Sevilay Yurdakul

Tietoturvallinen web-ohjelmointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

5.4.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Sevilay Yurdakul Tietoturvallinen web-ohjelmointi 39 sivua + 9 liitettä 5.4.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaaja(t)	Lehtori Ilpo Kuivanen

Tässä opinnäytetyössä selvitettiin web-ohjelmoinnin tietoturvaa. Aluksi käytiin läpi tietoturvallisuuden perusteet ja sitten OWASP Top 10 -listan kolme yleisintä tietoturvauhkaa, jotka ovat SQL-injektio, XSS-hyökkäys ja istunnon kaappaus. Viimeisenä käsiteltiin uhkien estämiskeinoja.

Opinnäytetyön tarkoituksena oli antaa tietoa ohjelmoijalle siitä, miten verkkosovellukset altistuvat hyökkäyksille ja miten voidaan kehittää tietoturvallisempia verkkosovelluksia. Työ rajoitettiin ohjelmointikieliseen PHP:hen ja tietokannoista MySQL:ään. Keskeisiä tutkimusalueita olivat muun muassa tietoturvauhat ja niiden ehkäiseminen.

Opinnäytetyössä kerrottiin syötteiden tarkistamisen ja salasanojen tiivistämisen tärkeydestä. Suurin osa haavoittuvuuksista sivuilla johtuu nimenomaan huolimattomasta syötteiden tarkistamisesta. Salasanojen tiivistämisessä on tärkeä käyttää vahvaa standardialgoritmia ja suolaa, joka tekee tiivisteistä huomattavasti vaikeampia murtaa.

Työssä käsiteltiin myös web-palvelimen ja MySQL-tietokannan turvaamista. Verkkoselaimessa yhteyden suojauksessa käytetään yleisesti SSL- ja HTTPS-salausprotokollia. Tietokannan uhkien yleisimpiä torjuntatapoja ovat käyttöoikeuksien rajoittaminen ja varmuuskopioiden ottaminen.

Tietoturvallisuuden lisääminen ohjelmiston ominaisuudeksi jälkikäteen on huomattavan hankalaa, joten on suotavaa ottaa se mukaan alusta alkaen. Kaiken kaikkiaan tietoturvallisuus pitää ottaa huomioon koko ohjelmistokehitysprosessissa.

Avainsanat	tietoturva, web-ohjelmointi, PHP, MySQL

Author(s) Title Number of Pages Date	Sevilay Yurdakul Secure Web Programming 39 pages + 9 appendices 5 April 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Ilpo Kuivanen, Senior Lecturer

This thesis examines the information security of web programming. At first the study goes through the basis of information security and 3 of the most common threats listed in OWASP Top 10 list: SQL injection, XSS attack, session hijacking. Finally the thesis shows ways to prevent these threats.

The purpose of this thesis was to offer informaation for programmers about how the web application are exposed to attacks and how they can develop a secure web application. The study was limited to the programming language PHP and database MySQL. The methodology of this thesis was largely related security threats and their prevention.

This thesis is explanes the importance of checking inputs and hashing passwords. Especially because most of the vulnerabilities in web pages are due to sloppy checking of user input. In hashing passwords it is crucial to use standard algorithm and salt which make the hash much more harder to break.

Web server and MySQL database securing was also discussed in this thesis. SSL and HTTPS were used to secure the connection in internet browsers. The most commonly used ways to prevent information security threats is to limit user priviledges and to take back ups.

It is very hard to implement information security to a software so it is crucial to recognize it from the early start of the project. Information security should be included in all areas of software development.

Keywords	information security, web programming, PHP, MySQL

Sisällys

Liitteet

Liite 1. Tietokanta

Liite 2. Sovelluksen lähdekoodi

Lyhenteet ja määritelmät

Apache	Avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma.
ASP	Active Server Pages. Microsoftin palvelimilla käytettävä ohjelmointikieli.
CSS	Cascading Style Sheets. HTML-kielen rinnalla toimiva tyylikieli.
HTML	Hypertext Markup Language. Internet-sivujen teossa käytettävä merkkauskieli.
HTTP	Hypertext Transfer Protocol. Protokolla, jonka avulla selain ja WWW-palvelin keskustelevat.
HTTPS	Hypertext Transfer Protocol Secure. HTTP-protokollan ja SSL/TLS-protokollan yhdistelmä.
JavaScript	Netscapen kehittämä oliopohjainen ohjelmointikieli.
MD5	Datan eheyden tarkistamiseen tehty tarkistussumma.
MySQL	WWW-sovelluksissa käytettävä ilmainen tietokanta.
OWASP	Open Web Application Security Project. Vapaaehtoisjärjestö.
PERL	Practical Extraction and Report Language. Ohjelmointikieli, jota käytetään paljon Unix-ympäristöissä ja etenkin CGI-ohjelmien toteutukseen.
PDO	PHP Data Objects. Tietokantojen käsittelyyn tarkoitettu rajapinta PHP:lle.
PHP	PHP: Hypertext Preprocessor. Ilmainen, palvelinpohjainen skriptikieli, jota käytetään usein toiminnallisten www-sivujen toteuttamiseen.
SHA	Secure Hash Algorithm. Tiivistealgoritmi.
SSL	Secure Sockets Layer. Salausprotokolla.

SQL	Structured Query Language. Relaatiotietokantojen kyselykieli.
UPS-laite	Uninterruptible Power Supply. Suojelee IT-laitteita kriittisimmiltä sähköhäiriöiltä.
URL	Uniform Resource Locator. Määrittää WWW:ssä olevien tietojen osoitteet selkeästi ja yksiselitteisesti.
W3C	World Wide Web Consortium. Kehittää yhteisiä ja yhteensopivia Webin pelisääntöjä ja teknologioita.
WWW	World Wide Web. Internet-verkossa toimiva hajautettu hypertekstijärjestelmä.
XHTML	EXtensible HyperText Markup Language. HTML:stä kehitetty XML-pohjainen koodikieli.
XSS	Cross-site scripting. Tietoturva-aukko, jonka avulla web-sivustolla voidaan ajaa ulkoista selainpuolen koodia, käytännössä JavaScriptiä.

1 Johdanto

Tämän insinöörityön tarkoituksena on tutkia tietoturvallista web-ohjelmointia. Työssä tutustutaan tietoturvaan liittyviin keskeisiin ongelmiin ja uhkiin sekä niiden käytännönläheisiin ratkaisuihin.

Nykyään verkkomaksaminen on monelle jo arkipäiväistä rutiinia, esimerkiksi ostamme lentolippuja, vaatteita, kenkiä ja maksamme laskuja. Verkko on tuonut useat erilaiset kaupat sekä kauppiaat kuluttajille vain muutaman klikkauksen päähän. Tämä on antanut paljon uusia mahdollisuuksia, mutta ongelmaksi nousee, kuinka liikutaan tietoturvallisesti verkossa. Valitettavasti verkko houkuttelee myös huijareita, kiusaajia ja häiriköitä, jotka aiheuttavat mielipahan lisäksi taloudellisia ja sosiaalisia ongelmia.

Tietoturva on monelle epäselvä termi, jota käsitellään tarkemmin kappaleessa 2.1., mutta toisaalta se on nykyaikana tärkeä ja suuri asia verkkosivuilla. Internet-sivua tehdään käyttäen dynaamisia ohjelmointiteknikoita, eivätkä saa tietoturvariskiä itsenäisenä. Ongelma syntyy siinä tilanteessa, kun yritetään ottaa vastaan käyttäjän lähettämää dataa. Sen lisäksi suurin osa ongelmista perustuu siihen, kuinka hyvin ohjelma osaa erottaa oikeaa ja väärää dataa. [1.] Tietoturvaongelmat johtuvat yleensä tietämättömyydestä ja huolimattomuudesta.

Työn tavoitteena on kehittää tietoturvallinen verkkosovellus ja sen testata sitä erilaisilla tekniikoilla. Työn tarkoitus ei ole käydä läpi sitä, että miten tehdään verkkosivuja, vaan työssä keskitytään verkkosivujen tietoturvaan. Työ vastaa tutkimuskysymykseen ”mitä on tietoturvallinen web-ohjelmointi ja miten sitä kehitetään”. Työn aihetta rajataan palvelinpuolen PHP- (Hypertext Preprocessor) ja MySQL- (My Structured Query Language) ohjelmointikieliin. Työssä toteutetaan yksinkertaisia tietoturvallisia PHP-sivuja, jotka käyttävät MySQL-tietokantaa. Työssä keskitytään sivulle rekisteröintiin ja sisäänkirjautumiseen.

2 Taustat

Tässä luvussa käsitellään insinöörityön taustoja, kuten tietoturvaa, web-ohjelmointia ja ohjelmointikieliä PHP ja MySQL.

2.1 Tietoturva

Tietoturvalla tarkoitetaan tietojen, järjestelmien ja palveluiden suojaamista erilaisilta tietoturvariskeiltä. Suojaaminen onnistuu sekä normaali- että poikkeusoloissa hallinnollisilla ja teknisillä toimenpiteillä, joilla turvataan tiedon luottamuksellisuus (confidentiality), saatavuus (availability) ja eheys (integrity). Näiden lisäksi peruselementeiksi luetaan myös todennus, kiistämättömyys ja pääsynvalvonta. [2.] Tietoturva ei koskaan ole perusteellista, vaan se on jonkinasteisten kompromissien summa. Edellä mainittujen käsitteiden avulla määritellään, mikä on suojattavalle kohteelle tärkeintä. [3.]

Hyvä tietoturvan taso riippuu siitä, toimivatko tietojärjestelmät häiriöttömästi ja pysyykö niiden kontrolli omissa käsissä. Hyvällä tietoturvatasolla säästetään aikaa ja rahaa sekä lisätään myös yrityksen uskottavuutta. [4.]

2.1.1 Luottamuksellisuus

Luottamuksellisuudella varmistetaan, että tiedot ovat vain niiden käytettävissä, joille ne on tarkoitettu. Tieto on vain siihen oikeutettujen luetettavissa tai muokattavissa. Valtuutettujen käyttäjien tunnistamiseksi heidät täytyy ensin todentaa. Tiedon suojaamiseksi ulkopuolisilta tarvitaan salausta. Riittävän turvallisella salausmenetelmällä siirrettävä tai tallennettava tieto ei paljastu, jos joku salakuuntelee tiedonsiirtoyhteyttä tai varastaa tallennuksessa käytetyn koneen. [5, s. 22.]

2.1.2 Saatavuus

Tietojen ja palvelun saatavuus liittyy tietojärjestelmien toiminnan turvaamiseen. Saatavuus tarkoittaa todennäköisyyttä, että järjestelmä on tiettynä ajan hetkenä toiminnassa ja pystyy tarjoamaan käyttäjän haluamat palvelut silloin, kun käyttäjä ne haluaa.

Tarkoituksellinen ylikuormitus voi häiritä palvelun saatavuutta esimerkiksi palvelunestohyökkäystä, jossa hyökkääjä estää palvelun normaalin toiminnan. [5, s. 24.] Normaalitilanteissa järjestelmässä ei voi olla pullonkauloja eikä normaali päivittäinen käyttö saa aiheuttaa saatavuusongelmia. On huomattava, että järjestelmien ja palvelinsovellusten päivitykset ovat ajan tasalla, sekä yleinen tietoturva hyvällä mallilla. Sovelluksen pullonkaulat ja tietoturvaongelmat olisi hyvä selvittää suorituskykytestauksilla.

Merkittävin tapa saatavuuden varmistamiseen on tiedostojen varmuuskopiointi ja luotettavien verkkoyhteyksien muodostaminen sekä laitteiden toiminta turvaava tekniikka, kuten vikasietoinen RAID-järjestelmä ja UPS- (Uninterruptible Power Supply) laite. [5, s. 24.] RAID:n idea on saada monista kiintolevyistä tehtyä yksi looginen asema, jonka suorituskyky ja vikasietoisuus on parempi kuin yksittäisillä levyillä. Yleisesti RAIDia käytetään palvelimissa, joilta vaaditaan suurta datansiirtokykyä ja vikasietoisuutta.
RAID-järjestelmällä on eri RAID-tasoa, joista käytetyimmät RAID-tasot ovat RAID0,RAID1,RAID10 ja RAID5. Esimerkiksi RAID 5 taso käytetään yleisesti palvelimissa ja sen kokoonpano kestää yhden fyysisen levyn täydellisen vikaantumisen. [6.] UPS-laitteen tehtävänä on suojata laitteita sähköverkkojen häiriöiltä sekä sähkökatkoilta, esimerkiksi palvelin ja RAID-järjestelmä voivat jatkaa toimimiseen myös sähkökatkon aikana. [7.]

2.1.3 Eheys

Tiedon eheydellä tarkoitetaan sitä, että tiedon sisältö ei pystytä muuttamaan tahallisesti tai tahattomasti. Tällöin tieto säilyy luotettavana ja se on saatavilla tarvittaessa. Käyttäjän tietämättömyys ja huolimattomuus hänen käsitellessään tietoa voivat rikkoa eheyden. Esimerkiksi käyttäjä voi vahingossa poistaa tiedon tallennuspaikasta. Eheyteen voi vaikuttaa myös levyrikko, kun kyseessä ovat tekniset laitteet. Tiedon häviäminen, tiedoston eheysvika tai virukset voivat tuhota tiedon tai tehdä siitä käyttökelvotonta. Lisäksi sähköpostin sisällön tai lähettäjätietojen peukalointi kohdistuu tiedon eheyteen. [5, s.22–23.]

Tiedoston sisälle iskevä eheysvika on tietoturvan kannalta kiusallinen ongelma, sillä vika huomataan vasta silloin, kun tiedostoa yritetään käyttää. Vioittunut tiedosto ei käynnisty, vaan jumiuttaa koneen. Tiedon eheyden turvaamiseen liittyy tarkistussummia, lokitiedostoja, tiedonsiirron protokollia sekä erilaisia sisäisiä tarkistuksia ja tarkis-

tusohjelmia, kuten virustentorjuntaohjelmia. [5, s. 23.] Hash-funktiolla eli yhdensuuntaisella salauksella voidaan myös varmistaa tiedoston eheyttä. Sen tuloksena on hyvin pitkä merkkijono, joka on yksilöllinen ja täysin erilainen, jos yksikin merkki alkuperäisestä tiedostosta muutetaan. [3.]

2.1.4 Todentaminen

Luottamuksellisuus vaatii todentamista, jolla saadaan selville käyttäjän aitous riippumatta siitä, onko kyseessä ihminen, sovellus, WWW-sivu tai ohjelmakoodi. Tiedolla ja biteillä ei ole ihmisaistein havaittavaa ominaisuutta. Sen takia keinotekoiset todennusmenetelmät ovat erittäin tärkeitä. Käyttäjien todentaminen tapahtuu yleensä salasanan perusteella. Verkkotiedon ja verkkopalvelun todentaminen on hankalaa. Yleensä todennus tapahtuu verkko-osoitteen perusteella. [5, s. 24-25.]

2.1.5 Pääsynvalvonta

Pääsynvalvonta on tärkein tietoturvakysymys. Sillä huolehditaan siitä, että tietoihin pääsee vain todennettuja käyttäjiä. Pääsynvalvonta suoritetaan usein käyttäjän todentamisella ja tapahtuu tavallisesti kirjauduttaessa työasemaan, ohjelmistojen tai sisäverkkoon. Pääsynvalvontaa sisältyy myös käytön seuranta. Kaikki käyttäjien tekemiset lisätään usein lokitiedostoihin, jolloin myöhemmin huomattua väärinkäyttöä on helppo selvittää tutkimalla lokitiedostoja. [5, s. 27.]

2.1.6 Kiistämättömyys

Kiistämättömyyttä tarvitaan erityisesti esimerkiksi sähköisessä kaupankäynnissä, jossa tilaukset, toimitukset, tilisiirrot ym. muut sopimukset on pystyttävä todistamaan lain edellyttämällä tavalla. Esimerkiksi, jos asiakas ei saa tilaamansa tavaraa, on hänen pystyttävä todistaman tilaus ja maksutapahtuma kiistattomasti. Toisaalta myös kauppiaan on pystyttävä todistamaan se, että hän on lähettänyt kauppatavaran. Kiistämättömyys voidaan saada soveltamalla eheydestä ja todennuksesta. Lisäksi se edellyttää osto ja tilaustapahtumien aikaleimaamista ajankohdan varmistamiseksi. [5, s. 27-28]

2.2 Web-ohjelmointi

Ohjelmointikielellä määritellään ohjeet, kuinka ohjelman tulee toimia. Web-ohjelmoinnissa ohjelma tai sovellus tehdään selainkäyttöiseksi. Kaikki internetissä tapahtuvat asiat vaativat toimiakseen HTML:n (Hypertext Markup Language).

HTML on siis WWW- (World Wide Web) sivujen esittämiseen kehitetty sivunkuvauskieli. HTML ei ole mikään ohjelmointikieli kuten esim. PHP, Java tai C-kieli. HTML-kielellä ei voi siis tehdä tietokoneohjelmia, vaan se on tarkoitettu www-sivujen rakenteen esittämiseen. HTML:ää voidaan kirjoittaa millä tekstieditorilla tahansa, ja Windowsin Notepad onkin aivan riittävä ohjelma.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5  <title>Esimerkki</title>
6  </head>
7
8  <body>
9      Hello world!
10 </body>
11
12 </html>
13
14
```

Kuva 1. WWW-sivun rakenne.

HTML-dokumentti alkaa aina dokumenttityypillä. Kuva 1 on esitetty yksinkertainen internetsivun lähdekoodi, jossa on käytetty dokumenttityyppi XHTML (EXtensible HyperText Markup Language) 1.0. Nykyään HTML:n paranneltu versio XHTML on yleistymässä, koska se sopii HTML:ää paremmin myös muihin medioihin kuin perinteisiin tietokoneisiin, esimerkiksi mobiililaitteisiin. Lisäksi XML-pohjaisuus tekee XHTML:stä käyttökelpoisemman, koska se estää sivun rakenteeseen syntyviä virheitä ja tekee XHTML-dokumentista helposti validoitavan. Validoimalla varmistetaan, että dokumentti on oikein muodostettu. HTML:ssä tyhjän elementin lopputägin voi jättää kirjoittamatta, mutta XML-pohjaisuuden takia XHTML:ssä lopputägi on pakollinen, ja se pitää olla oikeassa paikassa. W3C:n (World Wide Web Consortium) validaattorilla voidaan tarkastaa sivun rakenteen oikeellisuus, esimerkiksi www-sivuja URL- (Uniform Resource

Locator) osoitteesta tai omalta koneelta kokonaisen HTML-dokumentin. [8.] Jos validointi ei raportoinut virheitä, saa läpäisy sivulla käyttää merkkinä Kuva 2 näkyvää kuvaketta.

Kuva 2. W3C:n XHTML 1.0- validoinnin läpäisyn ilmaiseva kuva.

XHTML:n rinnalla HTML5 on voimakkaasti yleistymässä ja herättää suurta mielenkiinntoa. Siinä käytetään HTML5:n lisäksi webistä tavanomaisia teknikoita kuten CSS:ää (Cascading Style Sheets*)* ja JavaScriptiä. [9.] Se toimii myös sellaisilla internet-sivuilla, joissa ei ole verkkoyhteyttä.

HTML5 tarjoaa mahdollisuuden digitaaliseen palveluun. Tämä tarkoittaa sitä, että sovellukset saadaan toimimaan kaikenlaisissa laitteissa, niin työpöydällä kuin mobiilissakin, ilman työlästä sovittamista tai uudelleenkoodaamista. Musiikin tai videon toistamisessa ei käytetä enää erillisiä selainlaajennuksia eli plugineja. HTML5-sovellukset ovat erityisesti sopivia sellaiselle myytäville tai jaettaville ohjelmille, joiden tarkoitus on käyttää eri laitteita. [9.]

2.3 PHP

PHP:n ensimmäinen versio julkaistiin vuonna 1995 ja tällä hetkellä kaikkein uusin versio on PHP 5.4.8, joka julkaistiin lokakuussa 2012. Versio 5.4 tuo paljon uusia toimintoja ja parantaa käyttönopeutta. Esimerkiksi PHP:stä löytyy nyt itsestään web-palvelin.

PHP:n ensimmäinen versio julkaistiin vuonna 1995 ja tällä hetkellä kaikkein uusin versio on PHP 5.4.8, joka julkaistiin lokakuussa 2012. Versio 5.4 tuo paljon uusia toimintoja ja parantaa käyttönopeutta. Esimerkiksi PHP:stä löytyy nyt itsestään web-palvelin. Ennen sitä pitäisi asentaa rinnalle jokin toinen web-palvelin, kuten Apache. [10.]

PHP on komentosarjakieli, joka soveltuu erityisesti web-sovelluskehitykseen. Kieli on tarkoitettu dynaamisten web-sivustojen suorittamiseen eli sivustojen toiminnallisuuden luomiseen. [11.] Se muistuttaa hyvin paljon Perliä (Practical Extraction and Report Language), C:tä ja Javaa, mutta sillä on myös paljon aivan omia, uniikkeja, piirteitä. PHP toimii niin sanotusti upotetussa tilassa, eli sitä voidaan kirjoittaa suoraan kaikkien HTML-sivujen lähdekoodiin. [12.] PHP-koodi kirjoitetaan merkintöjen ”<?php” ja ”?>” väliin.

PHP on tulkattava kieli, eli WWW-sivun sisällä oleva PHP-koodi ajetaan joka kerta, kun WWW-palvelin lähettää sivun selaimelle. PHP-sivujen tuottamiseen tarvitaan palvelin, joka tukee PHP:tä. PHP-koodi suoritetaan aina palvelimella ennen kuin sivu lähetetään selaimeen. [13, s. 13.] Tämä tarkoittaa sitä, että käyttäjä ei näe sivua selatessaan koodia, vaan pelkästään suoritetun toiminnon tai lauseen.

Esimerkkikoodi 1 nähdään, miten PHP-koodi sijotetaan HTML-muotoilujen sekaan.

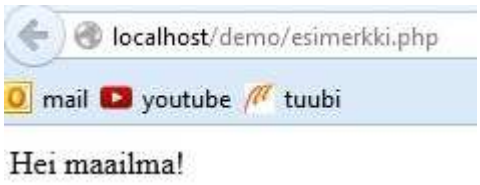
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; char-
set=utf-8" />

<head><title>Yksinkertainen PHP-esimerkki</title></head>
<body>
<?php
echo "Hei maailma!";
?>
</body>

</html>
```

Esimerkkikoodi 1. PHP yksinkertaisimmillaan.

Kun yllä oleva koodi tallennetaan nimellä esimerkki.php ja sitä kutsutaan PHP-sallitulta Web-palvelimelta, tuloksena on Kuva 3 nähtävä tuloste.



Kuva 3. Esimerkki PHP-tulosteesta.

PHP-koodi on periaatteessa aina sarja lauseita. Lause on käytännössä ohjelmassa itsenäisesti toimiva elementti. Esimerkkikoodi 1 oleva echo-komento lopetetaan puolipisteeseen, mikä kertoo PHP:lle lauseen päättymisestä. Puolipisteen unohtuminen on yksi usein tapahtuvista ohjelmointivirheistä. [13, s. 65.]

2.4 MySQL

MySQL on maailman suosituin avoimen lähdekoodin relaatiotietokantaohjelmisto, jonka kehittäjä on MySQL AB. MySQL:n ensimmäinen versio julkaistiin 1996, ja kaikkein uusin versio on MySQL 5.6 . [14, 15.]

MySQL on saatavilla monelle eri käyttöjärjestelmälle, ja se on tunnettu toimivuudestaan, nopeudestaan ja siirrettävyydestään. Tuetut käyttöjärjestelmät ovat kaupalliset ja vapaat Unixit, eri Windows-versiot ja Linuxit. MySQL on yleisimmin web-palvelimilla saatavilla oleva tietokantaohjelmisto. Monet maailman suurimmista sovelluksista, kuten Facebook, Google, Adobe luottavat MySQL:iä, joka säästää aikaa ja rahaa. [16.]

MySQL:n omistaa nykyisin maailman suurimpiin tietokantaratkaisujen toimittajana tunnettu Oracle, joka osti vuonna 2009 Sun Microsystemsin, joka vuorostaan oli ostanut MySQL AB:n vuotta aiemmin. [17.]

Toisaalta MySQL:n kehittäjän Michael ”Monty” Wideniuksen mielestään Oracle ei ole tehnyt mitään tärkeää MySQL:n kehittämistä varten. Sen lisäksi hän kertoo olevansa tietokannassa useita virheitä. Hän väittää tähän liittyen, ettei ohjelmistojätti halua MySQL:n kasvavan suureksi ja käytännössä kilpailevan itse kehittänsä Oracle-tietokannan kanssa. [18.]

Widenius on kehittänyt perustamaansa avoimen lähdekoodin MariaDB-tietokantaa, joka nimittäin sisältää MySQL:n osia, joiden tekijänoikeus on Sunilla. MariaDB:n tietokantaan on siirtynyt Elisa ja internetin vapaasti käytettävä tietosanakirja Wikipedia vaihtaa MySQL:sta MariaDB-tietokantaan. [19.]

Wikipedian sivuston arkkitehti Asher Feldmanin mielestään siirtopäätöksenä ovat tekniset ominaisuudet. Hänen tilastojensa mukaan tietojen käsittelynopeus on kasvanut 2–10 prosenttia, kun mitataan kyselyjen määrää sekunnissa. Keskimäärin suorituskyky nousi 8 prosenttia.

MySQL:n jälkeen toiseksi suosituin avoin tietokanta on PostgreSQL, joka on avoimena lähdekoodina jaettava tietokannan hallintajärjestelmä ja sitä on kehitetty jo yli 15 vuotta. Sen parhaita puolia on tyyppien määritteleminen: se tarjoaa mahdollisuuden rakentaa uusia monimutkaisia tietotyyppejä. PostgreSQL on hieman MySQL:ää hitaampi, mutta ominaisuuksiltaan parempi ja luotettavampi tietokanta.

PHP:tä ja MySQL-tietokantaa käytettäessä on olemassa kaksi vaihtoehtoa tietokantayhteyden muodostamiseen ja kyselyiden suorittamiseen: PDO (PHP Data Objects) ja MySQL. Nykyaikainen tapa käyttää MySQL-tietokantaa PHP:llä on PDO-rajapinta, joka on käytössä PHP-versiosta 5.1 lähtien. PDO:lla voi käyttää MySQL:n lisäksi muitakin tietokantaohjelmia samalla tavalla. [20.]

Kuva 4 on esitetty tietokantayhteyden muodostamista PDO:lla.

```php
<?php

//yhteyden muodostus tietokantaan
try {
    $yhteys = new PDO("mysql:host=localhost;dbname=testi", "kayttaja", "salasana");
} catch (PDOException $e) {
    die("VIRHE: " . $e->getMessage());
}

// virheenkäsittely: virheet aiheuttavat poikkeuksen
$yhteys->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
// merkistö: käytetään latin1-merkistöä; toinen yleinen vaihtoehto on utf8.
$yhteys->exec("SET NAMES latin1");

?>
```

Kuva 4. Tietokantayhteyden muodostaminen.

PDO on tietoturvallisempi tapa verrattuna mysql_ -funktiot, eikä tietoja tarvitse tarkistaa haitallisilta merkeiltä, koska PDO tekee käyttäjän syöttämän tiedon vaarattomaksi lisäämällä tarvittavat erikoismerkit. PDO hyödyntää ”prepared statements” tekniikkaan ja se on erinomainen metodi tietoturvan kannalta. Sillä tekniikalla toteutettu kyselyt ovat valmiiksi määriteltyjä kyselyitä, joissa kyselyissä käytettävät arvot annetaan parametreina. Valmiita kyselyitä käyttämällä voidaan estää automaattisesti SQL-injektiot, sillä parametreja käytettäessä varmistetaan että parametrien arvot liittyvät aina vain tiettyyn kenttään. [21.] Tähän asiaan palataan myöhemmissä luvuissa tarkemmin.

MySQL-tietokanta on hyvin suosittu ja tehokas tietokanta web-palveluiden käytössä. MySQL-tietokannan päälle rakennettava ohjelmalogiikka luodaan usein PHP-, Python- tai Perl-ohjelmointikielellä, sivut julkaistaan Apache-web palvelimella, joka lisäksi toimii Linux-käyttöjärjestelmän päällä. Tätä sanotaan joskus LAMP (Linux, Apache, MySQL, PHP)- alustaksi. [14.]

3 Web-palveluihin kohdistuvat uhat

Uhat, jotka kohdistuvat Web-palveluihin, ovat saaneet uusia ilmenemistyyppejä internetin kehittyessä. Ennen hyökkäykset kohdistuivat palveluita tarjoaviin alustoihin kuten Apacheen ja Microsoft IIS:iin (Internet Information Server). Nämä hyökkäykset käyttävät hyväkseen tunnettuja haavoittuvaisuuksia, ja asianmukaisilla työkaluilla varustautunut hyökkääjä onnistui kaatamaan haavoittuvan palvelun muutamassa minuutissa. [22.]

PHP sekä muut ohjelmointikielet voivat olla turvallisia, mutta se ei tarkoita, että ne ovat suojassa hyökkäyksiltä. Web-sovelluksissa on paljon huolestuttavia tietoturva-aukkoja. Syynä tähän ovat yleensä kehittäjien huolimattomuus syötteiden tarkistuksessa ja epäturvallisen koodin käyttö. [23.]

Nykyään havaittuihin haavoittuvaisuuksiin reagoidaan aiempaa nopeammin. Yhä enemmän alusta on määritelty oikein ja on tarjolla välineitä, joiden avulla voidaan havaita yleisimmät tietoturvariskit nopeasti ja helposti. [24.]

3.1 OWASP

Tässä kappaleessa käydään läpi yleisimpiä verkkouhkia käyttäen apuna OWASP-(Open Web Application Security Project) tietoturvaorganisaation Top 10 -tietoturvariskien tuloksia.

OWASP on maailmanlaajuinen voittoa tavoittelematon vapaaehtoisjärjestö, joka on keskittynyt parantamaan verkkosovelluksien tietoturvallisuutta. Yhteisön päämääränä on kasvattaa tietämystä turvallisten sovellusten kehitystä ja ylläpitoa. [24.]

OWASP Top 10 on lista, jolle on kerätty kymmenen kriittisintä verkkosovellusten tietoturvariskiä. OWASP on julkaissut Top 10 -listan ensimmäisen kerran vuonna 2003, ja se on päivitetty vuosina 2004, 2007 ja 2010. Viimeinen versio uudistettiin riskin mukaisesti vuonna 2013. OWASP Top 10:n merkittävin tarkoitus on kouluttaa sovelluskehittäjiä, suunnittelijoita, arkkitehtejä ja organisaatioita uhkaavien sovellushaavoittuvuuksien tuloksia. [24.]

Taulukko 1. OWAPS:n TOP 10 -tietoturvariskit vuosina 2007, 2010 sekä 2013. [24.]

OWASP Top 10 - 2007	OWASP Top 10 - 2010	OWASP Top 10 - 2013
A1 - Cross Site Scripting (XSS)	A1-Injection	A1-Injection
A2 - Injection Flaws	A2-Cross Site Scripting (XSS)	A2-Broken Authentication and Session Management
A3 - Malicious File Execution	A3-Broken Authentication and Session Management	A3-Cross Site Scripting (XSS)
A4 - Insecure Direct Object Reference	A4-Insecure Direct Object Reference	A4-Insecure Direct Object Reference
A5 - Cross Site Request Forgery (CSRF)	A5-Cross Site Request Forgery (CSRF)	A5-Security Misconfiguration
A6 - Information Leakage and Improper Error Handling	A6-Security Misconfiguration	A6-Sensitive Data Exposure
A7 - Broken Authentication and Session Management	A7-Insecure Cryptographic Storage	A7-Missing Function Level Access Control
A8 - Insecure Cryptographic Storage	A8-Failure to Restrict URL Access	A8-Cross Site Request Forgery (CSRF)
A9 - Insecure Communications	A9-Insecure Cryptographic Storage	A9-Using Components with Known Vulnerabilities
A10 - Failure to Restrict URL Access	A10-Unvalidated Redirects and Forwards	A10-Unvalidated Redirects and Forwads

Taulukossa 1 nähdään 2007, 2010 sekä 2013 vuosien OWASP:n Top 10 -listan. Vuosina 2010 sekä 2013 on tapahtunut muutoksia ja listaan on tullut uusia riskejä. Vuoden 2010 listan kolmen kärkeen kuuluvat riskit näkyvät myös vuoden 2013 listan kolmen kärkeen XSS-hyökkäyksen sijoitus on pudonnut yhden sijan alaspäin.

3.2 Yleisimmät web-sovelluksia vastaan kohdistuvat hyökkäykset

Tässä kappaleessa tutustutaan tarkemmin niihin injektioihin, jotka kuuluvat vuoden 2010 Top 10 -listan kolmen kärkeen.

3.2.1 SQL-injektio

SQL-injektiolla tarkoitetaan tavanomaista hyökkäystä web-sovelluksia vastaan, missä hyökkääjä pyrkii syöttämään muuttujien avulla SQL-komentoja, joita ei pitäisi pystyä antamaan. [25.] Näissä tilanteissa hyökkääjä muokkaa hakemiseen tarkoitettua SQL-lausetta niin, että hän saa esiin listan käyttäjätunnuksia ja salasanoja. [26.]

SQL-injektio tapahtuu silloin, kun verkkosovelluksessa ei tarpeeksi tarkisteta ja puhdisteta käyttäjältä tulevaa syötettä, vaan ne päätyvät sellaisenaan tietokantakyselyyn. [27.] Kaiken käyttäjältä tulevan tiedon oikeellisuus pitää tarkistaa ja tietotyyppien kohdalla tulee tarkistaa, että tieto on oikeassa muodossa. [25.]

Injektiohyökkäyksessä on olemassa kolme erilaista vaihetta. Hyökkääjä määrittää ensimmäisessä vaiheessa Web-sovelluksessa käytetyt teknologiat. Tämä saadaan selville esimerkiksi kuvaamalla sivujen antamia virheilmoituksia ja analysoimalla sivun lähdekoodia käyttäen ohjelmia kuten Nessua ja Nmapia. Ammattitaitoiselle hyökkääjälle tämä vaihe on aika yksinkertainen, mutta toisaalta se on välttämätöntä injektiohyökkäyksen onnistumisen kannalta, koska se riippuu käytetystä alustasta. Toisessa vaiheessa suunnitellaan pääasiallista hyökkäystä. Tämä tehdään määrittämällä käyttäjän sovellukselle antamia syötteitä. Tämä sisältää käytännössä kaiken sen tiedon, joka saadaan siirtää HTTP:n (Hyperttext Transfer Protocol) GET- ja POST-pyynnöissä. Viimeisessä vaiheessa hyökkääjän tehtävänä on tunnistaa niitä syötteitä, joiden avulla muokataan sovelluksen toimintaa. Tähän tehtävään löytyy myös monia valmiita ohjelmia. [26.]

Oletetaan, että verkkosivuilla kirjautumiskohta on se, mihin käyttäjä syöttää käyttäjätunnuksen ja salasanan. Kirjautumislomakkeessa SQL-lause voisi olla esimerkiksi seuraavanlainen:

”SELECT * FROM users WHERE username = 'admin' AND password = 'salasana'”

Hyökkääjä pyrkii muokkaamaan lausetta siten, että se näyttää esimerkiksi seuraavalla tavalla:

”SELECT * FROM users WHERE username = '' OR 1=1- - AND password = 'salasana'”

Hyökkääjän kirjoittaessa username-kenttään komennon ‟OR 1=1--‟ haetaan users-taulukosta käyttäjänimeä, joka täyttää ehdon ”1 = 1”, koska tämä ehto on tosi. Hyökkääjä ei tarvitse salasanaa tutkiakseen tietokantaa, koska vaatimukset salasanasta poistuvat ”- -”-merkin jälkeinen salasanakysely jää pelkkänä kommentiksi eli sitä ei huomioida! [28.]

3.2.2 Istunnon kaappaus

Istunnolla (session) tarkoitetaan käyttäjän työpöytää ja tietyn ajan auki olevia ohjelmia. Käyttäjän kirjautuessa sisään istunto käynnistyy ja uloskirjatuessa päättyy. Istuntomuuttujat ovat vain yhden käyttäjän muuttujia, joita toiset eivät pääse näkemään. Istunnon tunnistaminen onnistuu evästeiden avulla, jotka ovat käytössä vain istunnon ajan eivätkä tallennu tietokoneelle.

PHP-skriptissä istunnon aloittaessa lähetetään käyttäjän selaimelle eväste, jonka nimi on PHPSESSID eli ”PHP session id”. [13, s. 273.] Hyökkäyksiä, jotka kohdistuvat istuntoihin, on kolmenlaisia. Ne ovat istunnon kaappaus (capture), arvaaminen (prediction) ja muuttaminen etukäteen (fixation). [29.] Istunnon kaappauksella tarkoitetaan, että hyökkääjä varastaa käyttäjän istuntotunnuksen.

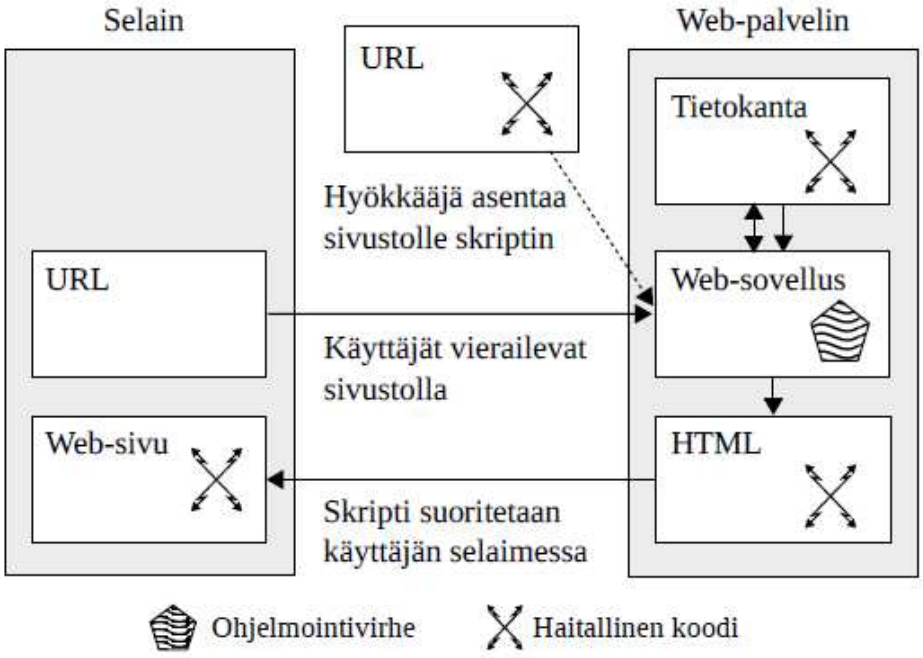
Sessiotunnisteen kuljettaminen tapahtuu PHP:ssä kahdella tavalla: URL:n mukana tai evästeessä. PHP kuljettaa itse niitä automaattisesti, joten koodissa sitä ei oteta huomioon. Se on toisaalta huomattava PHP:n asetuksia määritellessä. Sessiotunnisteen välittäminen evästeessä on tietoturvallisempi kuin välittäminen URL:n mukana. [30.]

Istunnon id-tunnisteen arvaaminen on olennaisesti toteuttamiskelvoton. Esimerkiksi PHP:n istuntomekanismilla muodostettu PHPSESSID niminen eväste, jonka arvo on äärimmäisen satunnainen, on 32 merkkiä pitkä teksti. Lukuisen vaihtoehtojen takia id:n selvittämiseen kuluisi paljon aikaa. [29.]

Istunnon tunnisteen kaappaukselle on olemassa erilaisia keinoja. Tietovirtoja voidaan salakuunnella tai uhrin koneelle voidaan tartuttaa troijalaisia XSS-hyökkäyksen avulla. Sessiontunnistetta kuljetettaessa URL:n mukana uhri saattaa itse epähuomiossa ilmaista oman tunnisteensa. Tämä saadaan onnistumaan esimerkiksi siten, että käyttäjä ohjataan toiselle sivulle huijaamalla esimerkiksi hyökkääjä välityspalvelimelle. [29, 31.]

3.2.3 Cross Site Scripting

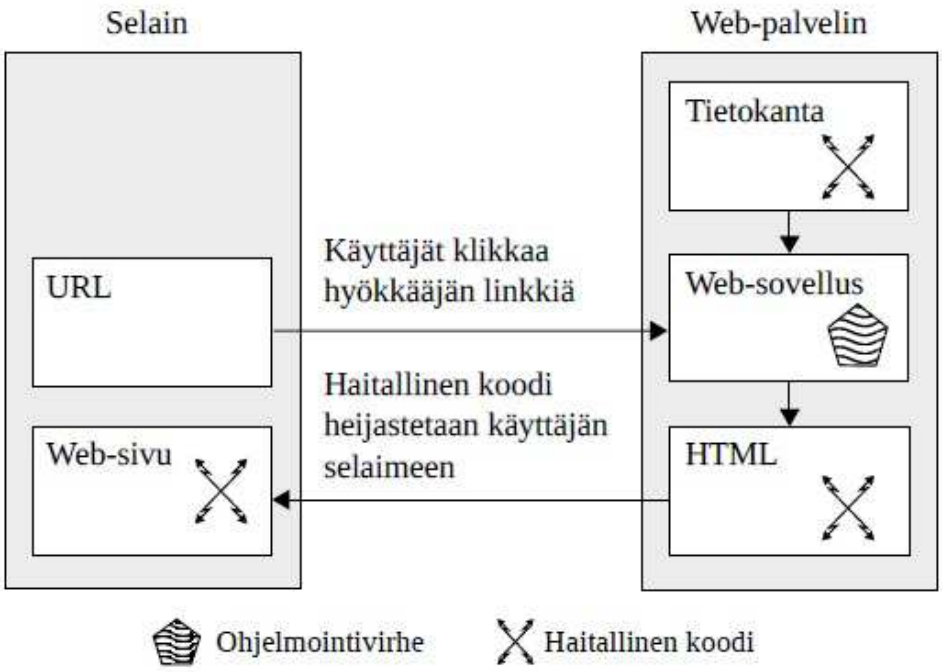
Cross Site Scripting (XSS) -hyökkäykset ovat yleisiä injektiohyökkäyksiä, joita toteutetaan kirjoittamalla jotain verkkoselaimessa toteutettavaa koodia (käytännössä javascriptiä). [32.] XSS-hyökkäyksen onnistumiseen tarvitaan kolme osapuolta; hyökkääjä, haavoittuva Web-palvelu ja myös uhri. Kolmannen osapuolen lisääntyessä XSS-hyökkäyksen suorittaminen vaikeutuu. Hyökkäyksen onnistuessa hyökkääjä pystyy kirjoittamaan haluamaansa koodia käyttäjän selaimeen XSS-haavoittuvan Web-sivun kautta. [25.] Erilaiset JavaScript-pohjaiset komponentit kuten RSS-syötteet, widgetit ja mashupit antavat useita tilauisuuksia hyökkääjille iskemistä varten. [22.]



Kuva 5. Palvelimelle tallennettu XSS-hyökkäys. [18.]

Hyökkäyksen toteutuksen mukaan XSS-hyökkäykset ryhmitellään tallennettuihin ja heijastettuihin hyökkäyksiin. Tallennetussa hyökkäyksessä sivun vierailun yhteydessä suoritetaan vaurioituneelle Web-sivustolle asennettu haitallinen skripti (Kuva 5). Tämän kaltaiset skriptit on yleensä toteutettu JavaScriptillä. Ne saatetaan tallentaa esimerkiksi Web-palvelimen tietokantaan, kommenttikenttiin tai foorumin viesteihin. Hyökkääjä onnistuu lukemaan ja muokkaamaan selaimen henkilökohtaista sisältöä, koska selain uskoo siihen, että sivuston sisältö tulee luotettavasta lähteestä. Hyökkääjä yrittää ensin kaapata Web-sovellusten toiminnan kannalta merkittäviä evästeitä, koska niiden avulla ylläpidetään yhteyksien tiloja eri sivustoille oikean käyttäjän tunnistamiseen tarpeellisia tietoja. Näitä sieppaamalla hyökkääjä voi toimia verkossa toisena käyttäjänä ja käyttää oman tarkoituksensa mukaan, kuten verkkopankissa tai webmailissa. [22.]

Heijastetussa hyökkäyksessä vahingollista skriptiä ei tallenneta pysyvästi WWW-sovellukseen (Kuva 6), toisin kuin tallennetusta hyökkäyksestä.



Kuva 6. Palvelimelta heijastettu XSS-hyökkäys. [18.]

Hyökkäys tapahtuu silloin, kun käyttäjä painaa haitallista koodia upotetulle linkille, jonka href-attribuuttina voi olla seuraava:

”http://site.com/?name=<script>alert(’XSS’)</script>”

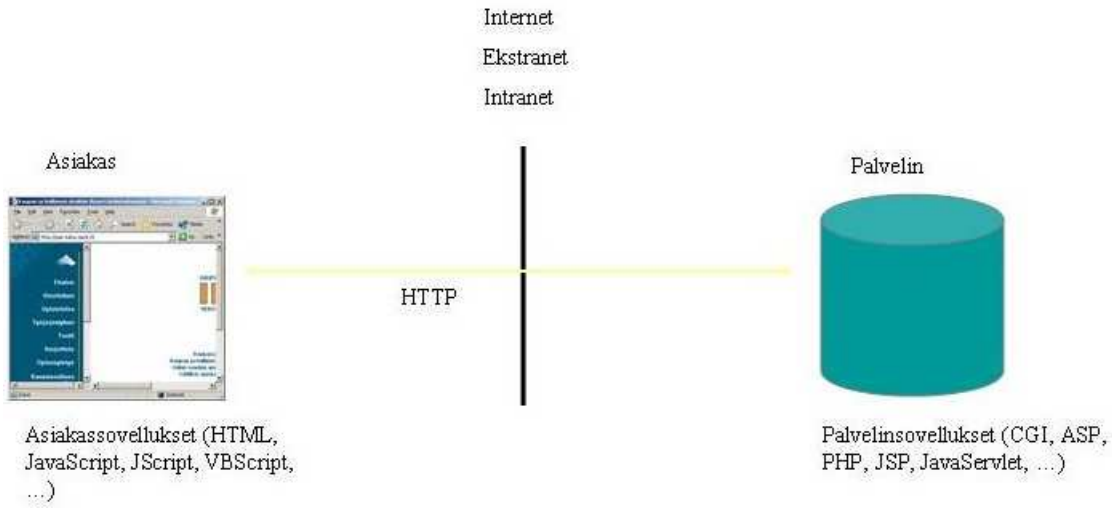
Linkki suorittaa käyttäjän sijasta kyselyn haavoittuvalle sivustolle. Haitallinen koodi suoritetaan käyttäjän selaimessa, kun vaurioituva sivusto näyttää kyselyn tulokset käyttäjälle, koska selain luulee sen tulevan luotettavasta lähteestä. Hyökkääjällä on mahdollisuus varastaa evästeitä sekä kaapata istuntoja ja käyttäjätilejä. [22.]

4 Web-sovelluksen tietoturvallinen toteutus

Tässä luvussa käydään läpi PHP:n ja MySQL:n perusteita tietoturvallisuuden näkökulmasta.

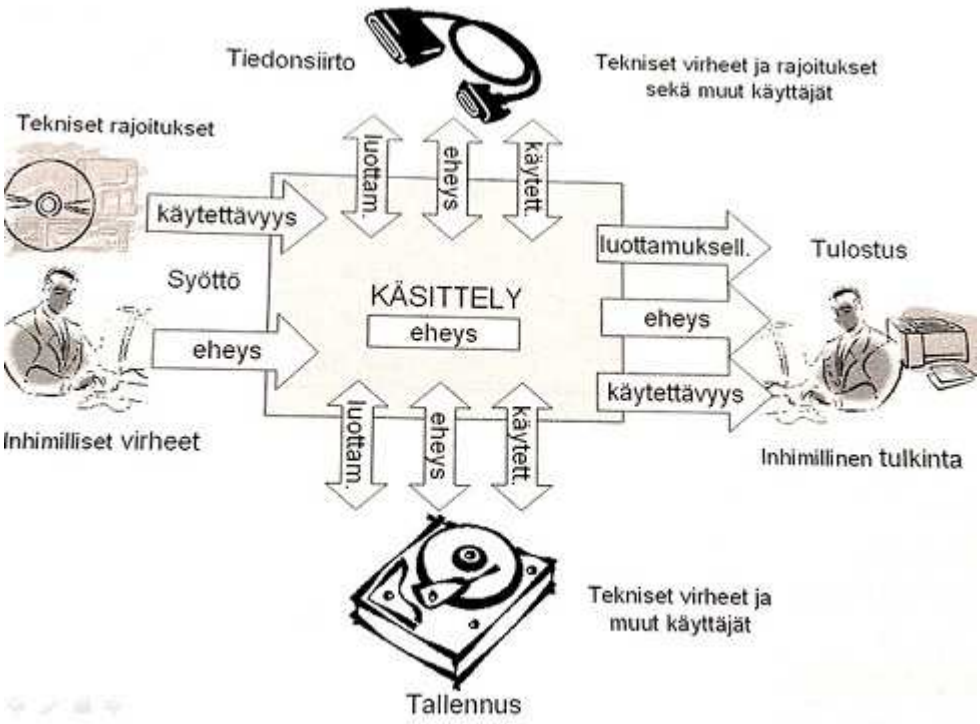
Tietoturvallisuus on merkittävä tekijä web-sovelluksia suunniteltaessa, asennettaessa, kehitettäessä ja laajennettaessa. Sovelluksien suunnitteluvaiheessa tulee muistaa, että turvallisuus on jatkuva prosessi, ei ominaisuus.

Web-sovellukset jaetaan asiakas- ja palvelinsovelluksiin (Kuva 7). Ne eroavat toisistaan siten, missä web-sovelluksen toiminta suoritetaan. Asiakassovelluksessa toiminnallisuus suoritetaan käyttäjän selaimessa. Toteutustekniikkana voi olla jokin skriptikieli, esimerkiksi JavaScript. Toisaalta palvelintekniikassa web-sovelluksen toiminta suoritetaan web-palvelimella. Palvelinsovelluksen toteutustekniikoita voi olla useita, joista ehkä tärkeimmät ja käytetyimmät tällä hetkellä ovat CGI (Common Gateway Interface), PHP, ASP (Active Server Pages), JSP (JavaServer Pages) ja JavaServletit. [33.]



Kuva 7. Web-sovelluksen toteutus. [33.]

Ohjelmoidessa web-sovelluksia pitää ottaa huomioon, mistä tieto tulee ja minne se menee. Informaatio voi kulkea sovelluksesta ulos (esimerkiksi käyttäjän selaimelle) tai sovellukseen sisään (esimerkiksi lomakkeelle syötetyt tiedot). Sen lisäksi sovellussuunnittelijan ja ohjelmoijan pitää ottaa myös huomioon luvussa 2 esitetyt tietoturvallisuuden perustekijät.



Kuva 8. Ohjelmistosuunnittelussa huomioitavat riskitekijät. [34, s. 319]

Kuva 8 nähdään, että sovellusta kehittäessä on otettava huomioon todellisten ohjelmavirheiden lisäksi informaation syöttäessä muodostuneet inhimilliset virheet, virheelliset tulosteiden tulkinnat sekä tietojenkäsittely- ja tiedonsiirtolaitteista johtuneet tekniset virheet. Toisaalta informaatiota tulostettaessa, siirrettäessä ja tallennettaessa pitää huomioita luottamuksellisuuden säilyttämiseksi. [34, s. 319.]

4.1 PHP:n sovellustason tietoturva

Tässä luvussa tutkitaan PHP-ohjelmoinnin tietoturvaa ja selvitetään, miten voidaan ennaltaehkäistä luvussa 3 esitettyjä hyökkäyksiä.

Yleensä tietoturvahaavoittuvuudet johtuvat kokemattomasta ohjelmoijasta, joka jättää vaarallisia aukkoja PHP-skripteihin. Tietoturvallisessa ohjelmoinnissa tärkeintä on se, että tietää, mitä on tekemässä. Web-sovellusten yleisestä tietoturvariskistä lukuun ottamatta PHP-kielen toiminnot edellyttää enemmän huomautuksia tietoturvan huomioista.

Suurin osa PHP-sovelluksiin kohdistuvista hyökkäyksistä johtuu käyttäjien syötteistä. PHP-kielessä ohjelmoijan ei käytännössä täydy hallita ohjelman muistia itse. PHP:ssä on kuitenkin sellaisia funktioita, jotka sallivat puskurin ylivuodon. PHP-ohjelmoijan pitää tietää hyvin haavoittuvuuksista, erityisesti käyttäessään vanhaa PHP-versiota. Syynä on se, että myös hyökkääjät tutkivat haavoittuvuuksien listaa. [35.]

Tässä tutkitaan PHP-ohjelmoinnin tietoturvaa ja selvitetään miten voidaan ennaltaehkäistä luvussa 3 esitettyjä hyökkäyksiä. Selvittämisessä käytetään esimerkiksi apuna rekisteröinti- ja sisäänkirjautumissivuja.

4.1.1 Lomaketietojen välitys

Lomakkeet käyttävät kahta tapaa välittää tietoja: GETiä ja POSTia. GET-metodi on oletusarvo. Silti POST-metodia käytetään yleisesti, koska se pystyy välittämään suurempia tietomääriä palvelimelle. [36, s. 260.] Sitä ei kuitenkaan voi käyttää muutoin kuin form-tagin sisällä olevien HTML-elementtien sisältämien tietojen lähettämiseen. Näitä elementtejä voivat olla esimerkiksi tekstikentät, valintaruudut, valintanapit sekä painikkeet.

Käytettäessä POST-metodia kaikkeen PHP-skriptille välitettyyn tietoon pitää osoittaa käyttämällä $_POST-syntaksia. [36, s. 260.]

GET-metodissa lomakkeen tiedot välitetään käsittelysivun osoitteen mukana. Näin ollen menetelmä ei sovi henkilökohtaisten tietojen välittämiseen kuten käyttäjätunnuksien ja salasanojen välittämiseen. GET-metodia käytetään yleensä niissä tilanteissa, joissa tietoja ei muuteta tietokannassa. Esimerkiksi sitä käytetään yleensä resurssien haussa. [13, s. 223-224.]

Toisaalta POST-metodissa lomakkeen tiedot välitetään sivupyynnön mukana käyttäjältä näkymättömissä, joten se soveltuu paremmin henkilökohtaisten datan lähettämiseen. Kuitenkin huolehdittava siitä, että verkkoa ei päästä salakuuntelemaan eli annetaanko tiedot suojatun yhteyden kautta.

4.1.2 PHP:n konfiguraatio

PHP:n tarjoama konfiguuratioparametrien päämääränä on nostaa PHP:n turvallisuustietoisuuden tasoa. [36, s. 476.] Tässä käydään läpi pääasiallisia konfiguraatiodirektiivejä.

Ensin käsitellään direktiivi error_reporting, jolla tutkitaan PHP:n virheiden raportointitoiminta. Se kertoo raportoinnin herkkyystason, jolle on olemassa 12 erilaista tasoa. Esimerkiksi E_ALL-taso näyttää kaikki virheet ja varoitukset. [36, s. 28-29.] Virheet pitäisi kirjata lokiin. Sen kohde määritellään php.ini-tiedoston error_log-direktiivillä. [36, s. 176.] Tällöin virheilmoitukset eivät näy webissä. Voidaan myös muokata php.ini-tiedostoa niin, että virheet menevät haluamaasi tiedostoon:

”error_log = /home/matt/logs/php_errors”

Toisaalta display_errors-direktiivi tulee käyttää vain testausvaiheessa, koska hyökkääjä voi käyttää sitä omaksi hyödykseen. Tämä direktiivi voidaan saada pois käytöstä muokkaamalla php.ini-tiedostosta ”display_errors = Off”. [36, s. 175.]

Yksi tärkeimmistä direktiiveistä on register_globals, joka on php.ini-tiedoston määritelmä. Ennen versiota 4.1 register_globals oli päällä eli ”register_globals=on”. Käytännössä tämä tarkoittaa sitä, että ne olivat myös muokattavissa globaalisesti. Tämä antaa hyökkääjille mahdollisuuden päästä hyökkäämään ohjelmaan. Tämä direktiivi on estetty oletuksena lähtien versiosta 4.2.0, koska kaikki ohjelmaan saapuneet muuttujat, jotka ovat tyyppiä COOKIE, ENVIRONMENT, GET, POST ja SERVER, olivat automaattisesti globaaleja. [36, s. 31-32.]

Php.inissä on myös safe_mode-direktiivi, joka rajaa tiedostojen käsittelyoikeuksia. Esimerkiksi kaikkien syöttö- ja tulostusfunktioiden käyttö on sallittua vain niihin tiedostoihin, joiden omistaja on sama kuin skriptillä. Verrataan, että UID- tunnus (käyttäjätunnus) ja skriptin UID:n tunnukset ovat samat. Näin ollen skripti voidaan suorittaa. Muutoin skripti epäonnistuu. [36, s. 476.]

Disable_functions-direktiivillä voidaan estää vaarallisten funktioiden suoritus. Funktioita voidaan laittaa samaan listaan, joissa ne erotetaan pilkulla esimerkiksi seuraavalla tavalla: disable_functions = fopen, popen, file. [36, s. 478.]

4.1.3 Syötteiden tarkistaminen

Huolimattomasta syötteiden tarkistamisesta johtuvien haavoittuvuuksien määrä on suuri. Käyttäjän antama tieto voi olla mitä tahansa. Tästä johtuen käyttäjän antamiin syötteisiin ei pidä koskaan luottaa, eikä niitä voi jättää tarkistamatta. Käyttöjen syöttötiedot on ehdottomasti käsiteltävä varsinkin, jos siitä muodostetaan suoritettava koodia. Syötteen haavoittuvaisuuksiin kohdistuvia uhkia ovat olennaisesti XSS- ja SQL-injektiohyökkäykset.

Syöttötiedon oikeellisuudessa tarkistetaan yleensä syötteen pituutta ja muotoa. PHP:ssä voidaan selvittää syötteen pituutta strlen-funktiolla ja syötteen olemassaoloa voidaan tarkistaa käyttäen isset- tai empty-funktioita. Funktiolla preg_match voidaan etsiä jostakin merkkijonosta täsmäävyyttä. Tarkastuksen jälkeen puhtaat arvot sijoitetaan omaan taulukkoonsa. POST-taulukkoa ei ole mielekästä käyttää, vaan luodaan oma taulukko, johon sijoitetaan kaikki lomaketiedot POST-taulusta. Näin ei tarvita koko ajan miettiä, onko tieto puhdasta.

```
$clean = array();

if(empty($_POST['username'])){
     die("Käyttäjätunnus ei saa olla tyhjä.");

}else if( strlen($_POST["username"]) < 4 ){
     die("Käytäjätunnuksessa pitää olla vähintään 4 merkkiä ");

}else if(!preg_match('/^[ a-zA-Z0-9]+[.-_]*[a-zA-Z0-
9]$/',$_POST['username'])){
     die("Käyttäjätunnuksen alussa ja lopussa on sallittu vain
       kirjaimia ja numeroita, toisaalta se voi sisältää
       väliviiva,alaviiva sekä piste.");

}else {
     $clean['username'] = strip_tags($_POST["username"]);
}
```

Esimerkkikoodi 2. Syötteen tarkistaminen

Esimerkkikoodissa 2 nähdään, miten tarkistetaan syötteen olemassaoloa, pituutta ja täsmäävyyttä. Tarkistamisen jälkeen lomaketieto (username) sijoitetaan $clean-nimiseen taulukkoon ja strip_tags()-funktiolla poistetaan HTML- ja PHP-tagit.

PHP-versiosta 5.2.0 lähtien on käytössä filter_var-funktio, jolla voidaan validoida esimerkiksi sähköpostiosoitteen tai url-osoitteen oikeellisuutta. [25.] Funktion avulla hoidetaan asia helposti. Enää ei tarvitse kirjoittaa säännöllisiä lausekkeita.

Seuraavassa Esimerkkikoodi 3 nähdään, miten sähköpostiosoite validoidaan käyttäen filter_validate-funktiota.

```
if(filter_var($_POST['email'], FILTER_VALIDATE_EMAIL))
{
   $clean['email'] = $_POST['email'];
}else{
   die("Virheellinen sähköpostiosoite");
}
```

Esimerkkikoodi 3. Sähköpostiosoitteen tarkistaminen.

SQL-injektiolta suojautuminen

SQL-injektion ehkäisemiseksi on tärkeää käyttää sopivia rajapintoja, jolloin hyökkäyksiltä ilmenevä virhe on mahdoton. [27.] Ohjelmoijat voivat suojautua hyökkäyksiltä käsittelemällä tarkasti sisään tulevaa tietoa, esimerkiksi varmistamalla tiedontyyppi ja syöttötiedon pituus, kuten **Virhe. Viitteen lähdettä ei löytynyt.**.

PHP:ssä injektion suojaamiseen käytettiin yleensä mysql_real_escape_string-funktiota, joka automaattisesti lisää kenoviivat lainausmerkkeihin. Muuten nykyään PDO-kirjaston metodilla SQL-injektiolta torjuminen onnistuu helpommin ja tehokkaammin verrattuna mysql-yhteyteen.

Esimerkkikoodi 4 käytetään PDO-kirjaston metodeja, joilla torjutaan SQL-injektiolta. SQL-lauseen valmisteluvaiheessa ei laiteta suoraan muuttujaa tai saatua syötettä vaan muuttuvien parametrien tilalle merkitään joko kysymysmerkki (?) tai tunniste (:username). Tässä esimerkissä käytin tunnisteetta. Kysely valmistetaan ensin metodil-

la prepare, sitten se suoritetaan metodilla execute. PDO-kirjaston prepare-metodilla tarkistetaan SQL-lauseita. Mikäli kyselyistä löytyy virheitä, sen suoritus loppuu siihen eikä se tulosta mitään. Kyselyä suorittaessa tunniste korvataan oikealla arvolla turvallisesti.

```
$query = " SELECT 1 FROM users WHERE username = :username" ;
 $query_params = array(':username' => $clean['username']);
 try
 {
     $stmt = $db->prepare($query);
     $result = $stmt->execute($query_params);
 }
 catch(PDOException $ex)
 {
    die("Kyselyn suoritus epäonnistui: " . $ex->getMessage());
 }
```

Esimerkkikoodi 4. SQL-injektiolta suojautuminen.

XSS-hyökkäyksiltä suojautuminen

XSS-hyökkäyksiltä voidaan suojautua yksinkertaisesti tunnistamalla syötteistä erikoismerkit. Monessa tapauksessa riittäisi vain (<)-merkin ja lainausmerkkien suodattaminen. Yleinen virhe onkin se, että suojataan ainoastaan <script>-tagia vastaan. XSS-hyökkäyksen pystyy nykyisin suorittamaan usealla eri tavalla. Esimerkiksi skripti voidaan lisätä <img> tagin kuvapolun sisään seuraavalla tavalla: [32.]

”<img src=”javascript:alert(’XSS’);”> ”

Melkein jokaisessa web-ohjelmointikielissä ja rajapinnoissa löytyy valmiita funktioita XSS-hyökkäystä vastaan. [32.] PHP:ssä löytyy myös valmiita funktioita, kuten htmlentities(), strip_tags() ja utf8_decode()-funktioita, joiden avulla voidaan suojautua XSS-hyökkäyksiltä. Erityismerkit voidaan muuttaa html-muotoon käyttäen funktioita htmlntities tai vaihtoehtoista funktiota htmlspecialchars. Lisäksi strip_tags poistaa merkkijonosta kaikki html- ja php-tagit. Funktio utf8_decode muuttaa tekstit tallennettaessa ISO-8859-1-muotoon.

```
submitted_username = '';

$submitted_username=htmlentities(['username'],ENT_QUOTES',  UTF-
8');
```

Esimerkkikoodi 5. XSS-hyökkäyksiltä suojaaminen käyttäen htmlntites-funktiota.

Esimerkkikoodi 5 nähdään htmlentities()-funktion käyttöä. Kyse on sisäänkirjautumisesta, jossa käyttäjä syöttää käyttäjätunnuksen ja salasanan. Käyttäjän syöttäessä kelpaamattoman salasanan tulostetaan käyttäjätunnusta uudestaan, kunnes käyttäjä syöttää oikean salasanan. Htmlentities()-funktion avulla html-koodit näkyvät sivuilla samanlaisena kuin editorissa.

4.1.4 Salasanojen suojaus

Kun käyttäjä rekisteröityy johonkin palveluun, rekisteröintivaiheessa määritetään jokin salasana. Myöhemmin salasana käytetään käyttäjänimen kera käyttäjän tunnistamiseen. [36, s. 290.] Kun käyttäjä lähettää rekisteröintilomakkeen, tiedot tallennetaan tietokantaan. Salasanaa ei koskaan tallenneta tietokantaan selväkielisenä tekstinä, koska SQL-injektiolla hyökkääjä saa ottaa koko listan.

Salasanojen tallentamisessa käytetään usein tiivistealgoritmeja, joten luovat kiinteämittaisen lyhyen tarkistussumman alkuperäisestä tiedosta. Tunnettuja tiivistealgoritmeja ovat MD5 (Message Digest 5), SHA-1 (Secure Hash Algorithm) ja SHA-2.

MD5 on suosittu tiivistealgoritmi, joka tuottaa 32 merkkiä pitkää merkkisarjaa (128 bittinen) vastaavan tiivisteen. Esimerkiksi lauseesta ”tämä on opinnäytetyö” syntyy 46236881efe5e96b51b3b01f07d35996 heksadesimaalimerkkisarja. Jos edellisestä lauseesta yksikin kirjain muuttuu, niin koko tiiviste muuttuu. Tiivistettä ei ole todennäköistä saada takaisin sanasanaksi. Tästä johtuen salasanan kadottaessa saa tilalle uuden salasanan. [23, 13, s. 265.]

SHA-1- ja SHA-2-algoritmit ovat uudempia kuin MD5. SHA-1-tiiviste tuottaa 160-bittisen tiivisteen eli 40 heksadesimaalisen merkkijonon ja se on MD5:tä turvallisempi.

[13, s. 266.] SHA-2 on parannettu versio edellisestä ja itse asiassa keskenään samantyyppinen tiivistealgoritmi, mutta tuottaa pidemmän tiivistearvon (224-, 256-, 384- tai 512-bittinen). Sitä voidaan kutsua myös nimillä SHA-224, SHA-256, SHA-384- ja SHA-512. [33.] Suositeltavia valintoja ovat 256- ja 512-bittiset tiivistealgoritmit. Itse olen käyttänyt esimerkkisovelluksessa SHA-256-algoritmia salasanojen salaamiseen.

Nykyään tavallisimmat tiivisteet SHA ja MD5 on molemmat kyetty murtamaan yksinkertaisesti tehokkailla tietokoneilla, varsinkin näytönohjaimilla, jotka voivat suorittaa tuhansia säikeitä samaan aikaan. Murtamalla tarkoitetaan sitä, että kyetään generoimaan muutamassa tunnissa toinen viesti, jolla on sama tiiviste kuin alkuperäisellä. [23.] Salasanan murtamisessa käytetään usein kahta erilaista murtamiskeinoa. Ne ovat sanakirjahyökkäys ja ns. brute-force-hyökkäys. [37.]

Sanakirjahyökkäys tarkoittaa sitä, että murto-ohjelmalle syötetään lista sanoista ja niiden avulla yritetään löytää oikea salasana. Ohjelma testaa läpi koko listan käyttämällä niitä sanoja, kunnes oikea salasana löytyy. [37.] Yksi vaihtoehtoisista sanakirjahyökkäysmenetelmistä on sateenkaritaulukot (rainbow tables), jotka sisältävät valmiiksi eri sanoista ja sanayhdistelmistä laskettuja tiivisteitä. Salasanatiedostossa olevia tiivisteitä verrataan taulukon tiivisteisiin.

Toisaalta brute-force-hyökkäyksestä ei ole mitään listoja. Tämän hyökkäyksen tarkoituksena on kokeilla kaikki mahdolliset merkkiyhdistelmät yksi kerrallaan, kunnes löytyy oikea merkkijono. Tästä syystä hyökkäys on saanut nimen brute force = raaka voima. [37.]

Pituus ja monimutkaisuus ovat keskeisiä tekijöitä suojattaessa salasanan arvauksilta. Esimerkiksi rekisteröintilomakkeessani olen edellyttänyt käyttäjältä sellaista salasana, jonka tulisi olla vähintään 8 merkkiä pitkä. Tämän lisäksi sen pitää sisältää isoja ja pieniä kirjaimia tai isoja tai pieniä kirjaimia ja numeroita (Esimerkkikoodi 6).

```
if(empty($_POST['password']))
{
    die("Salasana ei saa olla tyhjä.");
}else if(!preg_match("([a-z][A-Z]|[A-Z][a-z]|[0-9][a-z]|[0-
9][A-Z]|[a-z][0-9]|[A-Z][0-9])", $_POST['password']))
{
    die("salasanan pitää sisältää isoja ja pieniä kirjaimia
tai joko isoja tai pieniä kirjaimia ja numeroita. ");
}else if(strlen($_POST['password']) < 8 )
{
    die("Salasanassa pitää olla vähintään 8 merkkiä.");
}else{
    $clean['username'] = $_POST['username'];
}
```

Esimerkkikoodi 6. Salasanan validointi.

Salasanojen suojaamisessa viisas tapa on valmistautua hyökkäykseen ja minimoida sen vaikutukset, ennen kuin se tapahtuu. Salasanojen tietoturvaa voidaan parantaa "suolaamalla". Suola on satunnaisesti generoitu merkkijono, joka käytetään ennen tiivisteen luomista. Sen tarkoitus on estää etukäteen laskettujen murtotaulukoiden käyttöä, esimerkiksi sateenkaritaulukkoa. [38.]

```
$salt = dechex(mt_rand(0, 2147483647)) . dechex(mt_rand(0,
2147483647));

//salasana varastoidaan turvallisesti tietokantaan
//käyttämällä hash()-funktio suolan kanssa
$password = hash('sha256', $_POST['password'] . $salt);

for($round = 0; $round < 65536; $round++)
{
   $password = hash('sha256', $password . $salt);
}
```

Esimerkkikoodi 7. Salasanan suojaaminen. [23.]

Esimerkkikoodi 7 salasana- ja suolayhdistelmää kierretään 65 536 kertaa SHA-256-tiivistealgoritmilla. Tiivistealgoritmin pyörittäminen 65 536 kertaa vaikeuttaa hyökkäystä, sillä hyökkääjän täytyy toistaa sama vaihetta, jotta hän saa yhden tiivisteen tarkistettua. [23.]

4.1.5 Istunnon suojaus

Istuntojen varastamista voidaan vaikeuttaa generoimalla istunto id uudelleen käyttäjän kirjautumisen jälkeen. [33.] Seuraavassa Esimerkkikoodi 8 on esitetty istunnon id:n uudelleengenerointia.

```
<?php
session_start(); // aloitetaan istunto

session_regenerate_id(true); //korvataan id uudella id:llä ja
//poistetaan id evästeistä ja toimivien listasta
```

Esimerkkikoodi 8. Istunnon id uudelleengenerointi

Ennen versiota PHP 5.1 oli käytössä vain ominaisuus ”session_regenerate_id”, joka luo toki uuden istunnon id:n, mutta silti säilyttää vanhan. Korjatussa versiossa oleva ”session_regenerate_id(true)” on tietoturvallisempi johtuen siitä, että uudelleen-generoimisen lisäksi se poistaa myös edellisen istunnon id:n evästeistä ja toimivien listasta.

Istunnon id:n kuljettaminen evästeessä on turvallisempi tapa, sillä sieltä kaappaajan on merkittävästi vaikeampi saada se haltuunsa. Istuntotieto säilyy palvelimen asetuksista riippuvan ajan istuntotunnisteen avulla saatavissa. Sen vuoksi palveluun täytyy aina toteuttaa uloskirjautumistoiminto, joka tuhoaa istuntotiedon. Kuitenkaan käyttäjää ei voida pakottaa käyttämään tätä toimintoa.

Istuntojen kaappaamista voidaan ehkäistä myös sillä, että istuntoon tallennetaan IP-osoite sekä user-agent-määre. Näitä tietoja sitten tarkistetaan kirjautumisen tarkistuksen yhteydessä, että ovatko ne samoja kuin kirjautumishetkellä. [33.]

4.2 Web-palvelimen tietoturva

Web-palvelin tarkoittaa tietokonetta tai ohjelmistoa, joka vastaa asiakkaiden eli WWW-selaimien lähettämiin tiedostopyyntöihin. Suosituin web-palvelinohjelmisto on Apache, joka toimii sekä Windowsissa että Linuxissa. Sen sijaan toiseksi suosituin web-palvelinohjelmisto on Microsoft IIS, joka on tunnettu etenkin yritysjärjestelmissä. Molemmat palvelimet ovat helppokäyttöisiä ja ilmaisia.

Tietoturvallisuuden kannalta tiedonsiirrossa on salaamisen käyttäminen suositeltava. On huomioitava, että arkaluontoinen tieto siirretään verkon ylitse salakirjoitetussa muodossa. On käytössä erilaisia salausprotokolloja, mutta käytetyimpiä ovat

- verkkoselaimessa SSL ja HTTPS
- tekstipohjaisissa pääteyhteyksissä SSH ja SSH2
- tiedostojen siirrossa SCP ja SFTP.

Kyseisten protokollien avulla voimme salata (kryptata) verkkoliikenteen siten, ettei verkon nuuskija saa selville datapakettien sisältöä. Seuraavassa tutustutaan verkkoselaimessa käytettäviin salauksiin.

SSL

SSL (Secure Sockets Layer), joka tunnetaan nykyisin nimellä TLS (Transport Layer Security), on Netscapen kehittämä maailmanlaajuisesti hyväksytty salausprotokolla. Se mahdollistaa suojatun yhteyden käytön asiakkaan, koneen ja palvelimen välillä. [39.]

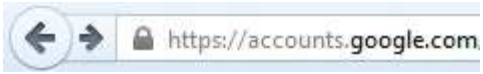
SSL:n tavoitteena on saavuttaa tiedon luottamuksellisuus ja autenttisuus, jolla tarkoitetaan tietojärjestelmän käyttäjien ja siihen osallistuvien laitteiden luotettava tunnistusta. SSL:n käyttämät algoritmit ovat RC4, RCZ, RSA, MD5. SSL:ssä käytetään kryptograafisia algoritmeja kolmessa eri tarkoituksessa: avainten vaihdossa, yhteyden salauksessa ja tiivistefunktioiden laskennassa. [40.]

Yhteyden salauksen lisäksi SSL mahdollistaa myös kommunikoivien osapuolien todentamisen. Siihen SSL voi käyttää RSA-avainta ja julkisia sertifikaatteja. Todentaminen

tapahtuu sertifikaattien eli varmenteiden avulla siten, että palvelimella on oma palvelinvarmenne, jonka perusteella käyttäjä voi varmistua kommunikoivansa oikean web-palvelimen kanssa. SSL:ää käytetään laajasti muun muassa sähköisen kaupankäynnin yhteydessä.

SSL:stä löytyy erilasia versioita. Selainohjelmistojen uusimmat versiot käyttävät yleensä 128-bitistä salausta. Mitä isompi luku on, sitä vaikeampi salausta on murtaa. 256-bittinen salaus on monin verroin varmempi kuin 40-bittinen salaus. [41.]

Voidaan helposti tarkistaa, onko sivu suojattu SSL-salauksella. Jos selaimessa näkyy lukkokuvake (Kuva 9), on olemassa SSL-yhteys. Varmistamisen vuoksi voidaan myös klikata kuvaketta, joka antaa lisätietoja suojatusta yhteydestä. Sen lisäksi SSL-salausta käyttävät verkko-osoitteet alkavat etuliitteellä https: etuliitteen http: sijaan.



Kuva 9. SSL-yhteys on muodostettu verkkosivuilla.

HTTPS

HTTP-protokolla on turvaton, koska viestintä lähetetään salaamattomana tekstinä. HTTPS (Hypertext Transfer Protocol Secure) on sen kryptattu versio, joka hyödyntää SSL-suojausta. HTTPS käytetään tiedon suojattuun siirtoon internetissä. Kun käyttäjä muodostaa yhteyden sivuston kautta, salaa HTTPS-sivusto istunnon digitaalisen sertifikaatin. HTTPS käyttää RSA, IDEA ja MD5 algoritmejä.

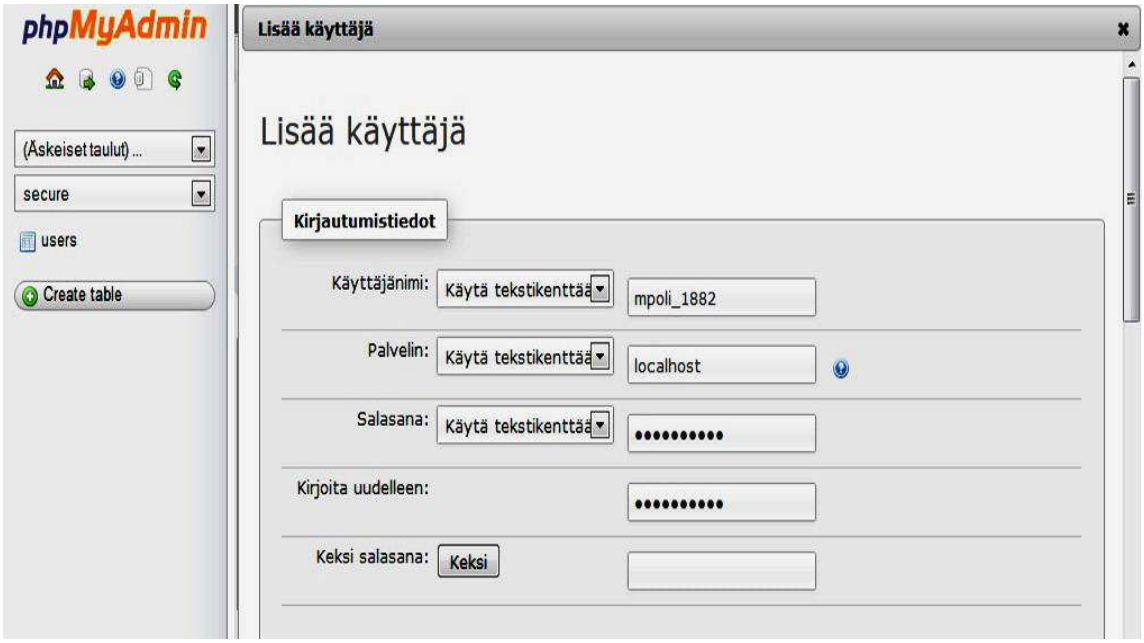
4.3 MySQL:n turvaaminen

Tässä luvussa pyritään esittelemään MySQL:n tietoturvaa. Erityisesti käsitellään käyttöoikeuksien hallintaa sekä tietokannan varmuuskopiointi. Muuten ei enää mainita SQL-injektioita, koska ne on käsitelty aikaisemmissa luvuissa.

4.3.1 Käyttöoikeudet ja niiden hallinta

Sovellusta varten luodessaan tietokanta ja tietokantatunnukset on tärkeä antaa tunnuksille mahdollisimman rajatut käyttöoikeudet. Näin minimoidaan tietoturvariskejä ja rajoitetaan vahinkoa ainakin vain yhteen tietokantaan. [42.] Tietokannan käyttäjätunnukset koostuvat käyttäjänimestä, isäntäkoneen nimestä ja salasanasta.

Nykyään uuden käyttäjän lisääminen ja käyttöoikeuden määrittäminen/muokkaminen onnistuu helpommin käyttäen phpMyAdminia, joka on PHP-kielellä rakennettu, MySQL-tietokantojen ylläpitoon tarkoitettu hallintatyökalu.



Kuva 10. Uuden käyttäjän luominen käyttäen työkalua phpMyAdmin.

Luodaan ”secure”-tietokannelle käyttäjä, jolla annetaan nimi mpoli_1882. Salasanaksi asetetaan "eHTb7%Pxa9" ja määritellään, että käyttäjä voi ottaa yhteyttä paikalliselta koneelta, kuten ”localhostilta” (Kuva 10). Pystytään määrittämään, kuka ja mistä kyseinen henkilö voi yhdistää tietokantaan.

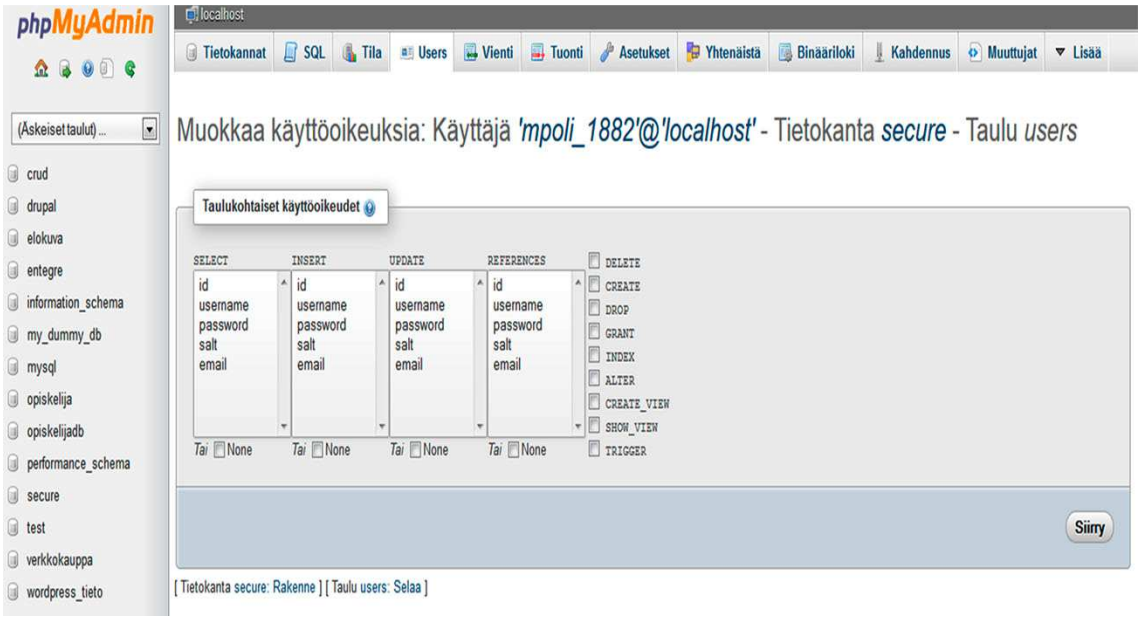
Käyttäjänimi on yksityinen tunniste, joka saa ottaa yhteyden tietokantapalvelimeen. Käyttäjätunnuksen turvaamista varten käytetään salasanaa, joka tallennetaan kryptatussa muodossa, eli niitä ei voida noutaa. Versiosta 4.1 lähtien salasanan tallentamiseen edellyttävien bittien määrä nousi 128 bitistä 328 bittiin. [34, s. 596]

Isäntäkoneen nimessä määritellään, miltä koneelta käyttäjä saa ottaa yhteyden. Se voidaan määritellä paikallisina koneina (localhost) tai koneen IP-osoitteina tai jokerimerkkinä (%). [34, s. 596.] Jokerimerkki tarkoittaa, että yhteyden ottaminen on sallittu kaikilta koneilta.

Uutta käyttäjää lisättäessä voidaan myös määrittää käyttöoikeuksia phpMyAdminissa laittamalla ruksin ruutuun (Kuva 11). Uuden käyttäjän luomisen jälkeen on mahdollista myös määrittää käyttäjälle käyttöoikeudet. Oikeuksilla tarkoitetaan sitä, miten käyttäjät pääsevät käsiksi tietokantoihin ja taulukoihin.

Kuva 11. Käyttöoikeuksien määrittäminen.

Oikeuksien myöntämisperiaatteena tulee olla, että käyttäjälle sallitaan mahdollisimman vähän oikeuksia siten, että käyttäjä voi kuitenkin tehdä hänelle tarpeellisia toimenpiteitä tietokannan tiedoille. Pitää tunnistaa tietokannan käyttäjän tarvittavat käyttöoikeudet ja sen mukaan pitää rajoittaa oikeuksia. Käyttötarpeiden muuttuessa voi muokata käyttöoikeuksia helposti. Voidaan myös määritellä oikeudet taulukon mukaisesti (Kuva 12).



Kuva 12. Oikeuksien myöntäminen taulukolle.

Taulukossa 2 on mahdolliset eri oikeudet ja niiden sallimat toiminnot.

Taulukko 2. MySQL-tietokannan oikeudet ja niiden sallimat toiminnot

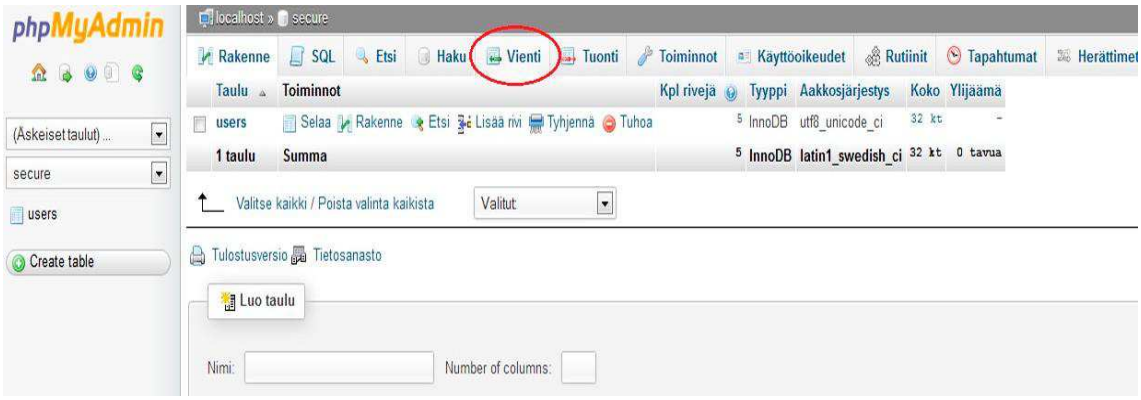
Oikeus	**Toiminta**
ALTER	Taulujen, sarakkeiden (rakenteen) ja indeksien muokkaus.
CREATE	Tietokantojen ja taulujen luonti.
DELETE	Tietojen poisto tauluista.
DROP	Taulujen ja tietokantojen poistaminen
FILE	Tiedostojen luku ja kirjoitus, esim. mysqldump-käyttö skriptin luomiseksi.
INDEX	Indeksien luonti ja poisto
INSERT	Tietojen lisääminen tauluun.
PROCESS	Käynnissä olevien prosessien seuraaminen ja pysäytäminen.
RELOAD	Käyttöoikeuksien päivittäminen (voimaansaattaminen) tietokantaan.
SELECT	Tietojen haku tauluista.
SHUTDOWN	Tietokantapalvelimen pysäyttäminen
UPDATE	Tauluissa olevien tietojen päivittäminen (muokkaaminen).

Esimerkiksi oikeuksia DROP, PROCESS, RELOAD ja SHUTDOWN ei tavallisesti pitäisi myöntää tietokannan käyttäjille. Toisaalta lauseiden INSERT-, ALTER-, UPDATE- ja DELETE-oikeuksia kannattaa antaa sen mukaan, millaisia tietojen päivitystarpeita käyttäjillä on.

4.3.2 Varmuuskopiointi

Mikään tallennusväline ei ole ikuinen. Tietokanta voi tuhoutua jostain syystä. Sen takia on tärkeää ottaa tietokannasta varmuuskopioita.

Tietokannan varmuuskopiointikin on yksinkertaista phpMyAdminin avulla. Varmuuskopiointi suoritetaan hallintapaneelista kohdasta vienti (Kuva 13). Tietokanta tuodaan takaisin tuontisivun kautta.



Kuva 13. Tietokannan varmuuskopiointi.

5 Yhteenveto

Opinnäytetyössä käytiin läpi yleisesti tietoturvaa, hyökkäysmenetelmiä ja hyökkäyksien suojaamismenetelmiä. Tietoturvauhkien määriä on lukuisia, siksi minun pitäisi rajoittaa aihetta tärkeämpien ja yleisimpien tietoturvauhkien mukaan. Tästä johtuen olen esitellyt työssä OWASP Top 10 -listan kolme yleisintä tietoturvauhkaa.

Työssäni suunnittelin ja toteutin yksinkertaisia tietokantapohjaisia PHP-sivuja, joissa käytin keinoja tietoturvariskien parantamiseksi. Tietokantayhteyden luomista ja kyselyiden suorittamista varten käytin PDO:ta. Suosittelen myös muille käyttää PDO:ta SQL-kielen sijaan tietoturvan kannalta, erityisesti SQL-injektion estämistä varten. Itse asiassa tässä työssä olen tutustunut PDO:hin, jonka syntaksi on vähän vaikeampaa. Tästä johtuen ajastani meni jonkin verran sen opettelemiseen.

Tietoturvauhkien estämisen kaikkein todennäköisin ja keskimmäisin tapa on varmistaa aina käyttäjän syöttämät tiedot. Hyvä sääntö on pitää jokaista vierailijaa potentiaalisena hyökkääjänä. Suodattamalla käyttäjien syötteet voidaan minimoida tietoturvariskejä. PHP:n direktiivejä on otettava myös huomioon, eli mitkä direktiivit laitetaan päälle tai pois päältä. Järkevä tapa tallentaa salasana sisäänkirjautumista varten on käyttää erityistä salasanatiivistealgoritmia. Sen lisäksi pitää pakottaa käyttäjä antamaan riittävän pitkää ja monimutkainen salasana.

Tietoturvauhkien määrä ja laajuus jatkoivat kasvamistaan. Sen takia pitää seurata koko ajan tietoturva-asioita. Sen lisäksi pitää olla ajan tasalla ohjelmistojen päivityksien kanssa, koska jatkuvasti löytyy joiltain osin korjattavaa tai parannettavaa.

Yhteenvetona voin havaita, että opinnäytetyö voi auttaa ohjelmoijia, jotka haluavat kehittää tietoturvallista web-ohjelmointia. Toisaalta aihe oli suhteellisen laaja ja oli vaikea vastata ihan kaikkiin tietoturvallisuuteen liittyviin asioihin.

Lähteet

1 Tietoturva 2006 [verkkodokumentti, viitattu 17.10.2012]. Saatavissa: http://hyl.edu.hel.fi/~petri/petri/oppilastyot2006/9at1/JaakkoR/nettisivut/tietoturva.html.

2 Yleisesti tietoturvasta. Oulun yliopisto 2006 [verkkodokumentti, viitattu 18.10.2012]. Saatavissa: http://www.oulu.fi/tietohallinto/tietturva/sisalto/kannettavientietoturva/tietoturvasta.html.

3 Rosendahl, Mauri, Tietoturva palvelee kaikkia 2003 [verkkodokumentti, viitattu 18.10.2012]. Saatavissa: http://www.helsinki.fi/atk/lehdet/103/Tietoturva%20palvelee%20kaikkia.html.

4 Behm, Jukka, Tietoturvaviikko 2009 [verkkodokumentti, viitattu 20.10.2012]. Saatavissa: http://www.tietoturvaopas.fi/attachments/5EtXRbSHD/Tietoturvapaiva_01_links.pdf.

5 Järvinen, Petteri 2002. Tietoturva & Yksityisyys. 1.painos. Jyväskylä: Docendo.

6 Kurtti, Niko, RAID-tekniikka 2010 [verkkodokumentti, viitattu 21.3.2013]. Saatavissa: https://jop.cs.tut.fi/twiki/bin/view/Tietoturva/RAID-tekniikka%282-A%29.

7 Sähkön ammattilainen 2008 [verkkodokumentti, viitattu 21.10.2012]. Saatavissa: http://www.utuelec.fi/sivu.aspx?taso=2&id=395.

8 Html-kieli [verkkodokumentti, viitattu 21.10.2012]. Saatavissa: http://www.mvnet.fi/index.php?osio=Kotisivun_teko&sivu=HTML-kieli.

9 Korpela, Jukka K. 2012, HTML5-kirja [verkkodokumentti, viitattu 27.2.2013] Saatavissa: http://html5kirja.fi/2012/08/31/mita-html5-sovellukset-ovat/.

10 Polso, Kristian 2012, PHP:stä julkaistu versio 5.4 [verkkodokumentti, viitattu 10.11.2012]. Saatavissa: http://www.vaiste.com/fi/blogi/phpsta-julkaistu-versio-54.

11 Johdatus Php-kieleen [verkkodokumentti, viitattu 29.10.2012]. Saatavissa: http://users.jyu.fi/~kolli/ITK215_05/php/.

12 Koulutus- ja konsultointipalvelu KK Mediat 2012 [verkkodokumentti, viitattu 31.10.2012]. Saatavissa: http://www.2kmediat.com/php/.

13 Heinisuo, H., Rauta I. 2007. PHP ja MySQL Tietokantapohjaiset verkkopalvelut. 4.uudisteetu painos Helsinki:Talentum.

14 MySQL, Wikipedia 2012 [verkkodokumentti, viitattu 12.11.2012]. Saatavissa: http://fi.wikipedia.org/wiki/MySQL.

15 MySQL News Announcements 2013 [verkkodokumentti, viitattu 27.3.2013]. Saatavissa: http://www.mysql.com/news-and-events/.

16 MySQL:n kotisivu 2012 [verkkodokumentti, viitattu 13.11.2012]. Saatavissa: http://www.mysql.com/.

17 Kolehmainen, Aleksi, Wikipedia hylkää MySQL:n 2012 [verkkodokumentti, viitattu 21.3.2013]. Saatavissa: http://www.tietoviikko.fi/kaikki_uutiset/wikipedia+hylkaa+mysqln/a866352.

18 Kolehmainen, Aleksi, Monty Widenius: MySQL käyttää vanhaa koodia 2013 [verkkodokumentti, viitattu 21.3.2013]. Saatavissa: http://www.tietoviikko.fi/kaikki_uutiset/monty+widenius+mysql+kayttaa+vanhaa+koodia/a878553.

19 It-viikko, Wikipedia siirtyy suomalaiseen tietokantaan 2012 [verkkodokumentti, viitattu 21.3.2013]. Saatavissa: http://www.itviikko.fi/ratkaisut/2012/12/19/wikipedia-siirtyy-suomalaiseen-tietokantaan/201244093/7?&n=1.

20 Jääskeläinen, Lassi, PDO vs MySQLi 2012 [verkkodokumentti, viitattu 19.11.2012]. Saatavissa: http://www.crodesoft.com/2012/08/31/pdo-vs-mysqli/.

21 Pelttari, Martti, Tietokantalähtöisen PHP-ohjelmoinnin tietoturvaoptimointi 2008 [opinnäytetyö, viitattu 28.03.2013]. Saatavissa: https://theseus17-kk.lib.helsinki.fi/bitstream/handle/10024/12212/PHP%20ja%20tietoturvaoptimointi%20Pelttari.pdf?sequence=2.

22 Joel, L. , Kristian, J. 2010 [verkkodokumentti, viitattu 25.11.2012]. Saatavissa: https://koppa.jyu.fi/kurssit/96852/luento/web-tietoturva.

23 Tilja, Niko, PHP-verkkokauppaohjelmisto MVC-arkkitehtuurilla 2012 [opinnäytetyö, viitattu 25.11.2012]. Saatavissa: http://publications.theseus.fi/bitstream/handle/10024/42207/Tilja_Niko.pdf?sequence=1 .

24 OWASP [verkkodokumentti, viitattu 27.2.2013]. Saatavissa: https://www.owasp.org.

25 Php Manual 2012 [verkkodokumentti, viitattu 30.11.2012]. Saatavissa: http://php.net/manual/en/.

26 Tietoturvaraportti 2012 [verkkodokumantti, viitattu 5.1.2013]. Saatavissa: http://www.kpmg.com/FI/fi/Ajankohtaista/Uutisia-ja-julkaisuja/Neuvontapalvelut/Documents/KPMG-Tietoturvaraportti-2012.pdf.

27 Mikä on sql injektio ja kuinka siltä suojaudutaan 2012 [verkkodokumentti, viitattu 15.1.2013]. Saatavissa: http://www.gelo.fi/mika-on-sql-injektio-ja-kuinka-silta-suojaudutaan/.

28 SQL Book 2007 [verkkodokumantti, viitattu 15.1.2013] Saatavissa: http://www.sqlbook.com/SQL/SQL-Injection-Attacks-29.aspx.

29 Auralinna, Tero, Verkkokaupan tietoturvallisuus 2005 [opinnäytetyö, viitattu 10.2.2013]. Saatavissa: http://www.auralinna.fi/files/verkkokaupan_tietoturvallisuus_2005.pdf.

30 Johdatus PHP-kieleen [verkkodokumentti, viitattu 10.2.2013]. Saatavissa: http://users.jyu.fi/~kolli/ITK215_05/php/?sivu=sessiot.

31 Huhtala, Pirkka, WWW-sovelluskehitys 2010 [opinnäytetyö, viitattu 10.2.2013]. Saatavissa: https://theseus17-kk.lib.helsinki.fi/bitstream/handle/10024/24468/Huhtala_Pirkka.pdf?sequence=1.

32 Jääskeläinen, Lassi, XSS-injektio 2012 [verkkodokumentti, viitattu 29.1.2013]. Saatavissa: http://www.crodesoft.com/2012/11/07/xss-injektio/.

33 Web Sovellusten Ohjelmointi [verkkodokumentti, viitattu 11.2.2013] Saatavissa: http://www.oamk.fi/sbc/www/johdanto.php.

34 Hakala, M., Vainio, M., Vuorinen, O. 2006. Tietoturvallisuuden käsikirja. 1.painos. Jyväskylä: Docendo.

35 Tolvanen, Sampo, PHP-hyökkäykset 2012 [verkkodokumentti, viitattu 25.2.2013]. Saatavissa: https://jop.cs.tut.fi/twiki/bin/view/Tietoturva/Tutkielmat/PhpHyokkaykset.

36 W.Jason Gilmore 2005. PHP&MySQL 5. 1.painos. Jyväskylä: Gummerus.

37 Tietoturva käytännössä 2013 [verkkodokumentti, viitattu 14.3.2013]. Saatavissa: http://www.tietojesiturvaksi.fi/content/salasanan-murtaminen.

38 Lavitt, Samuel, Mikä tiivistefunktio on vielä turvallinen? 2011 [verkkodokumentti, viitattu: 11.3.2013]. Saatavissa: http://www.nixu.com/blogi/tagit/sha-1.

39 Wikipedia TLS 2013 [verkkodokumentti, viitattu 22.3.2013]. Saatavissa: http://fi.wikipedia.org/wiki/TLS.

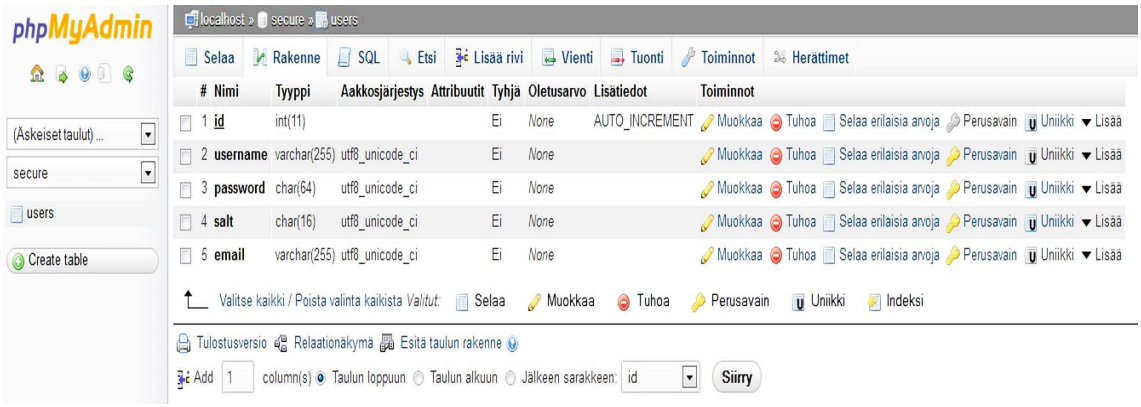
40 Elo, Tommi, SSL-protokollan yleiskuvaus 1997 [verkkodokumentti, viitattu 22.3.2013]. Saatavissa: http://www.tml.tkk.fi/Studies/Tik-110.300/1997/Essays/ssl.html.

41 Sosiaali- ja terveyspalvelukeskuksen tietoturva 2013 [verkkodokumentti, viitattu 22.3.2013]. Saatavissa: http://www.sosiaalikollega.fi/asiakkaat/SSL.pdf.

42 Wikipedia HTTPS 2013 [verkkodokumentti, viitattu 22.3.2013]. Saatavissa: http://fi.wikipedia.org/wiki/HTTPS.

Tietokanta

”secure”-tietokannan users-taulukko



Sovelluksen lähdekoodit

Register.php

```
<?php

// Luodaan yhteys tietokantaan ja istunto alkaa
require("C:/wamp/includes/yhteys.php");

if(!empty($_POST)){

$clean = array();

   if(empty($_POST['username'])){

      die("Käyttäjätunnus ei saa olla tyhjä. ");

   }else if( strlen($_POST["username"]) < 4 ){

      die("Käytäjätunnuksessa pitää olla vähintään 4 merkkiä ");

   }else if(!preg_match('/^[a-zA-Z0-9]+[.-_]*
          [a-zA-Z0-9]$/',$_POST['username'])){

      die("Käyttäjätunnuksen alussa ja lopussa on sallittu
       vain kirjaimia ja numeroita, toisaalta se voi sisältää
       väliviiva, alaviiva sekä piste.");

   }else {
       $clean['username'] = strip_tags($_POST["username"]);
   }

   if(empty($_POST['password'])){
        die("Salasana ei saa olla tyhjä.");

   }else if(!preg_match("([a-z][A-Z]|[A-Z][a-z]|[0-9][a-z]|
```

```
[0-9][A-Z]|[a-z][0-9]|[A-Z][0-9])", $_POST['password'])){

     die("salasanan pitää sisältää isoja ja pieniä
     kirjaimia tai joko isoja tai pieniä kirjaimia ja
     numeroita. ");

}else if(strlen($_POST['password']) < 8 ){

     die("Salasanassa pitää olla vähintään 8 merkkiä.");

}else{
     $clean['username'] = $_POST['username'];
}

// Varmistetaan että käyttäjä on antanut kelvollisen
//sähköpostiosoiten
if(filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)){

     $clean['email'] = $_POST['email'];
}else{
     die("Virheellinen sähköpostiosoite");
}

// Suoritetaan SQL-kysely, onko olemassa jo käyttäjätunnus
//:username on erityinen symboli, kyselyn suorittaessa
//korvataan sitä oikealla arvolla
$query = "SELECT 1 FROM users WHERE username = :username";

//Suojataan SQL-injektiolta käyttämällä tunniste (token)
$query_params = array(':username' => $clean['username']);

try{
    $stmt = $db->prepare($query);
    $result = $stmt->execute($query_params);

}catch(PDOException $ex){
```

```
     die("Kyselyn suoritus epäonnistui: ");
}
$row = $stmt->fetch();

//jos rivi palautui, tiedämme että löytyi tietty
//käyttäjätunnus
if($row){
      die("Käyttäjätunnus on jo olemassa");
}

//Tarkistetaan samalla tavalla sähköpostiosoiten, että se on
//ainutlaatuinen
$query = "SELECT 1 FROM users WHERE email = :email";

$query_params = array(':email' => $clean['email']);

try{
        $stmt = $db->prepare($query);
        $result = $stmt->execute($query_params);
}catch(PDOException $ex){
        die("Failed to run query: " . $ex->getMessage());
}
$row = $stmt->fetch();
if($row){
        die("Tämä sähköpostiosite on jo rekisteröity");
 }

//Jälleen käytämme erityiset tunnisteet, jotka
//suojaavat SQL-injektiolta
    $query = "INSERT INTO users (
            username,
            password,
            salt,
            email
        ) VALUES (
            :username,
```

```
                :password,
                :salt,
                :email)";

    //Suola suojaa sateenkaritaulukko hyökkäyksiltä
    $salt = dechex(mt_rand(0, 2147483647)) .
    dechex(mt_rand(0,    2147483647));

    //salasana varastoidaan turvallisesti tietokantaan
    //käyttämällä hash()-funktiota suolan kanssa
    $password = hash('sha256', $_POST['password'] . $salt);

    for($round = 0; $round < 65536; $round++){
          $password = hash('sha256', $password . $salt);
    }
    $query_params = array(
            ':username' => $clean['username'],
            ':password' => $password,
            ':salt' => $salt,
            ':email' => $clean['email']);
    try{
            $stmt = $db->prepare($query);
            $result = $stmt->execute($query_params);

    }catch(PDOException $ex){

            die("Kyselyn suoritus epäonnistui: ");
    }
    header("Location: login.php");
    die("Redirecting to login.php");
}

?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-
8" />
<title></title>
<link   rel="stylesheet"   type="text/css"   href="css/register-
style.css" />
</head>

<body>
<h1>Rekisteröityminen</h1>

<form class="form1" method="post" action="register.php">

    <label>Käyttäjätunnus<br />
    <input type="text" name="username" value="" /></label>
    <br /><br />
    <label>Salasana<br />
    <input type="password" name="password" value="" /></label>
    <br /><br />
    <label>Sähköposti<br />
    <input type="text" name="email" value="" /></label>
    <br /><br />
    <input type="submit" value="Tallenna" />
</form>
<p>
    <a href="http://validator.w3.org/check?uri=referer"><img
      src="http://www.w3.org/Icons/valid-xhtml10"     alt="Valid
XHTML 1.0 Transitional" height="31" width="88" /></a>
  </p>

</body>
</html>
```

Login.php

```
<?php

require("C:/wamp/includes/yhteys.php");

//Käyttäjätunnus alustetaan tyhjällä arvolla
$submitted_username = '';

if(!empty($_POST)){

     $query = "SELECT id,username,password,salt,email FROM
               users WHERE username = :username";

     $query_params = array(':username' => $_POST['username']);

     try{
          $stmt = $db->prepare($query);
          $result = $stmt->execute($query_params);

     }catch(PDOException $ex){
          die("Failed to run query: ");
     }
     $login_ok = false;

     //noudatetaan käyttäjän tiedot tietokannasta
     $row = $stmt->fetch();

     if($row){
     //tarkistetaan, että täsmääkö salasanat, jonka hash antanut
     //Verrataan, että hajautettu versio on jo tietokannassa
          $check_password = hash('sha256', $_POST['password'] .
          $row['salt']);

          for($round = 0; $round < 65536; $round++){
```

```
                    $check_password = hash('sha256',
                    $check_password . $row['salt']);
               }
               if($check_password === $row['password']){
               // Jos ne on tehty, käännetään tilanne true
                    $login_ok = true;
               }
          }
          //Jos sisäänkirjautuminen onnistuu, siirrytään käyttäjä
          //yksityiselle sivulle
          //muussa tapauksessa näytetään login lomake uudelleen

          if($login_ok){

          //Tässä valmistellaan $row rivi joukon tallentamista
          //$_SESSION sisään poistamalla suola ja salasana sieltä
          //Vaikka $_SESSION varastoidaan palvelin-puolella,
          //silti ei ole mitään syytä varatoida niitä
          //henkilökohatisia arvoja
               unset($row['salt']);
               unset($row['password']);

          //tämä tallentaa käyttäjän tiedot sessionid:ina "user"
               $_SESSION['user'] = $row;

          header("Location: private.php");
               die("Redirecting to: private.php");
          }else{
               print("Kirjautuminen epäonnistui.");

        //htmlentities estää xss-hyökkäyksiltä
        $submitted_username = htmlentities($_POST['username'],
        ENT_QUOTES, 'UTF-8');}
}
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-
8" />
<title></title>
<link    rel="stylesheet"    type="text/css"    href="css/login-
style.css" />
</head>

<body>
<h1>Kirjaudu Sisään</h1>
<form class="form-style" action="login.php" method="post" >

    <label for="login"> Käyttäjätunnus </label>
    <input  type="text"  name="username"  value="<?php  echo  $sub-
mitted_username; ?>" />

    <label for="password"> Salasana </label>
    <input type="password" name="password" value="" />

    <input type="submit" value="Kirjaudu Sisään" />
    <br /> <br />
    <a href="register.php">Rekisteröidy</a>
</form>
</body>
</html>
```

Private.php

```
<?php

require("C:/wamp/includes/yhteys.php");

//tarkistetaan onko käyttäjä kirjautunut sisään vai ei
if(empty($_SESSION['user'])){

          // Jos ei, ohjataan sisäänkirjautumis sivulle.
          header("Location: login.php");
          die("Redirecting to login.php");
}

?>

Hello  <?php  echo  htmlentities($_SESSION['user']['username'],
ENT_QUOTES, 'UTF-8'); ?>, secret content!<br />

<a href="logout.php">Kirjaudu ulos</a>
```