella selaimella. HTML5 tarjoaa yhden yhteisen rajapinnan, joka tekee elementtien lataamisen helpommaksi. Esimerkiksi enää ei tarvitse asentaa Flash-laajennusta erikseen, koska HTML5 elementti hoitaa sen itse. (Rouse 2014a.)

Yksi HTML5:n suunnittelun tavoitteista on tukea multimediaa mobiililaitteissa. Uudet syntaktiset ominaisuudet esiteltiin tukemaan tätä, kuten video-, audio- ja canvas-tagit. HTML5 esitteli myös uusia ominaisuuksia, jotka voivat muuttaa käyttäjien tapaa kommunikoida dokumenttien kanssa. Näitä ovat muun muassa uudet jäsentelysäännöt, jotka parantavat joustavuutta, uudet määritteet, vanhanaikaisten ja tarpeettomien ominaisuuksien poistaminen, ”vedä ja pudota” –valmius HTML5 asiakirjasta toiseen, offline –editointi, viestintäparannukset, yksityiskohtaiset säännöt jäsentämiseen, MIME ja protokollakäsittelijän rekisteröinti sekä yhteinen standardi tiedon tallentamiseksi SQL-tietokantoihin (Web SQL). (Rouse 2014a.)

World Wide Web Consortium (W3C) hyväksyi HTML5:n vuonna 2007. Tämä ryhmä julkaisi ensimmäisen luonnoksen HTML5:stä tammikuussa 2008. Tällä hetkellä,HTML5 on ”Call for Review” tilassa ja W3C odottaa, että se saavuttaa lopullisen tilan vuoden 2014 loppuun mennessä (kuva 11) (Tämän hetkinen tila on 28.10.2014 suositus eli tila ei ole vielä lopullinen – tarkistettu 4.6.2015). (Rouse 2014a.)



Kuva 11. HTML5 ja siihen liittyvät teknologiat (Mavrody 2012).

HTML5:n ja varsinkin vanhempien HTML-versioiden kautta pystyy tekemään hyökkäyksiä asiakastietokoneelle. Näistä ehkä tunnetuin on XSS-hyökkäys (katso liite 2). HTML5:ttä kehitettäessä on kuitenkin painopiste ollut myös tietoturvassa ja HTML5 kehittäjiä varten on tehty muun muassa niin sanottu ”lunttisivu” (HTML5 Security Cheat Sheet), jonka neuvoja noudattamalla he pystyvät tekemään sivuistaan tietoturvalliset.

HTML5 tarjoaa erilaisia **yhteysrajapintoja** (Internetviestintä, Ristikkäisalkuperäresurssien jako sekä Websocket-protokolla). *Internet viestintä*, joka tunnetaan myös nimellä ristikkäistoimialueviestintä, tarjoaa tavat viestimiseen dokumenttien välillä, joilla on eri alkuperä, tavalla joka on yleensä turvallisempi kuin useimmat ohjelmakoodit, joita käytettiin aikaisemmin toteuttamaan tämä sama toimenpide. Kuitenkin on hyvä pitää muutamia suosituksia mielessä. (Kotiwitcz & Lara, Roxberry, Shah, Stranathan 2014.)

Kun viesti lähetetään, on syytä merkitä selvästi odotettu alkuperä toiseksi perusteeksi **postMessage** –kenttiin sen sijaan, että käytettäisiin *-merkkiä, jolla estettäisiin lähettämästä viestiä tuntemattomaan alkuperään uudelleenohjauksen tai jonkin muun syyn, missä kohde ikkunan alkuperä muuttuu, jälkeen. Vastaanottavan sivun pitäisi aina

tarkistaa lähettäjän alkuperäominaisuus todentaakseen, että tieto on peräisin oletetusta lähteestä sekä suorittaa syöttökelpuutus tapahtuman tieto-ominaisuudelle varmistaakseen, että se on halutussa muodossa. (Kotiwitcz ym. 2014.)
Myöskään ei pidä olettaa, että kontrolloi tiedon ominaisuuksia. Yksittäinen poikittaissivustokoodausvirhe (XSS flaw) lähettävällä sivulla antaa hyökkääjälle mahdollisuuden lähettää viestejä missä muodossa tahansa. Molempia sivuja pitäisi tulkita vain vaihdettujen viestien tietona. Koskaan ei pidä arvioida annettuja viestijä koodina (esimerkiksi **eval()**-komennon avulla) tai syöttää sitä sivun dokumenttioliomalliin (DOM) (esimerkiksi **innerHTML**-komennon avulla), koska se voi luoda DOM-pohjaisen XSS-haavoittuvuuden. Samoin, sen sijaan, että käytettäisiin turvatonta metodia tiedon arvon syöttämiseen elementille, kuten **element.innerHTML = data**; voidaan käyttää turvallisempaa vaihtoehtoa: **element.textContent = data;** (Kotiwitcz ym. 2014.)

Tarkistetaan alkuperä kunnolla täysin vastaamaan täydellistä toimialuenimeä (FQDN), jota odotetaan. Seuraava esimerkkikoodi: **if(message.orgin.indexOf(".owasp.org")!=-1) { /* ... */ }** on hyvin turvaton eikä se käyttäydy halutulla tavalla, sillä www.owasp.org.attacker.com pääsen sen läpi. Jos halutaan sisällyttää ulkoista sisältöä tai epäluotettavia vempaimia ja sallitaan käyttäjä-valvontaisia koodeja (joita ei missään nimessä suositella), kannattaa miettiä JavaScript uudelleenkirjoituskehystä, kuten Google Cajaa tai käyttää hiekkalaatikkokehyksiä. (Kotiwitcz ym. 2014.)

Ristikkäisalkuperäresurssien jaossa tarkistetaan niiden osoitteiden oikeellisuus, jotka tulevat **XMLLHttpRequest.open**-komennon kautta. Nykyiset selaimet sallivat tällaisten osoitteiden olevan ristikkäistoimialueilla: tällainen käytös voi johtaa ulkopuolisen hyökkääjän XSS-hyökkäykseen. On erityisesti kiinnitettävä huomiota täydellisiin toimialuenimiin. On varmistettava, että osoitteet vastaavat komentoon: **Access-Control-Allow-Origin: *.** On myös pidettävä huolta, ettei liitetä mitään arkaluonteista sisältöä tai tietoa, joka saattaa auttaa hyökkääjää tulevissa hyökkäyksissään. Käytetään edellä mainitun komennon otsikkoa ainoastaan valituissa osoitteissa, joiden tarvitsee päästä ristikkäistoimialueille. Ei siis käytetä otsikkoa koko toimialueella. (Kotiwitcz ym. 2014.)

Sallitaan vain valitut, luotetut palvelimet **Access-Control-Allow-Origin** –listaan. Suositaan sallittujen palvelimien listaa (whitelist) kiellettyjen palvelimien listan (blacklist) tai sen sijasta, että sallit kaikki palvelimet. Pidetään mielessä, että ristikkäisalkuperäresurssijako (Cross-origin resource sharing – CORS) ei estä pyydetyn tiedon menemistä todentamattomaan paikkaan. On silti tärkeää, että palvelin suorittaa tavallisen CSRF tarkistuksen. (Kotiwitcz ym. 2014.)

Vaikka Request for Comment (RFC – eräänlainen internetstandardi dokumenteille, katso liite 2) suosittelee ennen käyttöä tarkastusta OPTIONS verbille, eivät nykyiset sovellukset ehkä kuitenkaan toteuta tätä suositusta, joten on erittäin tärkeää suorittaa tavallisissa POST ja GET metodeissa tämä tarkastus. Ei luoteta pelkästään alkuperäiseen otsikkoon Access Control –tarkistuksissa. Selain lähettää aina tämän otsikon CORS pyyntöinä, mutta ulkopuolinen selain on saattanut huijata sitä. Arkaluonteinen tieto tulee suojata ohjelmatasoisella protokollalla. (Kotiwitcz ym. 2014.)

Websocket-protokolla tarkistuksissa tiputetaan taaksepäin yhteensopivuus toteutetuissa asiakas/palvelin-suhteissa ja käytetään ainoastaan protokollaversioita, jotka ovat hybi-00 -tasoa ja sen jälkeen. Suosittu Hixie-76 (hiby-00) ja vanhemmat ovat vanhentuneita ja turvattomia. Suositellut versiot tuetuista selaimista on vimeisimmät versiot, jotka löytyvät RFC 6455:stä (Websecket Protocol), näitä ovat Firefox 11+, Chrome 16+, Safari 6, Opera 12.50 ja IE10. Vaikka on varsin helppoa tunneloida TCP palveluita WebSocket protokollien läpi (esim. VNC, FTP), samalla mahdollistetaan pääsy näihin tunneloituihin palveluihin selaimessa olevalle hyökkääjälle esim. XSS-hyökkäykselle. Näitä palveluita voidaan kutsua myös suoraan ilkeämielisillä sivuilla tai ohjelmilla. (Kotiwitcz ym. 2014.)

Protokolla itsessään ei käsittele tunnistusta ja/tai valtuutusta. Ohjelmatason protokollien pitäisi käsitellä niitä erikseen niissä tapauksissa, jossa luottamuksellista tietoa siirretään. Käsitellään websocketien kautta saatua tietoa, kuten dataa: Ei yritetä suoraan siirtää sitä dokumenttioliomalliin eikä tarkistaa sitä ohjelmakoodina. Jos vastaus on JavaScript oliomerkintätapa (JavaScript Object Notation / JSON), ei koskaan käytetä turvatonta **eval();** funktiota, vaan käytetään sen sijaan **JSON.parse()**-funktiota. Välistävetohyökkäykseen voi parhaiten suojautua käyttämällä suojattua protokolla (SSL/TSL). Eli wss:// sen sijaan, että käyttäisi suojaamatonta ws://. (Kotiwitcz ym. 2014.)

Asiakkaan huijaaminen on mahdollista selaimen ulkopuolelta, joten Websocket palvelimen pitäisi pystyä käsittelemään väärä/ilkeä syöte. Tarkistetaan tuleva syöte aina ulkopuolisella sivustolla, koska syötettä on saatettu muuttaa. Koska Websocket asiakas selaimessa on saavutettavissa JavaScript kutsuilla, kaikki Websocket viestintä voidaan huijata tai kaapata XSS-kautta. Siksi on aina tarkistettava sisääntuleva data, joka tulee WebSocket viestinnän kautta. (Kotiwitcz ym. 2014.)

Kaikki URL-osoitteet on vahvistettava hyväksytysti **EventSource** konstruktorissa, vaikka vain saman alkuperän URL-osoitteet ovat sallittuja. Kuten jo edellä mainittiin, kaikki viestit (event.messages) käsitellään tietona, eikä HTML- tai script –koodina. Myös kaikkien viestien attribuuttien alkuperä (event.origin) tulee tarkastaa sen varmistamiseksi, että

viesti tulee luotetulta palvelimelta. Käytetään ”sallittujen palveliminen lista” –lähestymistapaa. (Kotiwitcz ym. 2014.)

HTML5 tarjoaa myös **tallennusrajapintoja** (paikallinen tallennus, indexedDB) *Paikallisissa tallennuksissa* kyseiset tallennusmekanismit vaihtelevat käyttäjä agentilta (katso liite 2) toiselle. Toisin sanoen, vaikka ohjelma vaatisikin todentamista, voi käyttäjä omilla paikallisilla oikeuksillaan ohittaa nuo suojaukset sillä koneella, mille tieto on tallennettu. Siksi suositellaan, ettei mitään arkaluonteista tietoa tallenneta paikallisille tallentimille. On käytettävä **localStorage** -objektin sijasta **sessionStorage** –objektia silloin, kun pysyvää tallennustilaa ei tarvitse käyttää. On huomioitava, että **sessionStorage** on voimassa ainoastaan niin kauan, kun kyseinen ikkuna / välilehti on auki. Yksittäistä XSS:ää voidaan käyttää varastamaan kaikki tieto tällaisilta objekteilta, joten, kuten edellä mainittiin, ei ole syytä tallentaa mitään arkaluonteista tietoa paikalliselle tallentimelle. (Kotiwitcz ym. 2014.)

On erityisesti huomioitava **localStorage.getItem** ja **setItem** kutsut, jotka tulee HTML5-sivulta. Nämä on helppo havaita silloin, kun kehittäjä ohjelmoi ratkaisun, jossa tietoa tallennetaan paikalliselle tallentimelle, joka, kuten jo yllä mainittiin, on huono käytäntö. Koskaan ei saa tallentaa istunnon tunnisteita paikalliselle tallentimelle, koska tieto on JavaScriptin saatavilla. Tätä riskiä voi pienentää keksien avulla käyttäen **httpOnly** lippua. (Kotiwitcz ym. 2014.)

Koska objektien näkyvyyttä ei voi mitenkään rajata tietyssä polussa, kuten esimerkiksi **HTTP Cookies** attribuutin kanssa, kaikki objektit jakavat saman alkuperän (katso edellä) ja ovat suojattuja saman alkuperän politiikalla. Tämä saattaa aiheuttaa tietoturvariskin, joten kannattaa välttää sijoittamasta useita ohjelmia samaan paikkaan, vaan vaikka kaikki jakavat saman paikallisen tallentimen, kannattaa käyttää alipalvelimia (esimerkiksi palvelin1.yritys.com). (Kotiwitcz ym. 2014.)

Marraskuussa 2010, World Wide Web Consortium (W3C) (katso liite 2) julkaisi internet SQL tietokannan (relaatio SQL-tietokanta) vanhentuneeksi määritteeksi. Uutta stardardia, nimeltä Indexed Database Rajapinta tai *IndexedDB*, kehitetään aktiivisesti, ja se tarjoaa avain/arvo tietokantatallennustilan, sekä tavat, jolla voidaan suorittaa kehittyneitä kyselyitä. Tallennusmekanismit vaihtelevat käyttäjä agentilta toiselle. Toisin sanoen, vaikka ohjelma vaatisikin todentamista, voi käyttäjä omilla paikallisilla oikeuksillaan ohittaa nuo suojaukset sillä koneella, mille tieto on tallennettu. Siksi suositellaan, ettei mitään arkaluonteista tietoa tallenneta paikallisille tallentimille. (Kotiwitcz ym. 2014.)

Jos nettitietokantaa hyödynnetään, sisältö asiakaspuolelta on altis XSS-hyökkäyksille ja siksi tarvitsee oikeanlaisen vahvistamisen ja parametrisoinnin. Kuten paikallisen tallennustilankin kanssa, XSS-hyökkäyksiä voidaan käyttää lataamaan ilkeämielistä tietoa verkon tietokantoihin. Näissä olevia tietoja ei kannata pitää luotettavina. (Kotiwitcz ym. 2014.)

Maantieteellinen RFC suositteleekin, että käyttäjäagentti kysyy käyttäjän lupaa, ennen kuin määrittelee (laskee) sijainnin. Kuinka tämä päätös muistetaan, tai muistetaanko sitä lainkaan, riippuu täysin selaimesta. Jotkut käyttäjäagentit vaativat käyttäjää käymään sivulla uudestaan poistaakseen päältä sen, että kysytään käyttäjän lupaa sijainnin määrittämiseen, tämä yksityisyyssyistä, joten on suositeltavaa pyytää käyttäjän syötettä ennenkuin kutsutaan **getCurrentPosition** tai **watchPosition** –metodia. (Kotiwitcz ym. 2014.)

Web Workereiden (katso liite 2) sallitaan käyttää XMLHttpRequest objektia suorittaakseen palvelimen sisällä ristikkäisalkuperäresurssijakopyyntöjä. Vaikka Web Workereillä ei ole pääsyä kutsuvan sivun dokumenttioliomalliin, ilkeät Web Workerit voivat käyttää huomattavasti prosessoritehoa laskemiseen, johtaa palvelinestohyökkäystilaan tai väärinkäyttää ristikkäisalkuperäresurssijakoja tuleviin hyväksikäyttöihin. Siksi tulee varmistaa ohjelmasta, ettei yksikään Web Worker ole pahansuopa. Ei myöskään pidä antaa mahdollisuutta siihen, että käyttäjällä olisi mahdollisuus syöttää Web Worker koodeja. Kaikki Web Workersien vaihtamat viestit tulee vahvistaa. Ei myöskään tule yrittää tarkastaa osia JavaScriptistä esimerkiksi **eval();** -komennolla, koska se saattaa johtaa XSS-haavoittuvuuteen. (Kotiwitcz ym. 2014.)

Kun käytetään tägia epäluotetun sisällön kanssa, on käytettävä **hiekkalaatikko-attribuuttia**, sillä hiekkalaatikko-attribuutti mahdollistaa sisällön rajoittamisen iframe:n sisällä. Kun hiekkalaatikko-attribuutti asetetaan, seuraavat rajoitukset ovat aktiivisia: (Kotiwitcz ym. 2014.)

1. Kaikki merkinnät (markup) käsitellään, kuten ne olisivat ainutlaatuisestä lähteestä
2. Kaikki lomakkeet ja komentosarjat ovat kytketty pois päältä
3. Estetään kaikki linkit kohdentamasta muita selailu sisältöjä
4. Kaikki automaattisesti käynnistyvät ominaisuudet ja liitännäiset ovat poistettu käytöstä

Pyytääkö käyttäjäagentti lupaa käyttäjältä tiedon tallentamiseen yhteydettömään selailuun ja poistetaanko välimuisti riippuu selaimesta. Välimuistin ”myrkytys” on ongelma, mikäli käyttäjä on yhteydessä suojaamattomaan verkkoon, joten yksityisyyssyistä ei ole

suositeltavaa vaatia käyttäjän syötettä, ennen kuin lähetetään minkäänlaista **manifest** tiedostoa. Käyttäjien tulee käyttää välimuistia vain luotettujen verkkosivustojen kanssa ja tyhjentää välimuisti selailun lopuksi tai jos menevät suojaamattomaan verkkoon. (Kotiwitcz ym. 2014.)

Paras käytäntö on määrittää selaimen tukemat kapasiteetit ja suurentaa niitä jonkinlaisilla sijaisilla tai kapasiteeteilla, joita ei suoraan tueta. Tämä saattaa merkitä sipulityyppistä osasta, esimerkiksi Flash Playerin ohittamista, mikäli <video> tagia ei tueta. (Kotiwitcz ym. 2014.)

HTML5 tarjoaa myös erilaisia **HTTP-otsikoita** parantamaan tietoturvaa (X-Frame, X-XSS –suojaus, tiukka siirtoturvallisuus, sisältöturvallisuuspolitiikka sekä alkuperä). *X-Frame* vaihtoehtoa voidaan käyttää estämään ClickJacking moderneissa selaimissa. **'Saman alkuperän '** -attribuuttia voidaan käyttää sallimaan tai estämään samasta lähteestä olevat URL-osoitteet. Esimerkiksi: **X-Frame-Options: DENY**. (Kotiwitcz ym. 2014.)

X-XSS -suojaus on laitettava päälle (toimii ainoastaan Reflected XSS:n kanssa – katso liite 2). Esimerkki: **X-XSS-Protection: 1; mode=block**. (Kotiwitcz ym. 2014.)

Tiukka siirtoturvallisuus on pakotettava jokaisen selaimen käyttävän TLS/SSL:ää pyyntöjen lähettämiseen (tämä voi estää SSL strip hyökkäyksen). Käytettävä **includeSubDomains**-komentoa. Esimerkiksi: **Strickt-Transport-Security: max-age: 8640000; includeSubDomains**. (Kotiwitcz ym. 2014.)

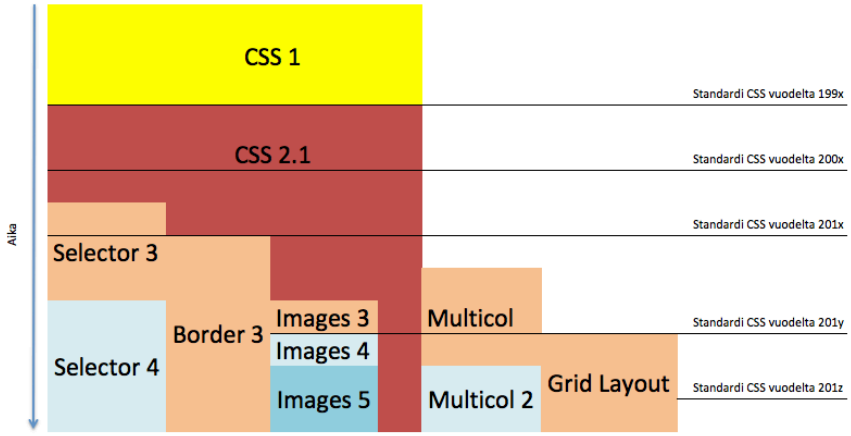
Sisältöturvallisuuspolitiikka määrittelee joukon sisältörajoituksia nettiresurssille, jonka tarkoitus on lievittää nettiohjelmien haavoittuvuuksia kuten poikittaissivustokoodausta. Esimerkiksi: **X-Content-security-Policy: allow 'self';img-src *;object-src media.example.com; script-src js.example.com**. (Kotiwitcz ym. 2014.)

Alkuperä (origin) on otsikko, jonka CORS ja WebSockets pyynnöt lähettävät. Tätä otsikkoa suositellaan käytettäväksi vähentämään CSRF hyökkäyksiä, mutta sitä ei ole vielä otettu käyttöön selainten valmistajien osalta tässä tarkoituksessa. (Kotiwitcz ym. 2014.) Esimerkiksi: **<?php if($_SERVER['HTTP_ORIGIN'] == "http://www.andlabs.org") { ?>**

2.7 CSS3

Cascading Style Sheet (CSS) on tyylitiedosto websivustolle, joka voi koostua useista eri sivuista. Kuten HTML, CSS:kin on kehittynyt vuosien varrella ja nykyinen versio on 3. CSS -tiedostoja käytetään web-sivujen muotoilemiseen. Niillä voidaan muotoilla tekstityylejä, taulukon kokoja ja muita web-sivun määritteitä, joita HTML-komennoilla ei pystynyt tekemään. (Rouse 2005b.)

CSS3:n ominaisuuksia ovat pyöristetyt kulmat, varjot, liukuvärit, siirtymät ja animaatiot. Samoin versio mahdollistaa myös uudet ulkoasut kuten multi-sarakkeet, joustavat laatikot ja taulukkotaiton. CSS on kehittynyt vuosien varrella ja on nyt versiossa 3.0 (kuva 12) (Flood 2015).



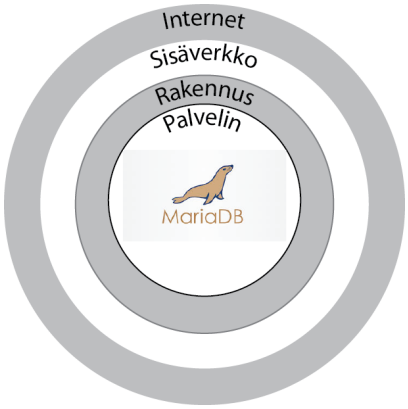
Kuva 12. CSS:n kehitys (Flood 2015).

3 Yleinen turvallisuus

Tässä opinnäytetyössä keskitytään tarkastelemaan turvallisuutta myös muista näkökulmista. Kansalaisopistossa, johon lainausjärjestelmä tehtiin on noin 300 opiskelukäyttöön laitteita lainaavaa opiskelijaa. Lainauksesta vastaavat työntekijät eivät tunne opiskelijoita henkilökohtaisesti. Olemassa olevaa varmistusjärjestelmää opiskelijan tunnistamiseksi ei ole, vaan lainaus perustuu luottamukseen. Lainaajan tunnistamisen varmuuteen halutaan muutos.

Koska uudessa järjestelmässä sekä opiskelijoilla että opettajilla on pääsy siihen, tulee henkilöstöturvallisuuteen kiinnittää erityistä huomiota. Käyttäjillä ei saa ole mahdollisuutta vahingossa tai tahallaan muuttaa sen tietosisältöä. Lisäksi järjestelmästä halutaan mahdollisimman helppokäyttöinen. Tämä lisää puolestaan turvallisuutta juuri käyttäjien mahdollisten inhimillisten virheiden varalta.

Turvallisuus voidaan ajatella kerrosmaisesti sisäkkäisinä renkaina, jossa tietokanta on kaiken ydin. Tietokanta sijaitsee palvelimessa, joka sijaitsee jossakin rakennuksessa, jotka molemmat pitää suojella hyvin. Tämän lisäksi myös yhteydet palvelimelle pitää suojata, eli ensin oma sisäverkko ja sitten internet (kuva 13).



Kuva 13. Tietokannan turvallisuus (Bartholomew 2013, 31).

3.1 Tietokannan turvallisuus ja sen suojaaminen

Tietokanta voi rikkoutua joko vahingossa tai tahallaan ja tämä halutaan estää oikeanlaisella suojauksella. Uhat voivat tulla eri muodoissa ja eri kautta. Ne voivat olla fyysisiä, tietojärjestelmään liittyviä, verkkoon tai käyttäjistä koituvia uhkia (Bartholomew 2013, 31).

Kansalaisopiston tietokannassa asiakkaat, siis opiskelijat ja opettajat, käyttävät tietokantaa web-käyttöliittymän kautta. Näin ollen tietokantaan on pääsy internetistä ja se muodostaa riskin. Koska suurin osa kaikista uhkista tulee juuri sitä kautta, tähän tulee varautua.

Vahtimestarit käyttävät tietokantaa sisäverkon kautta, joten myös sitä kautta on siis pääsy tietokantaan. Myös tämä pitää ottaa huomioon tietoturvassa. Esimerkiksi vain tietystä osoiteavaruudesta pääsee käsiksi niihin tietoihin, joita vahtimestarit tarvitsevat.

Tietokanta tulee todennäköisesti sijaitsemaan jossakin rakennuksen sisällä sijaitsevalla palvelimella, joten myös tämä tulee ottaa huomioon mahdollisten uhkien varalta. Palvelimen tulee sijaita lukitussa ja hyvin viilennetyssä paikassa.

Tämän lisäksi palvelimelle saa päästä käsiksi vain ainoastaan asianomaiset henkilöt. Tämäkin voidaan varmistaa siten, että taataan yhteys ainoastaan SSH:n (katso liite 2) avulla, jolloin pystytään estämään sellaiset yhteysyritykset palvelimelle, jotka eivät ole tunnistettuja. Lisäksi palvelimesta tulee ottaa päivittäiset varmuuskopiot jonnekin ulkopuolelle, mielellään toiseen rakennukseen, jotta mahdollisen ryöstön tai tulipalon tapahtuessa ei kaikkea tietoa menetettäisi.

Tietokanta voidaan suojata nopeasti ennen käyttöönottoa muutamalla yleisellä käskyllä, jolloin suojaustaso nousee huomattavasti. Tämä voidaan tehdä mysql_secure_installationiin. Toimenpiteinä ovat Root -käyttäjän salasana määrittäminen, anonymous-käyttäjien poistamisen, root-käyttäjän kirjautuminen verkosta käsin estämisen ja testitietokannan poiston sekä siihen pääsyn estämisen. Tämän lisäksi ladataan oikeustaulut ja varmistetaan, että kaikki edellä tehdyt muutokset otetaan saman tien käytäntöön. (Bartholomew 2013, 32-34.)

Toimenpiteiden jälkeen tietokantaan pääsee ainoastaan root-käyttäjä ja hänkin vain tietokoneelta, jossa tietokanta sijaitsee. Tämä ei välttämättä ole järkevää tietokannan käyttöä, joten tietokanta- ja/tai taulukohtaiset käyttäjät ja suojaukset kannattaa luoda.

Eräs erittäin tärkeä huomio tietokantaan kirjautumisessa on komentorivikirjautuminen. Koska systeemilokit tallentavat ylös kaiken, mitä käyttäjä kirjoittaa, ei siis kannata kirjautua komennolla **mysql –u root –p ’salasana’**, jossa tuon ’salasana’:n kohdalla on root-käyttäjän salasana, koska silloin ne käyttäjät, joilla on pääsy systeemilokeihin, näkevät mysql rootin salasanan. Parempi tapa on kirjautua komennolla **mysql –u root –p**

ja kirjoittaa salasana vasta, kun kone pyytää sitä, jolloin se ei kirjaudukaan systeemilokiin. (Bartholomew 2013, 35).

Sama koskee myös tiedostoa, jonka avulla voidaan kirjautua tietokantaan (esim. java-tiedostoa, joka ottaa yhteyden tietokantaan). Tämä tiedosto tulee suojata siten, että vain se käyttäjä, jolla on oikeus tiedoston käyttöön voi sitä käyttää, muilta se on kielletty (chmod 600 tiedosto). Kuitenkin on muistettava, että myös niillä ohjelmilla, jotka käyttävät tiedostoa, on oltava oikeus sen käsittelyyn (Bartholomew 2013, 36).

3.2 Palvelimen suojaaminen

Palvelin tulee suojata, kuten muutkin tietokoneet. Palvelimen yleistä päivityksistä ja virus- ja haittaohjelmaestoista tulee huolehtia. Samoin palvelin tulee sijaita turvatussa paikassa. Palvelimelle saa kirjautumisoikeudet vain hyvin rajattu joukko henkilöitä. Lisäksi kiintolevyn salakirjoitus (encrypt) lisää palvelimen suojausta. Myös päivittäiset varmuuskopiot jonnekin toiseen tilaan, mieluummin toiseen rakennukseen, ovat tärkeä osa tietojen suojaamista.

Palvelimen virran saanti tulee turvata sekä kahdentamalla palvelimen virtalähde että hankkimalla riittävän kokoinen UPS-laite. Siinä tulee olla toiminnot, joiden avulla palvelin voidaan ajaa automaattisesti alas, mikäli virran saanti jostain syystä katkeaa.

Mikäli palvelimen pääkäyttäjä ei toimi yrityksessä, tulee työntekijöillä olla tieto henkilöistä, joilla on riittävät oikeudet palvelimella tehdä sellaisia muutoksia, jotka saattavat olla haitallisia suhteessa yrityksen tietokantapalvelimeemme.

3.3 Rakennuksen suojaaminen

Palvelimen suojaaminen ei ole riittävää, mikäli se sijaitsee avoimessa tilassa. Siksi palvelimen tulee sijaita lukitussa tilassa ja jonne on pääsy vain asianomaisilla henkilöillä.

Lisäksi tilan tulee olla palo- ja vesivahingoilta hyvin suojattu. Mieluusti tila, jonka ilmastoinnista on huolehdittu, jotta palvelimen tuottama lämpö ei nostaisi huoneen lämpötilaa liian korkealle ja siten tuottaisi ongelmia itse palvelimelle. (Bartholomew 2013, 37)

Parhaassa mahdollisessa tapauksessa palvelin sijaitsisi omassa palvelinhuoneessa lukitussa palvelinräkissä, joka on pultattu seinään tai lattiaan. Tällä tavalla ehkäistään

ainakin laitteen fyysinen varastaminen. Lisäksi muut ominaisuudet pystyttäisiin varmistamaan sangen pitkälle huoneessa.

3.4 Sisäverkon suojaaminen

Mikäli tilanne on niin hyvä, että palvelin sijaitsee palvelinhuoneessa lukitussa palvelinräkissä, todennäköisesti yhteyttä siihen otetaan etänä. Tällöin tulee huolehtia sisäisen verkon turvallisuudesta. Avainasioita, joita tulee ottaa huomioon on useita. Ensiksikin ulkoisessa verkossa tulee olla palomuuri. Mikäli tätä ei vielä ole, se tulee hankkia välittömästi. (Bartholomew 2013, 38.)

Langattomasta verkosta tulee tarkistaa onko se suoraan yhteydessä sisäverkkoon vai onko se erotettu sisäverkosta. Mikäli se on suoraan yhteydessä sisäverkkoon, tulee harkita sen erottamista. Etätyöyhteyksissä kannattaa käyttää VPN tai SSH -yhteyksiä, sillä ne ovat salattuja yhteyksiä. Mikäli käytössä on jokin muu yhteys, täytyy tarkastaa suojaus ja salaus. (Bartholomew 2013, 38.)

Samoin tulee tarkistaa onko tietokantakäyttäjät määritelty %-muuttujalla verkko-osuudelta vai onko yhteys rajattu paikalliseen koneeseen tai tiettyihin osoitteisiin tai paikallisverkkoon. %-muuttuja tarkoittaa sitä, että käyttäjä voi ottaa yhteyttä mistä tahansa, mikä on käytännöllistä, mutta ei kovinkaan turvallista. Jos kyseessä on suuri yritys, onko osastoilla omat eristetyt verkkonsa, ja jos on, niin onko näistä verkoista pääsy tietokantapalvelimelle. Tietokannan ollessa osa tuotekehitystä, joka on vasta alkuvaiheessa, ei ehkä haluta, että esimerkiksi myyntiosastolla on pääsy tietokantaan, ennen kuin se on valmis. Kun otetaan etäyhteys tietokantaan, tulee se ottaa salatusti käyttäen SSH:ta tai jotain muuta salattua tunnelia. (Bartholomew 2013, 38.)

3.5 Internetin turvallisuus

Yleensä ei haluta näyttää MariaDB tietokantapalvelinta internetille suoraan. Tämä siksi, että MariaDB tietokantapalvelin olisi jotenkin herkemmin haavoittuvampi kuin jokin toinen ohjelma, vaan siksi, ettei yleensä ole tarvetta näyttää sitä internetille ja hyvään suojaustapaan kuluu, ettei mitään näytetä, ellei ole tarvetta. (Bartholomew 2013, 38.)

Mikäli MariaDB pyörii Webbipalvelimella, palvelimen softa voi ottaa suoraan yhteyttä tietokantaan tarvitsematta mitään internetyhteyttä. Mikäli taas MariaDB palvelin on erillisesti Webbipalvelimessa, voidaan melkein aina yhdistää nuo kaksi sisäverkon yli. Mikäli tämä ei onnistu, pystytään luomaan turvallinen tunneli noiden kahden välille. (Bartholomew 2013, 39)

3.6 Käyttäjäkohtainen tietokantaturvallisuus

Aiemmin mainittiin, että root -käyttäjää ei yleensä kannata käyttää, vaan luoda käyttöä varten tietokanta ja jopa taulukohtaisia käyttäjiä. Näin varmistetaan se, että tietokanta on mahdollisimman turvallinen. Jos ulkopuolinen jostain syystä on saanut jonkun käyttäjätunnuksen ja salasanan selville, hän ei pääse tekemään kovinkaan suurta tuhoa, koska käyttäjällä ei ole suuria valtuuksia tietokannassa.

Käyttäjäoikeudet jaetaan pääsääntöisesti kolmeen osioon. Puhutaan yleistä pääkäyttäjäoikeuksista, tietokanta-, taulu- ja kenttäoikeuksista sekä sekalaisista oikeuksista. Näitä kuvataan tarkemmin liitteessä Käyttäjäoikeudet. (Bartholomew 2013, 41.)

Käyttäjän luonti tapahtuu kahdessa vaiheessa. Ensin luodaan käyttäjä ja sitten annetaan käyttäjälle oikeudet. Jo ensimmäinen vaihe on turvallisuuden kannalta tärkeä, sillä käyttäjää luodessa määritellään, **mihin** käyttäjälle luodaan tili. Tili voi olla (Bartholomew 2013, 45.)

- mille tahansa palvelimelle (%)
- tietylle toimialueelle (%.lindroos.net)
- tiettyyn aliverkko peitteeseen (192.168.1.%) tai
- tiettyyn koneeseen (192.168.1.1),

Luonti tapahtuu komennolla: **CREATE USER 'kayttaja'@'palvelin' IDENTIFIED BY 'salasana'**. Käyttäjänimi, palvelin ja salasana on kirjoitettava lainausmerkkien sisään. (Bartholomew 2013, 45.)

On erittäin tärkeää määritellä **käyttäjän oikeudet**. Oletuksena uudella käyttäjällä ei ole muita oikeuksia kuin kirjautua sisään, mikä ei ole kovin hyödyllistä. Siksi seuraava toimenpide käyttäjän luonnin jälkeen onkin antaa hänelle ne oikeuden, jotka hän tarvitsee. Tämä tehdään GRANT -komennolla. Tämän avulla pystymme määrittelemään tietokanta ja jopa taulukkotasolla, millaiset oikeudet käyttäjällä on. Nuo oikeudet löytyvät kohdasta 3.2 ja 3.3. Muutama esimerkki: (Bartholomew 2013, s.46)

GRANT SELECT on edu.staff TO 'jpl'@'localhost'; - myöntää jpl-nimisellä käyttäjälle paikalliselle palvelimelle SELECT-oikeudet edu-tietokannan staff-taulukkoon. **GRANT ALL ON *.* TO 'jpl'@'%' WITH GRANT OPTION;** - myöntää jpl-nimiselle käyttäjälle kaikkiin palvelimiin kaikki oikeudet kaikille tietokannoille kaikkiin taulukkoihin siten, että hänellä on myös oikeus myöntää oikeuksia. **GRANT INSERT, DELETE, UPDATE on**

edu.* TO 'jpl'@'localhost' WITH MAX_UPDATES_PER_HOUR 100; -myöntää lisäys, poisto ja päivitys oikeuden jpl-nimiselle käyttäjälle paikalliselle palvelimelle edu-nimiseen tietokantaan kaikille tauluille siten, että maksimi päivityksiä tunnissa voi olla 100 kpl. (Bartholomew 2013, 46.)

Mikäli meille tulee joskus tarve poistaa käyttäjältä jokin tai jotkin oikeudet, se tapahtuu REVOKE-komennolla seuraavasti: **REVOKE oikeus (oikeudet) ON tietokanta FROM käyttäjä;** Mikäli haluamme nähdä, millaiset oikeudet tietyllä käyttäjällä on, se tapahtuu SHOW GRANT –komennolla seuraavasti: **SHOW GRANT FOR käyttäjä;** Mikäli haluamme vaihtaa käyttäjän salasanan, se tapahtuu SET PASSWORD –komennolla seuraavasti: **SET PASSWORD FOR käyttäjä = PASSWORD('salasana');** Mikäli haluamme poistaa käyttäjän, se tapahtuu DROP USER –komennolla seuraavasti: **DROP USER user;** (Bartholomew 2013, 46.)

Lisää käyttäjäoikeuksista löytyy liitteestä 1.

4 Ohjelmiston sijainti ja sen turvallisuus

Ohjelmiston sijainti vaikuttaa ratkaisevasti sen toiminnallisuuteen ja markkinointiin. Siitä voisi tehdä sellaisen ohjelmiston, jota vuokrattaisiin yrityksille, mutta jota ylläpidettäisiin omin voimin ja josta olisi periaatteessa vain yksi kappale olemassa, mutta useampi ilmentymä. Tämä kuitenkin vaatisi sitä, että ohjelmisto sijaitsisi pilvipalvelimessa. Toinen vaihtoehto on perinteinen ratkaisu, jossa tehdään ohjelmisto, joka myydään asiakkaille. Tätä ohjelmaa päivitetään tietyn aikajakson välein ja nuo päivitykset myydään asiakkaille. Tässä vaiheessa kuitenkin ohjelma tehdään vain kyseistä kansalaisopistoa ajatellen, mutta silti harkitaan ohjelmiston sijainnin paikkaa.

4.1 Pilvipalvelu

Pilvipalvelu tarkoittaa sellaista palvelumallia, jossa useiden käyttäjien kesken jaetaan tietoteknisiä resursseja, joita voidaan helposti säätää, tarjotaan tietoverkkojen yli. Palvelussa on toiminnallisuuksia, joita voidaan yhdistää keskenään helposti, kytkeä päälle ja pois käyttäjän tarpeiden mukaan. Kuormituksen ja käytön seuranta samoin kuin resurssien hallinta on tehty läpinäkyväksi ja helpoksi, joka mahdollistaa kulujen ja toiminnan optimoimisen (Kyberturvallisuuskeskus 2014, 5).

Pilvipalvelun turvallisuutta lähdettäessä arvioimaan, on huomioitava vuokraavan yrityksen tietoturvakäytännöt ja –teknologia ja jos mahdollista, stardardoitava, jotta pilvipalvelu olisi helpompi ottaa käyttöön. Pilvipalveluiden tietoturvahaasteet voidaan jakaa seuraavasti: (Tiger Team Blogi 2010, 6).

Datan varastointi: Yleisesti ottaen pilvipalveluiden tiedot sijaitsevat useassa eri paikassa yleensä useassa eri maassa. Pienemmillä toimijoilla voi olla oma konesali tai he voivat vuokrata tilaa suuremman toimijan konesalista. Yksittäiselle käyttäjälle tällä ei kuitenkaan ole väliä, sillä hän ei tiedä, missä hänen tietonsa sijaitsee, ellei hän sitä erikseen selvitä palveluntarjoajalta. Osa suurista tarjoajista pyrkii alueellisuuteen, eli siihen, että eurooppalaisten asiakkaiden tiedot ovat eurooppalaisilla palvelimilla (Kyberturvallisuuskeskus, 2014. 9).

Käyttöoikeuksien ja pääsynhallinta: Palvelun tarjoajan kanssa on syytä sopia ehdoista huolellisesti. Erityisesti on kiinnitettävä huomiota tiedon käsittelyoikeuteen ja hallintaan liittyviin seikkoihin (Kyberturvallisuuskeskus 2014. 17).
Tiedon elinkaaran hallinta: Yleisesti ottaen tiedon elinkaari alkaa, kun tieto luodaan ja päättyy kun se ja kaikki sen kopiot tuhotaan. Tiedon poistaminen tavanomaisin keinoin ei merkitse sitä, että tieto lakkaisi olemasta. Kannattaa siis varmistaa, mitä tiedolle oikeasti

tapahtuu, kun se poistetaan käyttäjän toimesta. Erityisen tärkeää on muistaa huolehtia siitä, mitä tiedoille tapahtuu, kun asiakassuhde päättyy. (Kyberturvallisuuskeskus 2014. 7 - 9).

Datakeskustoiminnot ja datan eristämisen hallinta: Datan eristämisen asiakkaiden kesken voi tehdä joko loogisesti tai fyysisesti. Looginen erottelu tapahtuu ohjelmallisesti (käyttävät samaa ohjelmaa) tai siten, että jokaisella käyttäjällä on oma instanssi käytössään ohjelmasta, mutta ne toimivat samalla virtuaalisella tai fyysisellä palvelimella. Fyysinen taas sitä, että jokaisella on oma instanssi omalla virtuaalisella tai fyysisellä palvelimella. Näistä viimeisin on kaikkein turvallisin (Kyberturvallisuuskeskus 2014. 9 - 10).

Palveluntarjoajan turvamallin luotettavuus: Pilvipalveluissa tietoa hajautetaan eri palvelimille toiminnan varmistamiseksi ja siksi on syytä varmistaa, missä käyttäjän tieto sijaitsee koko sen elinkaaren aikana. Tähän vaikuttavat mm. kyseisen maan lait (mitä saa tallentaa, mitä lakia sovelletaan poikkeus tapauksessa, onko maan viranomaisella oikeus tutkia tietoaineistoa?) (Kyberturvallisuuskeskus 2014. 7 - 9).

Yrityksen riskienhallinta ja IT-ympäristön hallintatapa palvelussa: Kun organisaatio miettii pilvipalvelun käyttöönottoa, sen kannattaa tehdä riskianalyysi. Siinä otetaan mm. huomioon se, että on mahdollista, että tieto katoaa, vääristyy, tuhoutuu tai joutuu kolmannen osapuolen haltuun (Kyberturvallisuuskeskus 2014. 11).

Omistusoikeuden tutkiminen: Pääsääntöisesti omistajuus ja siihen liittyvät oikeudet ovat sillä henkilöllä tai organisaatiolla, joka on alun perin asiakirjan pilveen tallentanut (Kyberturvallisuuskeskus 2014. 7).

Auditointimahdollisuudet, dokumentointi ja viranomaismääräykset: Organisaatio, joka ottaa palvelun käyttöön, on syytä sisällyttää sopimukseen audiointioikeus siihen, että organisaation tietoja käsitellään sopimuksen mukaisesti. Erityisesti tulee kiinnittää huomiota arkaluonteisen ja salassa pidettävän materiaalin käsittelyyn (Kyberturvallisuuskeskus 2014. 7).

Kryptaus ja avainhallinnointi: Vuonna 2013 julkisuudessa kohistiin siitä, että eri maiden turvallisuuspalvelut salakuuntelivat mannertenvälistä liikennettä sekä suurten palvelinkeskusten välistä liikennettä. Mikäli tieto tai tietoliikenne ei ollut salattua, se kulki selväkielisenä ja oli luettavissa. Tämän voi ratkaista kryptaamalla datan. Ongelmana kuitenkin on, että tämä jää käyttäjän vastuulle, koska palveluntarjoaja ei kryptausta tarjoa. Kryptaus vaatii salausavaimen, jolla data kryptataan. Palveluntarjoaja ei voi pitää tätä

avainta hallussaan, koska kyseinen toimintatapa ei takaisi tietojen salausta, sillä silloin sekä salausavain, että itse data olisivat palveluntarjoajan hallussa. On kuitenkin yrityksiä, jotka ovat erikoistuneet kryptaukseen, jolloin kyseisistä avaimista huolehtiminen siirtyy heidän vastuulleen (Rantanen 2012, 15).

Sopimusehdot ja niihin vaikuttaminen: Pilvipalveluiden käyttöä rajoittavat lainsäädäntö ja sopimusehdot. Jälkimmäiseen voidaan vaikuttaa sopimusta tehdessä. Sopimusta tehdessä organisaation on syytä muistaa myös omat sopimusvelvoitteensa, kuten esimerkiksi tiedon siirron ulkomaille (Kyberturvallisuuskeskus 2014. 11).

Pilvipalvelulaitteistot sijaitsevat ympäri maailmaa, josta aiheutuu hankaluus tietosuojan kannalta. Tallennetun tiedon säilytykselle, käsittelylle ja tallennukselle on eri maissa eri lait. Myös tietosuojan tarjoaminen vaikeutuu em. seikkojen vuoksi. Suuren riskin aiheuttaa myös tiedon vuotaminen. Joidenkin maiden viranomaisilla on oikeus ilman erillistä lupaa salakuunnella verkkoliikennettä omassa maassaan (Rantanen 2012, 15).

Ehkä kaikkein tutuin ja myös turhauttavin ongelma on se, ettei nettiyhteys toimi. Omalla kohdallani on muutamakin pitkään kirjoitettu sähköposti kadonnut, kun nettiyhteys on äkillisesti kadonnut. Onneksi monilla pilvipalvelun tuottajilla on mahdollisuus siihen, että niitä voi käyttää ns. offline ja ne päivittyvät sitten, kun saavat taas yhteyden. Ja vaikka laajakaista on Suomessa perusoikeus, ei se edelleenkään toteudu kaikkialla siinä mittakaavassa, kun se aikoinaan eduskunnassa määriteltiin. (Sulopuisto 2013).

Isot verkkopalvelut on tehty sietämään monenlaisia vikoja. Tiedot kopioidaan identtisesti useille kovalevyille ja palvelinsalien generaattorit varmistavat virransaannin. Mutta entä jos maanjäristys tuhoaa koko palvelinsalin? Joidenkin pilvipalveluiden tuottajien, kuten Microsoftin, tiedot ovat varmistettu Irlannista Manner-Eurooppaan, joten tällä tavoin pystytään varautumaan jopa näin suuriin katastrofeihin. Kuitenkin on nähty, kuinka sekä Gmail, että Amazon ovat olleet hätää kärsimässä, jolloin vika kohdistuu miljooniin käyttäjiin. (Sulopuisto 2013).

Vähän väliä saa lukea lehdistä, kuinka hakkeri on saanut haltuunsa tuhansia tai jopa satoja tuhansia luottokorttitietoja tai salasanoja tai tilitietoja. Tämä saa mietityttämään, ovatko tiedot todellakaan turvassa pilvessä. Esimerkiksi Dropboxiin oli pääsy paikallisilla tiedostelupalveluilla muutama vuosi sitten. (Sulopuisto 2013).

”Kun tieto on kerran verkkoon laitettu, niin se on siellä pysyvästi”. Tämä ei pidä paikkaansa, sillä esim. Googlella on palvelu, jonka avulla saa poistettua itseään

kohdistettua herjaviestintää. Eräät opiskelijat ovat saaneet apunani juuri tällaisessa asiassa ja se on toiminut, mutta on varmasti joitakin palveluita, jossa tiedot pysyvät ikuisesti. On olemassa eräs internetpalveluntarjoajan, jonka toiminta on lakannut jo yli 20 vuotta sitten, mutta edelleen sivut näyttävät toimivan (tosin sen osti toinen yritys, jonka osti toinen yritys …), vieläpä alkuperäisellä nimellä! (Sulopuisto 2013).

Mikään ei kestä ikuisesti, varsinkaan ilmaiset tai halvat nettipalvelut. Aikoinaan tällaista vastaan perustettiin iki.fi –niminen sähköpostipalvelu, jotta käyttäjillä olisi ikuinen sähköposti. Tämä yhdistys perustettiin jo vuonna 1995 ja toimii edelleen, joten edes jotain pysyvää on. Mutta aina on olemassa se mahdollisuus, että palvelu lakkaa olemasta. Jos tällaisessa palvelussa on paljonkin omaa tietoa, esim. oma sivusto ja vaikkapa tietokanta ja jokunen toimiva ohjelmisto, niin silloin on pakko keksiä keino siirtää tiedot jonnekin toisaalle. (Sulopuisto 2013).

4.2 Oma palvelin

Pilvipalvelin pystyy ”lähes” kaikkeen, mihin oma palvelinkin. Miksi sitten edes miettiä omaa palvelinta, eikö olisi järkevää suoraan ottaa käyttöön joustava ja kaikkialta käytettävä järjestelmä? Suoraa vastausta tähän ei pysty antamaan, se riippuu niin järjestelmän tarpeista, mutta on muutamia yleisiä asioita, joita tässä tarkastellaan:

Käytettävyyttä voidaan tarkastella kahdelta eri näkökulmalta. Ensimmäinen näkökulma on tietenkin pilvipalveluiden näkökulma, eli se, että palvelu on käytössä kaikkialta. Tämä saattaa olla monesti tarpeen sellaisissa ohjelmissa, joiden käyttäjäkunta on laaja. Mutta entä sellaiset ohjelmat, joissa käsitellään arkaluonteisia, koulun sisäisiä tietoja? Halutaanko, että sellaiset ohjelmat ovat käytettävissä koulun ulkopuolelta - tässä tulee vastaan kysymys tietoturvasta. Vaikka ohjelma itsessään olisi tietoturvallinen, onko tiedon siirto sitä? Eli siirtyykö tieto selväkielisenä käyttäjän ja ohjelman välillä? Tällaisen viestin voi hakkeri helposti napata ja jos siinä viestissä on esim. oppilaan henkilötiedot, pankkitilin numero tms. arkaluonteista tietoa, ei se ole kovin hyväksi koulun maineelle. Tästä syystä olisi hyvä pitää tietyt ohjelmat pelkästään koulun lähiverkossa.

Vaikka perustamiskustannukset omalle palvelimelle saattavatkin olla suuret, eivät ne pitkälle ajalle jaettuna ole verrattuna vastaavanlaisen palvelun ostamiseen pilvipalveluna. Toki pilvipalvelun etuna on se, ettei tilaa ym. resursseja tarvitse ostaa heti ajatellen tulevia tarpeita kuten omaa palvelinta hankittaessa, vaan resursseja voi kasvattaa tarpeen tullessa esiin, mutta siitä huolimatta oman palvelimen kustannukset ovat halvemmat. Toki oman palvelimen kustannuksiin on otettava huomioon myös päivityskulut, palvelimen

ylläpitoon kuluvat resurssit, tietoturva, varmistukset sekä käyttökatkosten estäminen (Clarke 2014).

Myös on otettava huomioon se, ettei joka tapauksessa kaikkia sovelluksia voida siirtää pilvipalvelimelle, joten oma palvelin on oltava, jonka takia kustannukset tällöin jakautuisivat vain kahteen eri paikkaan, mikäli päädyttäisiin pilvipalveluratkaisuun. Toki tässä tapauksessa voitaisiin hankkia kevyempi palvelin, mutta raudan osuus oman palvelimen kustannuksista on kuitenkin vain n. 25 – 40 % kokonaiskustannuksista, kun lasketaan kustannuksia esim. viisivuotisperiodilla (TechNet 2015).

Oman palvelimen hyvä puoli on se, että vaikka internet-yhteydet katkeaisivat, voi työtä edelleen jatkaa omassa lähiverkossa. Eräässä yrityksessä internet-yhteyksien katkeaminen tarkoitti samaa kuin työnteon katkeaminen. Kun 16 hengen työnteko katkeaa kahdeksi tunniksi, se tarkoittaa käytännössä jo neljää täyttä työpäivää, eli lähes viikon työtunteja.

Vaikka tätä ei tapahdu kovinkaan usein, oman sähköpostipalvelimen pitäminen on silti suositeltavaa. Tällöin käyttäjä voi lähettää, vastata ja vastaanottaa niitä viestejä, jota on jo postilaatikossa ja nuo viestit, jotka hän lähetti tai joihin hän vastasi lähtevät heti, kun yhteys on taas saatu kuntoon (TechNet 2015).

5 Ohjelman kuvaus

Ohjelma jakautuu kahteen osaan: käyttäjämoduuliin ja pääkäyttäjämoduuliin. Käyttäjämoduulissa on kaikki se, mitä tavallinen käyttäjä näkee ohjelmassa ja pääkäyttäjämoduulissa on kaikki ylläpitoa koskevat toiminnot. Ohjelman tehtävänä on suorittaa lainaustoimintaa siten, että lainauslomakkeelle saadaan ylös sekä lainaajan tiedot, että lainattavien laitteiden tiedot sekä niihin liittyvien lisäosien tiedot.

Ohjelma on rakennettu siten, että käyttäjä (yleensä opiskelija) tekee lainahakemuksen verkossa (alkuvaiheessa vain sisäverkossa, mutta tulevaisuudessa myös internetissä), jonka jälkeen tieto tästä menee vahtimestarille, joka valmistelee lainaustapahtuman hakemalla lainattavat tavarat valmiiksi esille. Tämän jälkeen oppilas noutaa siihen kellon aikaan, kun hän on tehnyt lainavarauksen, lainattavat tuotteet, tarkastaa laitteet yhdessä vahtimestarin kanssa, ettei mitään puutu mistään paketista ja kuittaa lainadokumenttiin saaneensa lainan. Tämän jälkeen vahtimestari antaa kopion lainadokumentista lainaajalle ja arkistoi oman kopionsa kansioon. Samalla vahtimestari merkitsee ohjelmaan ko. laitteet lainatuiksi, jolloin nämä laitteet eivät näy lainattavien tuotteiden listalla.

5.1 Käyttäjämoduulit

Pääkäyttäjämoduulissa pääkäyttäjällä on mahdollisuus hallita koulutusaloja, kursseja, asiakkaita, postiosoitteita, puhelinnumeroita, sähköposteja, lainauksia, tuoteryhmiä, tuotteita, tuotekuvauksia ja lisäosia.

Pääkäyttäjämoduulissa pääkäyttäjä valitsee alasvetovalikosta taulukon, jota hän lähtee hallitsemaan. Tämän jälkeen hänelle tulee näkyviin taulukko ja sen tiedot ja taulukon rivien viereen mahdollisuus hallita kyseistä riviä (editoida, poistaa). Lisäksi hänelle tulee näkyviin painike, jolla hän voi tarvitessa lisätä uuden tietueen.

Huomattavaa on, että mikäli asiakkaalla on useampi puhelinnumero tai sähköposti, kyseinen asiakas näkyy tässä vaiheessa useammalla rivillä, jokaista puhelinnumeroa ja/tai sähköpostia vastaa yksi uusi rivi.

Sama koskee myös tuotetta, eli kun tuotteella on useampi lisäosa, tuote esiintyy useammalla rivillä – jokaista lisäosaa vastaa yksi uusi rivi. Tämä koskee kuitenkin vain taulukko näkymää, kun lähdetään yksittäistä tuotetta tai asiakasta muokkaamaan tai valitsemaan, asia muuttuu toisenlaiseksi (liite 3).

Käyttäjämoduulissa käyttäjällä on mahdollisuus lainata tuotteita. Hän valitsee alasvetovalikosta tuoteryhmän, josta hän on kiinnostunut ja saa ne näkyviin taulukon, jossa näkyvät ne laitteet, jotka ovat sillä hetkellä vapaina. Hän voi tämän jälkeen varata yhden näistä laitteista itselleen ja joko jatkaa laitteiden varaamista tai tulostaa varaustositteen.

Ohjelmasta otetut kuvakaappaukset löytyvät liitteestä 3.

5.2 Tietokannan kuvaus

Tietokanta koostuu useasta taulusta. Käyttäjien tiedot tulevat suoraan Active Directorysta (tulevaisuudessa, ei vielä ohjelmoitu ominaisuus). On huomioitava, että yksikään kysely ei yksinään koske pelkästään yhtä taulua. Asiakas-taulu koostuu neljästä taulusta, jotka ovat Asiakas, Posti, Puhelin sekä Email. Kurssi-taulu koostuu kahdesta taulusta, jotka ovat Kurssi ja Koulutusala. Lisäksi Asiakas ja Kurssi-taulukoiden kokonaisuuksien yhdistää taulu KurssiAsiakas (kuva 14).

Samoin myös Tuote-taulu koostuu useammasta taulusta, eli Tuoteryhmä, Lisäosa ja Tuotekuvaus-tauluista. Lisäksi TuoteLisäosa yhdistää Tuote ja Lisäosa –taulut. Tämän lisäksi Lainaustaulu yhdistää Asiakas ja Tuote –taulut keskenään, kuitenkin niin, että Tuotetaulun ja Lainaustaulun välissä on TuoteLainaustaulu (katso liite 4).

Huomioitavaa on myös se, että Asiakas-taulussa Puhelin ja Email –taulut ovat laitettu omiksi tauluikseen sen vuoksi, että asiakkaalla saattaa olla useampi puhelinnumero ja/tai sähköposti. Postitaulu on omanaan taas sen vuoksi, että asiakastauluun riittää vain Postinumero ja Postitaulusta haetaan automaattisesti postitoimipaikka postinumeron perusteella.

Tuotetaulussa usealla tuotteella saattaa olla sama kuvaus, joten tuotekuvaus on eriytetty omaksi taulukseen. Koska tuotteella saattaa olla useita eri lisäosia, on lisäosataulu jouduttu yhdistämään tuotetauluun välitaululla, jotta tämä ongelma on saatu ratkaistua. Tämä sama koskee myös lainausta, eli koska samaan lainaukseen saattaa tulla useampi tuote, on lainaustaulun ja tuotetaulun väliin laitettu välitaulu, jotta tämä ongelma on saatu ratkaistua.

5.3 Ohjelman tietoturva

Ohjelman tietoturvaa haluttiin tarkastella myös muusta, kuin ohjelmalliselta kannalta, koska kyseessä oli kokonaisturvallisuus, josta toimeksiantaja oli huolissaan. Vaikka tämä ei ollut alkuperäinen opinnäytetyön aihe, aihetta kuitenkin käsiteltiin opinnäytetyössä luvuissa 3 ja 4.

Tämän opinnäytetyön aikana tietoturvaa parannettiin huomattavasti toimeksiantajalla. Opistolle oli vuotta aikaisemmin hankittu uusi palvelin, jonne ohjelmisto päätettiin laittaa. Palvelin sijaitsi palvelinräkissä, joka siirrettiin kellarista, jossa se oli alttiina vesivahingoille, autotalliin, jonne sille rakennettiin oma ilmastoitu huone. Tästä oli hyötyä myös siinä, että internet-yhteydet tulivat autotalliin, joten enää ei tarvinnut vetää pitkiä kaapeleita autotallista kellariin. Vaikka kellariinkaan ei periaatteessa päässeet muut kuin ne, joilla oli sinne avain, oli kuitenkin kellarin oven edessä luokkahuone ja aina välillä oli tilanteita, jolloin ulkopuolinen toimija oli tekemässä töitä kellarissa ja hänellä ei ollut avaimia, jolloin ovi piti jättää auki. Tämä aiheutti tietoturvariskin, joka nyt pystyttiin poistamaan, kun palvelinräkki siirrettiin autotalliin. Jo aiemmin oltiin hankittu kunnon palomuurilaite, joten siltä osin asiat olivat kunnossa. Langattoman verkon tietoturvaa lähdettiin myös kehittämään, mutta asia oli vielä kesken, kun tätä opinnäytetyötä kirjoitettiin.

Käyttäjäturvallisuutta nostettiin siten, että autotalliin oli vain tietyillä henkilöillä pääsy ja vielä rajoitetummalla määrällä henkilöstä oli pääsy tähän autotallissa olevaan räkkihuoneeseen. Lisäksi palvelimen käyttäjäoikeudet rajoitettiin niin, ettei palvelimelle päässyt etäkirjautumaan kuin tietyiltä päätteiltä. Palvelimelle rajoitettiin pääkäyttäjäoikeuksia siten, että kerrallaan vain kahdella henkilöllä oli pääkäyttäjäoikeudet.

Tietokannasta otetaan varmuuskopiointi joka yö verkon yli toiseen rakennukseen. Koska kyseessä on tietokantatyyppinen tiedosto, ei varsinaiselle varmuuskopioinnin kierrätykselle ole tarvetta, mutta varmuuden vuoksi kuitenkin tehtiin kolmen viikon sykli varmuuskopioinnin kierrätykselle, mikäli joku pääsee sittenkin hyökkäämään tietokantaan ja sitä ei huomata riittävän ajoissa. Myös palvelimesta otettiin varmuuskopio joka yö toisessa talossa olevaan NAS-palvelimeen (katso liite 2).

MariaDB-tietokanta suojattiin ohjeiden mukaisesti (katso luku 3.1) sekä tehtiin käyttäjäoikeudet seuraten ’vähimmän etuoikeuden’ periaatetta. Java ohjelmoinnissa seurattiin tarkasti Oraclen antamia ohjeita. Kaikki HTML5-lomakkeista sisään tuleva data tarkastettiin JavaScriptien avulla, jotta minkäänlaisia hyökkäyksiä ei pääse tulemaan sisään. Tämän jälkeen data syötettiin Java Server Page –servettiin ja sitä kautta

JavaEE:n enterprise beaniin. Myös tässä välissä käytettin Javan omia tarkistuksia, ei siis luotettu pelkästään JavaScriptien tarkistuksiin. Java Server Page käytti omia tarkistuksia käyttöoikeuksien (roolien) kanssa. Tunnistus tapahtui kaksi- tai jopa kolmivaiheisesti: ensin palvelin tarkistaa käyttöoikeuden HTML5-lomakkeella ja sen jälkeen JSP-sivu tarkistaa vielä, että rooli on oikea, sitten vielä tarkastetaan valtuutus, ennen kuin käyttäjä pääsee käyttämään enterprise bean -säiliötä (katso 2.2).

6 Pohdinta

Opinnäytetyö pyrki etsimään vastauksia kysymyksiin: onko kyseinen ohjelmointikieli tietoturvallinen? Sekä: miten tietoturvaa kyseisen ohjelmointikielen sisällä voi vahvistaa? Tässä mielestäni onnistuttiin hyvin.

Ohjelmointiprojektissa käytettiin uusimpia ohjelmointikieliä, jotka mahdollistivat ajan tasalla olevat tietoturvaratkaisut. Kun käytettiin suurta joukkoa eri ohjelmia, jouduttiin tarkastelemaan tietoturvaa eri näkökulmista. Tämä siksi, että tietoturva ei heikkenisi tai projekti muuten vaarantuisi.

Java on ohjelmointikieli, jonka siirrettävyys eri alustojen kesken teki siitä tähän projektiin ylivoimaisen ohjelmointikielen. JavaEE taas lisäsi ohjelmaan ominaisuuksia, jotta Java-ohjelmia voidaan käyttää nettisivuilla. MariaDB antoi tarvittavat tietokantaominaisuudet, joita tarvittiin tietojen tallentamiseen ja käsittelemiseen. HTML5 ja CSS3 pitivät taas huolen siitä, että tuloksia voitiin esittää nettisivuilla sekä lähettämään käyttäjän antaman tiedon palvelimelle. JavaScript mahdollisti käyttäjän antaman tiedon tarkistamisen ennen palvelimelle lähettämistä.

Ohjelmointikielinä Java on turvallinen ja jos turvallisuutta haluttiin lisätä, voitiin Javaohjelmat ajaa omassa ”hiekkalaatikossaan” ilman mahdollisia systeemi- tai käyttäjäriskejä. JavaEE:ssä turvallisuus voitiin viedä pitkälle käyttämällä Enterprise Bean Business metodeja, jolloin suojattava tieto kapsuloitiin omaan EJB-säiliöön ja sinne päästiin vain sillä valtuutuksella, joka on aikaisemmin saatu. MariaDB:ssä tietoturva on vielä pidemmällä kuin MySQL:ssä ja hyvällä hallintatavalla tietokannan ylläpitäjä pystyi tekemään tietokannasta erittäin tietoturvallisen. Jotta HTML5:stä sai tietoturvallisen, täytyi käyttää
JavaScriptiä apuna tarkistamaan kenttien sisällöt, ettei käyttäjä päässyt syöttämään sellaista sisältöä, joka olisi ollut haitallista palvelimelle (esim. XSS-hyökkäys).

Turvallisuutta piti tarkastella lyhyesti myös fyysiseltä kannalta, sillä ohjelmointiprojektin tuotos päätettiin sijoittaa omalle palvelimelle. Niinpä palvelin tuli suojata siten, ettei sille pääse kuin asianosaiset eikä sille voi aiheutua vahinkoa. Samoin rakennus tuli suojata siten, ettei palvelimelle voi aiheutua vahinkoa. Sisäverkko tui suojata palomuurilla eikä siihen tullut päästä ulkopuolelta kuin suojatuilla yhteyksillä ja internetin suuntaan ei pitänyt näkyä mitään muuta kuin tarvittavat palvelut ja sivut.

Tässä työssä pohdittiin myös oman palvelimen ja ulkoisen palvelimen etuja ja haittoja. Vaikka pilvipalvelussa on erittäin paljon etuja, skaalautuvat resurssit ehkä parhaimpana, päädyttiin kuitenkin omaan palvelimeen siksi, että yhteyksien katkettuakin ko. tuotosta voi käyttää ongelmitta. Toinen syy tähän oli se, ettei kaikkia ohjelmia kuitenkaan voi siirtää pilvipalvelimelle. Oma palvelin on joka tapauksessa oltava, joten tässä vaiheessa ei kannattanut hankkia turhia resursseja, kun on jo olemassa äskettäin hankittu oma palvelin.

Tätä projektia olisi voitu ohjelmoida toisilla ohjelmointikielillä, kuten .net-alustalla, eli C#:lla tai PHP:llä. Tuolloin tietoturvaratkaisut olisivat olleet erilaiset ja puutteellisemmat (pool-70-107-200-115.ny325.east.verizon.net 2007). Toisaalta tätä projektia olisi voinut myös lähestyä olemassa olevilla työkaluilla, mutta siten, että olisikin paketoinut javakoodin omaksi appletikseen ja ajanut sen HTML-sivun sisällä. Tämä varmasti olisi ollut tietoturvallinen tapa, mutta appletteihin liittyvät ongelmat, eli selainten haluttomuus ajaa niitä, oli yksi syy, miksi tähän ratkaisuun tässä työssä ei päädytty. Kuitenkin tuota lähestymistapaa kannattaa tulevaisuudessa tutkia, olisiko siitä mahdollisesti ratkaisu muun muassa XSS-ongelmiin.

Täysin turvallista nettiohjelmaa ei ole ja Javastakin löytyy aika-ajoin tietoturva-aukkoja. Siksi jokaisen ohjelmoijan / ylläpitäjän tulee olla ajan tasalla ja päivittää Java heti, kun siitä ilmestyy uusi versio. Siksi ohjeissa mainittiinkin, että ylläpitäjien tulee tilata sähköpostiinsa kriittisten päivitysten tiedotukset. Samoin ylläpitäjien tulee seurata palomuurien lokitiedostoja ja toimia, mikäli jotain poikkeavaa sieltä löytyy. Ohjelmoijankin olisi hyvä vähintään puolivuosittain tehdä päivitystarkastus, ellei sitä ennen ole ilmaantunut mitään hälyttävää tietoturvauhkaa joko itse ohjelmasta ja maailmalta jonkin em. ohjelman puolesta.

Kansalaisopiston tapauksessa ainakin alkuvaiheessa, kun kaikkia kehitysehdotuksia ei ole ehditty toteuttaa ohjelmaan, voitiin pääsy ohjelmaan rajoittaa ainoastaan paikallisverkkoon. Tällöin ulkopuolinen uhka pieneni huomattavasti, kun käyttäjiä voivat olla vain ne, joilla on pääsy oppilas-, opettaja- tai toimistoverkkoon.

Opinnäytetyön tekeminen oli opettavainen prosessi. Ennen tämän opinnäytetyön tekemistä tietoni ohjelmointikielten tietoturvasta olivat puutteelliset. Olin tutustunut HTML-kielen XSS-hyökkäyksiin, mutta kaipasin lisätietoa, miten hyökkäyksiin voidaan varautua sekä yleisesti ohjelmointikielten tietoturvasta. Erityisesti minuun vaikutti tieto siitä, että jo suhtautuminen tietoturvaan riittävällä vakavuudella poistaa neuvottomuuden hyökkääjiä vastaan. Voimme turvautua jopa erittäin hyvin erilaisia hyökkäyksiä vastaan. Toisaalta olin

hämmentynyt siitä, että uudelleen ja uudelleen sain lukea, että jos vain ohjelmoijat suhtautuisivat näihin asioihin riittävällä vakavuudella, ongelmilta vältyttäisiin. Onko todellakin niin, että ei haluta, tai ei viitsitä käyttää ylimääräistä aikaa sivujen tekemiseksi tietoturvallisiksi – vai onko kyse vain tietämättömyydestä.
Jos kyse on jälkimmäisestä, toivon, että tämä opinnäytetyö tavoittaa riittävän lukijakunnan tiedon lisäämiseksi. Tämä opinnäytetyö pystyy vastaamaan vain rajatusti tietoturvaa koskeviin asioihin. Toivon, että sen avulla heräisi kuitenkin kiinnostusta kyseenalaistaa, ovatko sivustot todellakin tietoturvallisia.

Työn rajaamisen kannalta tehdyt ratkaisut osoittautuivat hyviksi. Opinnäytetyön tuotoksena syntyi lainausjärjestelmä kansalaisopiston käyttöön. Jatkokehittelynä aiheena olisi käyttäjäkokemusten kerääminen ja ohjelmiston päivitys saatujen kokemusten perusteella. Kuten jo edellä mainittiin, mm. Javasta löytyy jatkuvasti tietoturva-aukkoja, joten ohjelman pitäminen tietoturvallisena vaatii sitä, että seuraa alan kehitystä ja on ajan hermolla jatkuvasti. Tämän opinnäytetyön aihealue vaikutti käytetyn lähdekirjallisuuden valintaan. Tekijä halusi hyödyntää mahdollisimman uutta tietoa. Koska muutokset olivat nopeita tietoturvan alueella, hyödyllisiksi lähteiksi valikoituivat verkkolähteet. Runsas verkkolähteiden määrä ei tässä tapauksessa ollut huono asia, vaan kertoi tietoturvasta liittyvien ajantasaisten artikkeleiden ja julkaisujen siirtymisestä pitkälti verkkoon.

Tämän opinnäytetyön jatkokehittelykohteena on ehdottomasti uudet ohjelmointikielet ja tekniikat (muun muassa Scala, Go / MEAN, Dart, Angular). Toki myös vanhoihin kieliin tutustuminen, joita tässä ei käsitelty, mutta joita voi käyttää internetohjelmoinnissa, kuten C/C#, Perl, PHP, Python, Ruby, on vaivan arvoinen asia. Myös olemassa olevien kielten kehitys, kuten HTML5:n, seuraaminen, on ehdottoman tärkeää.

7 Lähdeluettelo

AT&T Laboratories Cambridge, 1999. Virtual Network Computing. AT&T. Luettavissa: http://www.hep.phy.cam.ac.uk/vnc_docs/howitworks.html. Luettu: 27.4.2015.

Bartholomew, D. 2013. Getting Started with MariaDB. Packt Publishing Ltd, Birmingham B3 2PB, UK. Luettu: 13.2.2015

Chapman, S. 2015. What is JavaScript? About Tech. Luettavissa: http://javascript.about.com/od/reference/p/javascript.htm. Luettu: 24.4.2015.

Clarke, K. 2014. Pilvipalvelut vs. omapalvelin. CDKGlobal. Luettavissa: http://www.cdkglobal.fi/uutiskirje-ja-esitteet/artikkeli/hsp-vs-own-server.asp. Luettu: 17.3.2015.

eduCBA Academy for IT Training, Global IT Training Experts 2014. Introduction to Java - video. Educorporatebridge / Udemy. Katsottu: 13.4.2015.

Flood, B. 2015. CSS3. Mozilla Developer Network. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3. Luettu: 19.3.2015.

Glynn, F. 2015. JavaScript Security. Veracode. Luettavissa: http://www.veracode.com/security/javascript-security. Luettu: 25.4.2015.

Html5security, 2010. Guide to Secure Implementation of HTML5's Cross Origin Requests, Google Project Hosting. Luettavissa: https://code.google.com/p/html5security/wiki/CrossOriginRequestSecurity. Luettu 4.6.2015.

Java Server Pages 2014. JSP – Security. Tutorialspoint. Luettavissa: http://www.tutorialspoint.com/jsp/jsp_security.htm. Luettu: 17.4.2015.

Kotiwitcz, K. & Lara, J., Roxberry, M., Shah, S., Stranathan, W. 2014. HTML5 Security Cheat Sheet. OWASP Foundation. Luettavissa: https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet. Luettu: 20.4.2015.

Kyberturvallisuuskeskus 2014. Pilvipalveluiden turvallisuus. Viestintävirasto. Luettavissa: https://www.viestintavirasto.fi/attachments/tietoturva/Pilvipalveluiden_tietoturva_organisaatioille.pdf. Luettu: 24.2.2015.

The Linux Information Project. TCP Definition. Luettavissa: http://www.linfo.org/tcp.html. Luettu: 27.4.2015.

MariaDB 2010. About MariaDB. MariaDB. Luettavissa: https://mariadb.com/kb/en/mariadb/about-mariadb/. Luettu: 19.4.2015.

MariaDB Foundation 2015. Luettavissa: https://mariadb.org/en/about/. Luettu: 18.3.2015.

Mavrody S. Sergey's HTML5 & CSS3 Quick Reference. 2nd Edition. Belisso Corp., 2012. ISBN 978-0-9833867-2-8. Luettu: 18.3.2015.

Merriam-Webster. Spam. Luettavissa: http://www.merriam-webster.com/dictionary/spam. Luettu: 27.4.2015.

Mitchell, B. FTP - What Does FTP Stand For? Luettavissa: http://compnetworking.about.com/od/networkprotocols/g/bldef_ftp.htm. Luettu: 27.4.2015.

Oracle 2013. The Java EE 6 Tutorial. Oracle. Luettavissa: http://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf. Luettu: 19.4.2015.

Oracle Technology Network 2014. Java Security Resource Center. Oracle. Luettavissa: http://www.oracle.com/technetwork/java/javase/overview/security-2043272.html. Luettu: 14.4.2015.

Oracle Technology Network 2014. Java SE Security. Oracle. Luettavissa: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html. Luettu: 14.4.2015.

OWASP 2015. Cross-Site Request Forgery (CSRF). Luettavissa: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29. Luettu: 27.4.2015.

OWASP 2014. Cross-site Scripting (XSS). Luettavissa: https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29. Luettu: 20.4.2015.

pool-70-107-200-115.ny325.east.verizon.net 2007. Internet Programming Languages Pros And Cons. Cunningham & Cunningham, Inc. Luettavissa: http://c2.com/cgi/wiki?InternetProgrammingLanguagesProsAndCons. Luettu 4.6.2015.

Rantanen, T. 2012. Pilvipalvelut. Tampereen Ammattikorkeakoulu. Luettavissa: https://publications.theseus.fi/bitstream/handle/10024/41986/Rantanen_Tapio.pdf?sequence=1. Luettu: 23.2.2015.

Rouse, M. 2005a. Blackhole list (blacklist). WhatIs.com. Luettavissa: http://searchexchange.techtarget.com/definition/blacklist. Luettu: 26.4.2015.

Rouse, M. 2005b. CSS. WhatIs.com. Luettavissa: http://searchsoa.techtarget.com/definition/cascading-style-sheet-CSS. Luettu: 19.3.2015.

Rouse, M. 2005c. JAR file (Java ARchive). WhatIs.com. Luettavissa: http://searchdatamanagement.techtarget.com/definition/JAR-file. Luettu: 15.4.2015.

Rouse, M. 2005d. JavaEE. WhatIs.Com. Luettavissa: http://searchsoa.techtarget.com/definition/J2EE. Luettu: 24.3.2015.

Rouse, M. 2005e. Java Server Page (JSP). WhatIs.Com. Luettavissa: http://searchsoa.techtarget.com/definition/Java-Server-Page. Luettu: 24.3.2015.

Rouse, M. 2005f. Whitelist. WhatIs.com. Luettavissa: http://searchexchange.techtarget.com/definition/whitelist. Luettu: 26.4.2015.

Rouse, M. 2007. Java. WhatIs.Com. Luettavissa: http://searchsoa.techtarget.com/definition/Java. Luettu: 19.3.2015.

Rouse, M. 2008. Principle of least privilege (POLP). WhatIs.com. Luettavissa: http://searchsecurity.techtarget.com/definition/principle-of-least-privilege-POLP. Luettu: 14.4.2015.

Rouse, M. 2011. Request for Comments (RFC). Luettavissa: http://whatis.techtarget.com/definition/Request-for-Comments-RFC. Luettu: 27.4.2015

Rouse, M. 2014a. HTML5. WhatIs.com. Luettavissa: http://searchsoa.techtarget.com/definition/HTML5. Luettu: 19.3.2015.

Rouse, M. 2014b. network-attached storage (NAS). Luettavissa: http://searchstorage.techtarget.com/definition/network-attached-storage. Luettu: 22.4.2015.

Sleavely, 2015. HTTP access controll (CORS). Mozilla Developer Network. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS. Luettu 17.3.2015.

Sulopuisto, O. 2013. Pilvipalveluissa on useita porsaanreikiä. Helsingin Sanomat. Luettavissa: http://www.hs.fi/tekniikka/a1357891415247. Luettu: 25.2.2015.

TechNet 2015. Hyödynnä pilvipalvelua viisaasti. Microsoft. Luettavissa: https://technet.microsoft.com/fi-fi/ff934854.aspx. Luettu: 17.3.2015.

Techopedia. User Agent (UA). Luettavissa: http://www.techopedia.com/definition/1614/user-agent-ua. Luettu: 27.4.2015.

Techterms.com. 2015. ISP. Luettavissa: http://techterms.com/definition/isp. Luettu: 27.4.2015.

Tiger Team Blogi 2010, Pilvipalvelujen tietoturvallisuudesta. Luettavissa: http://www.nixu.com/fi/download/95/pilvipalvelujen-tietoturvallisuudesta___.pdf?redirect=node/218. Luettu: 23.2.2015.

W3C 2015. About W3C. W3C. Luettavissa: http://www.w3.org/Consortium/. Luettu: 27.4.2015.

W3School.com. HTML5 Web Workers. W3School.com. Luettavissa: http://www.w3schools.com/html/html5_webworkers.asp. Luettu 17.3.2015.

Liite 1 - Käyttäjäoikeudet

1 Yleiset pääkäyttäjäoikeudet

Alla olevassa taulussa on yleiset pääkäyttäjäoikeudet ja niiden selitykset. Yleiset pääkäyttäjäoikeudet koskevat kaikkia tietokantoja ja niiden tauluja MariaDB tietokannan sisällä.

Oikeus	Selitys
CREATE USER	Kyky luoda käyttäjiä em. lauseen avulla
FILE	Kyky käyttää LOAD DATA INFILE –lausetta ja LOAD_FILE( )-funtiota
PROCESS	Kyky käyttää SHOW PROCESSLIST komentoa
RELOAD	Kyky käyttää FLUSH lausetta
REPLICATION CLIENT	Kyky käyttää SHOW MASTER STATUS ja SHOW SLAVE STATUS komentoja
REPLICATION SLAVE	Kyky saada päivityksiä, jotka on tehty replikointipääpalvelimelle
SHOW DATABASES	Kyky listata kaikki palvelimen tietokannat
SHUTDOWN	Kyky ajaa palvelin alas käyttäen **mysqladmin shutdown** komentoa
SUPER	Kyky käyttää superkäyttäjä komentoja kuten CHANGE MASTER TO…, PURGE LOGS; muuttaa (SET) paikallisia muuttujia, tappaa (KILL) muiden käyttäjien ketjuja

1.1 Tietokanta-, taulukko- ja sarakeoikeudet

Alla olevassa taulussa on tietokanta ja tauluoikeudet. Nämä oikeudet soveltuvat ainoastaan tiettyyn tietokantaan tai tauluun tietokannan sisällä.

Oikeus	Selitys
ALTER	Kyky muuttaa indeksejä ja taulukoita
ALTER ROUTINE	Kyky muuttaa tai poistaa proseduureja ja tallentaa funktioita
CREATE	Kyky luoda tietokantoja ja taulukoita
CREATE ROUTINE	Kyky luoda proseduureja ja tallentaa funktioita

Oikeus	Selitys
CREATE TEMPORARY TABLES	Kyky luoda väliaikaisia tauluja
CREATE VIEW	Kyky luoda näkymiä
DELETE	Kyky poistaa rivejä taulukoista
DROP	Kyky poistaa kokonaisia tietokantoja ja taulukkoja
EVENT	Kyly muuttaa, luoda tai poistaa tapahtumia tapahtuma-aikatauluttajasta
EXECUTE	Kyky suorittaa tallennettuja funktioita ja proseduureja
INDEX	Kyky luoda ja poistaa indeksejä
INSERT	Kyky lisätä uusia rivejä tietoa taulukkoon
LOCK TABLES	Kyky lukita ja avata taulukoita
SELECT	Kyky lukea tietoa taulukosta
SHOW VIEW	Kyky käyttää SHOW CREATE VIEW komentoa
TRIGGER	Kyky käyttää CREATE TRIGGER ja DROP TRIGGER komentoja
UPDATE	Kyky muokata taulukon rivejä

Sarakeoikeudet soveltuvat yksittäisiin sarakkeisiin taulukossa. Näitä komentoja on ainoastaan kolme: INSERT, UPDATE ja SELECT.

1.2 Sekalaiset oikeudet ja rajoitukset

Oikeus	Selitys
USAGE	Ei todellisuudessa myönnä mitään, mutta voidaan käyttää muuttamaan käyttäjän globaaleja oikeuksia
ALL PRIVILEGES	Voidaan käyttää myöntämään kaikki oikeuden käyttäjälle. Ei myönnä GRANT OPTION oikeutta. Voidaan lyhentää ALL.
GRANT OPTION	Antaa käyttäjälle kyvyn antaa toisille käyttäjille oikeuksia. Tämä komento annetaan GRANT-komennon lopussa.

On myös useita rajoitteita joita voimme laittaa käyttäjän tilille. Nämä annetaan seuraavasti:

Rajoite	Selitys
MAX_QUERIES_PER_HOUR	SQL komentojen tai kyselyjen määrä, jonka käyttäjä voi suorittaa tunnissa. Tämä sisältää päivitykset.
MAX_UPDATES_PER_HOUR	SQL päivitysten (ei kyselyiden) määrä, jonka käyttäjä voi suorittaa tunnissa.
MAX_CONNECTIONS_PER_HOUR	Yhteyksien määrä, jonka käyttäjä voi aloittaa tunnissa.
MAX_USER_CONNECTION	Yhtä aikaisten yhteyksien määrä, joka käyttäjällä voi olla tietokantapalvelimeen. Jos asetettu nollaan, silloin sama kuin **max_connections** asetus. Jos **max_connections** asetus on myös asetettu nollaan, ei ole rajoitusta yhtäaikaisille yhteyksille.

Liite 2 - Terminologia

Aliverkko	Aliverkko on loogisen tietokoneverkon osa, joka sijaitsee OSI-mallin kolmannella kerroksella (verkkokerros). Aliverkotus on termi, jota käytetään kun pilkotaan suurempi verkko pienempiin osiin, aliverkkoihin.
Amazon	Maailman suurin verkkokauppa
Auditointi	Auditointi on määrämuotoinen ja objektiivinen arviointi sen havaitsemiseksi, onko auditoinnin kohteelle asetetut vaatimukset täytetty.
Blacklist	Blackhole list, tunnetaan myös nimellä Blacklist, on yleensä ottaen julkaisu tunnetuista ISP-osoitteista, jotka lähettävät SPAM-viestejä. Tässä tapauksessa lista palvelimia, joilta estetään liikenteen vastaanotto.
Chmod	Chmod (change mode) on Unix-ohjelma, jolla muutetaan tiedostojen ja hakemistojen käyttöoikeuksia.
CORS	Ristikkäisalkuperäresurssijako tarkoittaa sitä, että tehdään HTTP rersurssipyynnöt toiselle palvelimelle kuin se resurssipalvelin, joka lähettää pyynnön
CSRF	Poikittaissivupyyntöväärennös (Cross-Site Request Forgery) on hyökkäys, joka pakottaa loppukäyttäjän suorittamaan epätoivottuja toimenpiteitä nettiohjelmassa, jossa he ovat tällä hetkellä todennettuina.
CVE	Common Vulnerabilities and Exposures – Yleiset haavoittuvuudet ja vastuut kirjasto/sanakirja
DNS	Nimipalvelujärjestelmä, joka muuntaa verkkotunnuksia IP-osoitteiksi
Dropbox	Suosittu verkkotilan tarjoaja

FQDN	Täydellinen toimialuenimi, on toimialuenimi, joka määrittelee tarkan sijainnin nimipalvelujärjestelmän (DNS) puuhierarkiassa.
FTP	FTP avulla voidaan siirtää tiedostoja kahden tietokoneen välillä internetissä. FTP on yksinkertainen verkkoprotokolla, joka perustuu internet protokollaan. FTP tulee sanoista File Transfer Protocol.
Gmail	Suosittu sähköpostipalveluntarjoaja
Google	Maailman suosituin hakupalvelin
IPv4	IP (engl. Internet Protocol[1]) on TCP/IP-mallin Internet-kerroksen protokolla, joka huolehtii IP-tietoliikennepakettien toimittamisesta perille pakettikytkentäisessä Internet-verkossa.
ISP	Tulee sanoista Internet Service Provider. Jotta on mahdollista olla yhteydessä internetiin, tarvitaan ISP. ISP on yritys, joka tarjoaa palvelua, jonka kautta saa yhteyden internetiin. Tällaisia Suomessa ovat Sonera (TeleFinland), Elisa (Saunalahti) ja DNA.
JAR-tiedosto	JAR (**J**ava **AR**chive) tiedosto sisältää Javaohjelman luokan, kuvan tai äänitiedoston tai appletin, jotka on kerätty yhteen tiedostoon ja mahdollisesti pakattu.
Java	Java on Sun Microsystemsin kehittämä laaja teknologiaperhe ja ohjelmistoalusta, johon kuuluu muun muassa laitteistoriippumaton oliopohjainen ohjelmointikieli sekä ajoaikainen ympäristö virtuaalikoneineen ja luokkakirjastoineen.
Komentorivi	Komentoliittymä, komentorivi eli komentokehote (engl. command line interface, CLI) on tapa järjestää ihmisen ja tietokoneen välinen kommunikointi. Vaikka komentoliittymää voisi pitää graafisen käyttöliittymän vastakohtana, käytännössä vertailu ei ole aivan osuva.

Kryptaus	Salaus viittaa kryptologiassa prosessiin, jolla koodataan viestejä tai tietoja niin, että vain valtuutetut osapuolet voivat lukea niitä.
Käyttöliittymä	Käyttöliittymä on se laitteen, ohjelmiston tai minkä tahansa muun tuotteen osa, jonka kautta käyttäjä käyttää tuotetta.
Käyttäjäagentti	User agent on ohjelmaelementti, joka toimii käyttäjän puolesta. Monesti se tarkoittaa ohjelmaa, jota toimii taustalla websivuja ajettaessa, eikä vaikuta itse sivun toimintaan.
MariaDB	Tietokantapalvelinohjelmisto
NAS-palvelin	Verkkoon liitetty tietovarasto, joka käyttää vakio Ethernet liittymää tarjotakseen verkkosolmuja tietopohjaisille jaetuille tietovarastopalveluille
Ohjelmointi	Ohjelmointi tarkoittaa tietokoneelle tai vastaavalle laitteelle jollakin tavalla annettavia toimintaohjeita.
Osoiteavaruus	IPv4 jaettiin aiemmin neljään erilaiseen osoiteluokkaan. Nämä luokat olivat A-luokka, B-luokka, C-luokka ja varatut verkot. Nykyään tällaista jakoa ei enää ole, mutta termejä käytetään viittaamaan aliverkon kokoon.
Palvelin	Palvelimella (ark. serveri) tarkoitetaan tietoliikenteen yhteydessä tietokoneessa suoritettavaa palvelinohjelmistoa sekä tällaista ohjelmistoa suorittavaa tietokonetta.
Palvelinräkki	Palvelinräkki on kaappi, jossa on useita palvelimia.
PHP	PHP (lyhenne sanoista PHP: Hypertext Preprocessor) on Perlin kaltainen ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten web-sivujen luonnissa.
Protokolla	Protokolla eli yhteyskäytäntö on käytäntö tai standardi, joka määrittelee tai mahdollistaa laitteiden tai ohjelmien väliset yhteydet.

Reflected XSS	Heijastetut XSS hyökkäykset ovat sellaisia hyökkäyksiä, jotka kohdistuvat pois webserveristä, esimerkiksi error viesteihin, hakutuloksiin, sekä muihin vastauksiin, jossa osa tai kaikki syöte on osana vastausta.
RFC	Request for Comment on Internet Engineering Task Forcen (IETF) tekemä muodollinen dokumentti, joka on asianomaisten osapuolten tarkastelun sekä komitean valmistelun tulos.
Root-käyttäjä	Tietokannan pääkäyttäjä
Sisäverkko	Lähiverkko (LAN, Local Area Network) on rajatun alueen nopea tietoliikenneverkko.
SPAM	Epätoivottua sähköpostia. Sähköpostia, jota lähetetään suurina määrinä ja joka sisältää pääosin mainoksia
SSH	Secure Shell eli SSH on salattuun tietoliikenteeseen tarkoitettu protokolla.
Spotify	Ehkä maailman suosituin musiikkipalvelu
TCP	Transmission Control Protocol on yksi pääprotokolla TCP/IP-yhteydenpitoprotokollasarjassa, jota käytetään liittämään isäntäkone internetiin ja monet muut tietokoneet myös verkkoon.
Thin client	Tyhmä pääte, jossa ei ole edes kovalevyä, vaan se hakee kaiken tiedon palvelimelta. Nykyään on myös päätteitä, joissa on käyttöjärjestelmä, mutta muut tiedot haetaan palvelimelta.
Tietokanta	Tietokanta on tietotekniikassa käytetty termi tietovarastolle. Se on kokoelma tietoja, joilla on yhteys toisiinsa.
TLS	Transform Layer Security on protokolla, joka varmistaa yksityisyyden ohjelman ja sitä käyttävän käyttäjän välillä internetissä.

Toimialue	Toimialueet, työryhmät ja kotiryhmät edustavat eri tapoja järjestää tietokoneet verkoissa. Tärkein ero niiden välillä on se, miten verkkojen tietokoneita ja muita resursseja hallitaan.
Tunneli	kts. VPN
UPS	Uninterruptible Power Supply (UPS) on järjestelmä tai laite, jonka tehtävä on taata tasainen virransyöttö lyhyissä katkoksissa ja syöttöjännitteen epätasaisuuksissa.
VNC	Virtual Network Computing protokolla on yksinkertainen protokolla, jonka avulla saadaan graafisia etäkäyttöliittymiä.
VPN	VPN (Virtual Private Network) eli virtuaalinen erillisverkko on tapa, jolla kaksi tai useampia yrityksen verkkoja voidaan yhdistää julkisen verkon yli muodostaen näennäisesti yksityisen verkon.
W3C	World Wide Web Consortium (W3C) on kansainvälinen yhteisö, jossa yhteisöjäsenet, täysaikaiset työntekijät ja vapaaehtoiset työntekijät yhdessä kehittävät internetstandardeja.
Web	World Wide Web tai WWW on Internet-verkossa toimiva hajautettu hypertekstijärjestelmä.
Web Workers	Web Worker on JavaScript, jota ajetaan taustalla, riippumatta muista ohjelmista, ja joka ei vaikuta sivun toiminnallisuuteen.
Whitelist	Whitelist on yleensä ottaen lista e-mail-osoitteita tai palvelinnimiä, joilta sallitaan viestien vastaanottaminen. Tässä tapauksessa myös muun liikenteen vastaanottaminen.
XSS	Cross-site Scripting, eli poikittaissivustokoodaus, joka tarkoittaa sitä, että huonosti koodattuun lomakkeeseen syötetään hyökkäyskoodi, jonka avulla päästään käsiksi asiakas-koneeseen.

Liite 3 - Kuvakaappaukset

Pääkäyttäjämoduuli – taulunmuokkaus

Pääkäyttäjä voi muokata taulua joko taulukkomuodossa tai lomakemuodossa. Taulukkomuodossa pääkäyttäjä voi muokata ja poistaa tietueen ja lomake muodossa muokata, poistaa ja lisätä tietueen. Alla kuvakaappauksia eri tapauksista:

TUOTEYLLÄPITO

	TuoteID	Tuotenimi	TuoteryhmäID	TuotekuvausID
Muokkaa Poista Valitse	1	HXR-70 / 1	1	1
Muokkaa Poista Valitse	2	HXR-70 / 2	1	1
Muokkaa Poista Valitse	3	HXR-70 / 3	1	1
Muokkaa Poista Valitse	4	HXR-70 / 4	1	1
Muokkaa Poista Valitse	5	HDV / 1	2	2
Muokkaa Poista Valitse	6	HDV / 2	2	2
Muokkaa Poista Valitse	7	HDV / 3	2	2
Muokkaa Poista Valitse	8	HDV / 4	2	2
Muokkaa Poista Valitse	9	D90 / 1	3	3
Muokkaa Poista Valitse	10	D90 / 2	3	3
				1 2 3

Kuva 1. Ylläpitotaulukko, peruslähtökohta.

TUOTEYLLÄPITO

	TuoteID	Tuotenimi	TuoteryhmäID	TuotekuvausID
Päivitä Peruuta	1	HXR-70 / 1 uusi kamera	1	1
Muokkaa Poista Valitse	2	HXR-70 / 2	1	1
Muokkaa Poista Valitse	3	HXR-70 / 3	1	1
Muokkaa Poista Valitse	4	HXR-70 / 4	1	1
Muokkaa Poista Valitse	5	HDV / 1	2	2
Muokkaa Poista Valitse	6	HDV / 2	2	2
Muokkaa Poista Valitse	7	HDV / 3	2	2
Muokkaa Poista Valitse	8	HDV / 4	2	2
Muokkaa Poista Valitse	9	D90 / 1	3	3
Muokkaa Poista Valitse	10	D90 / 2	3	3
				1 2 3

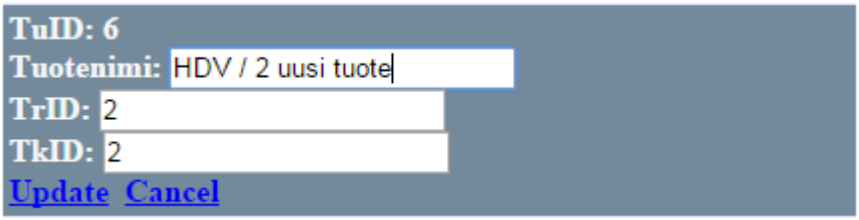
Kuva 2. Ylläpitotaulukko, taulukkomuokkausnäkymä.

TUOTEYLLÄPITO

	TuoteID	Tuotenimi	TuoteryhmäID	TuotekuvausID
Muokkaa Poista Valitse	1	HXR-70 / 1	1	1
Muokkaa Poista Valitse	2	HXR-70 / 2	1	1
Muokkaa Poista Valitse	3	HXR-70 / 3	1	1
Muokkaa Poista Valitse	4	HXR-70 / 4	1	1
Muokkaa Poista Valitse	5	HDV / 1	2	2
Muokkaa Poista Valitse	**6**	**HDV / 2**	**2**	**2**
Muokkaa Poista Valitse	7	HDV / 3	2	2
Muokkaa Poista Valitse	8	HDV / 4	2	2
Muokkaa Poista Valitse	9	D90 / 1	3	3
Muokkaa Poista Valitse	10	D90 / 2	3	3
				1 2 3

TuID: 6
Tuotenimi: HDV / 2
TrID: 2
TkID: 2
Edit Delete New

Kuva 3. Ylläpitolomake, perusnäkymä

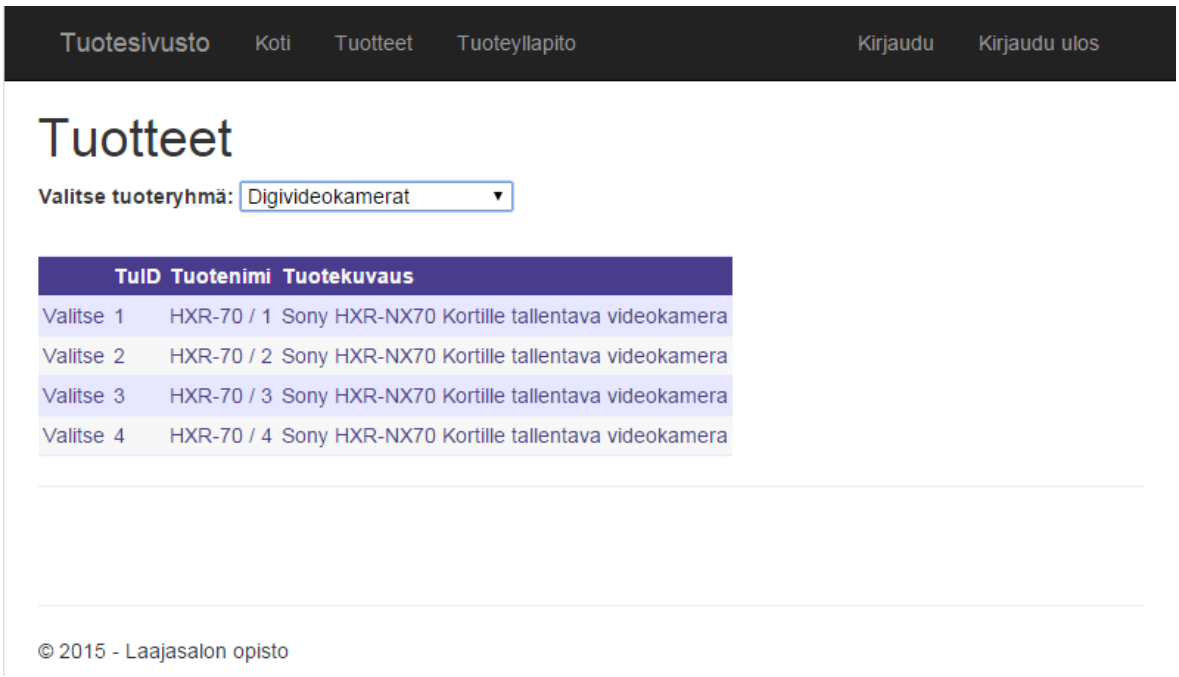


Kuva 4. Lomakemuokkausnäkymä

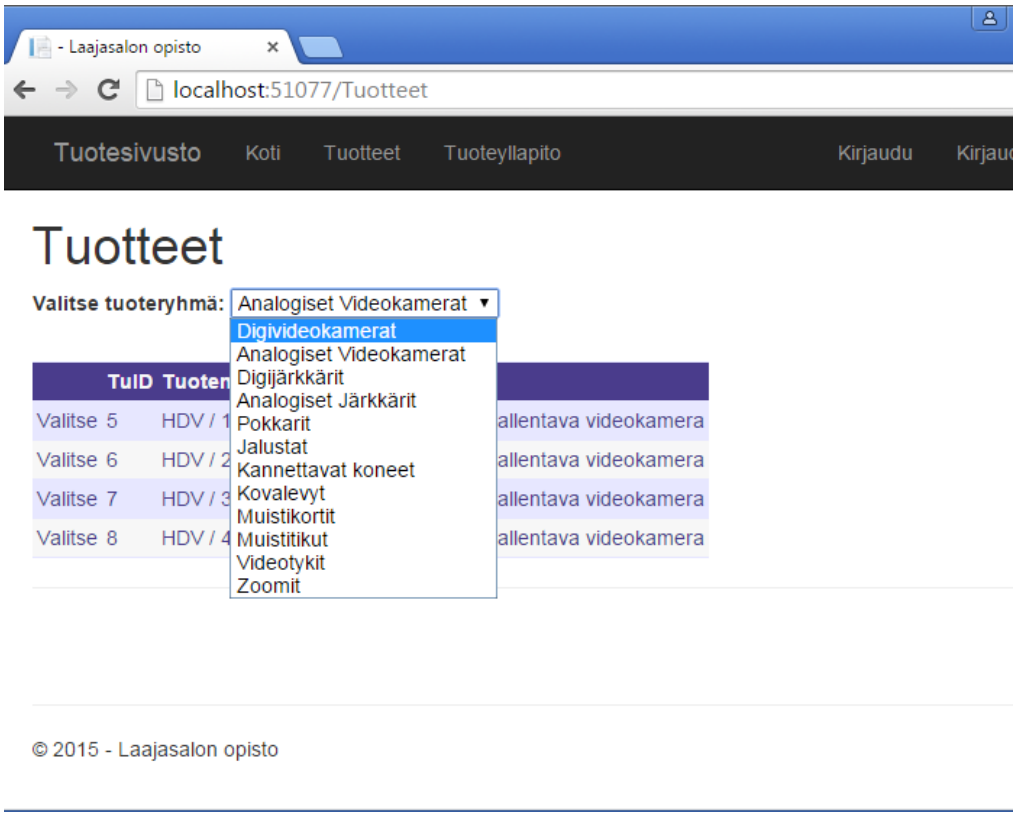
Kuva 5. Uuden tuotteen syöttönäkymä

Peruskäyttäjä – tilaus

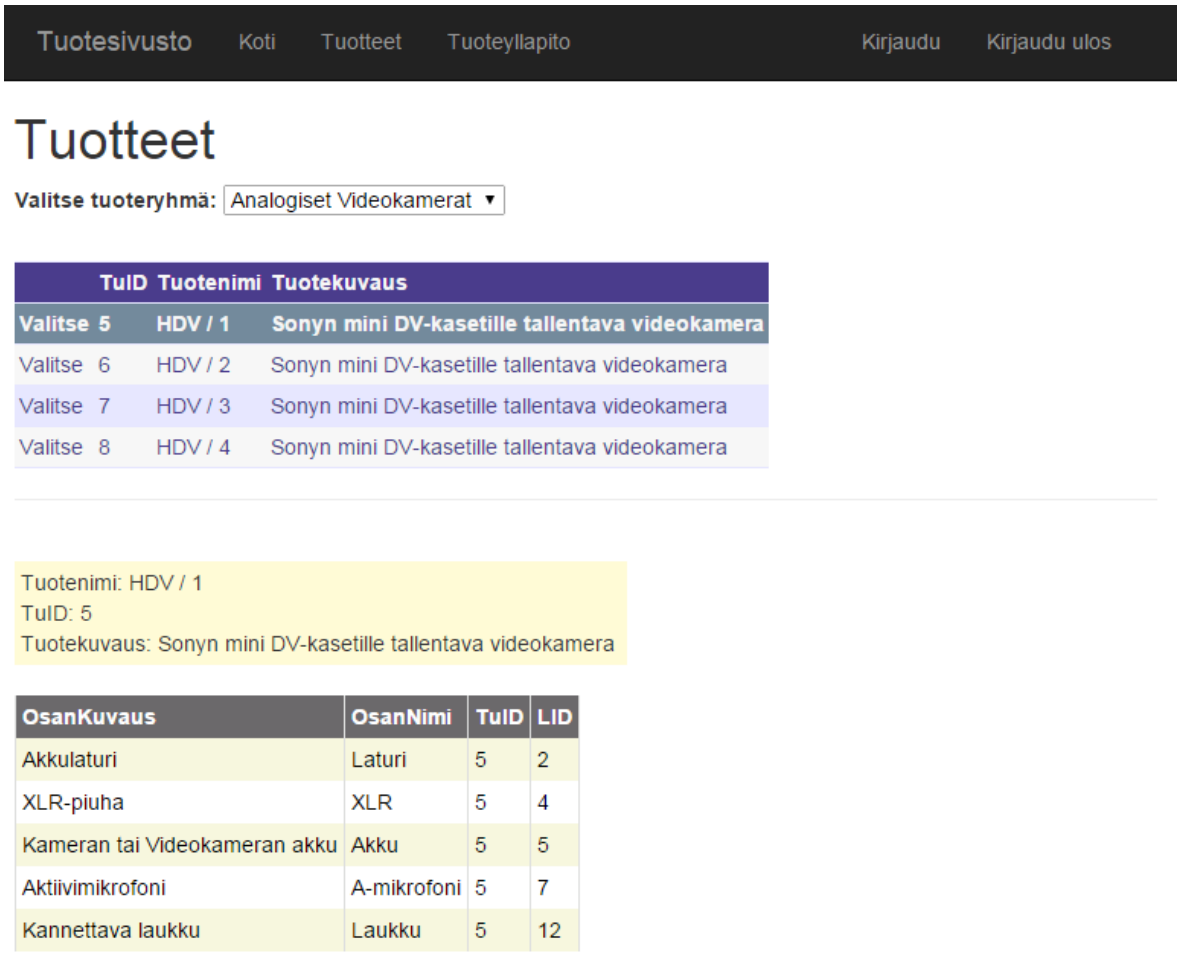
Peruskäyttäjänäkymässä peruskäyttäjä näkee alaspudotusvalikon, jossa on tuoteryhmäluettelo, josta asiakas voi valita tuoteryhmän, sekä sen alla olevan tuoteryhmätaulukon, joka päivittyy sen mukaan, mitä asiakas valitsee em. luettelosta. Lisäksi, kun asiakas valitsee tuoteryhmätaulukosta tuotteet, tulee tuoteryhmätaulukon alapuolelle lomake, jossa näkyy ko. tuotteen tiedot, sekä ko. tuotteen lisäosat omana taulukkonaan. Alla olevissa kuvakaappauksissa tämä sama on kuvina.

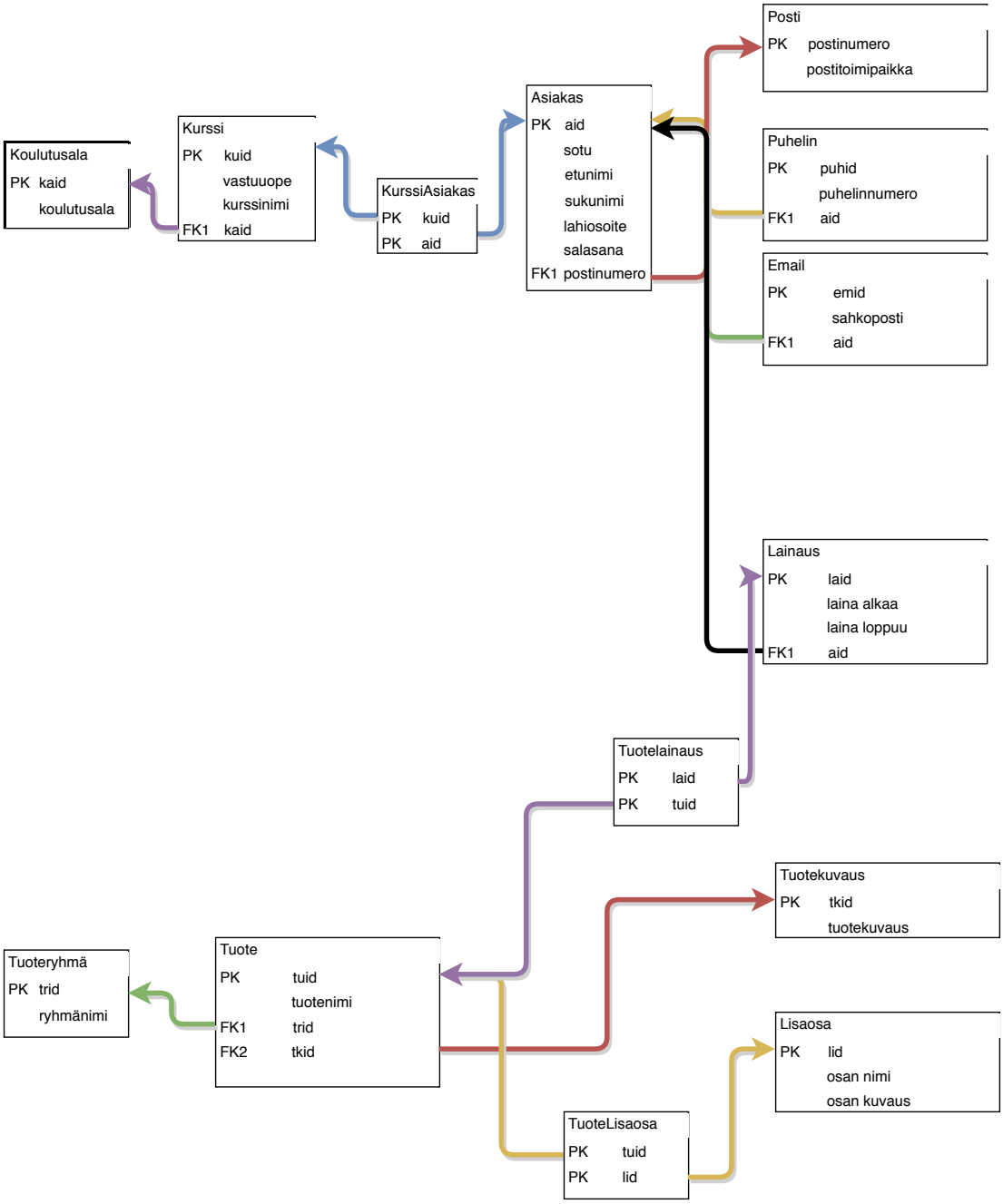


Kuva 6. Käyttäjätaulukko, perusnäkymä



Kuva 7. Käyttäjätaulukko, alasvetovalikkonäkymä





Kuva 1. Tietokannan kaaviokuva.