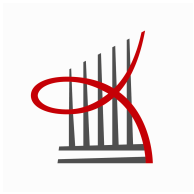


VERKKOSOVELLUSTEN TIETOTURVA

Kimmo Keskinen

Opinnäytetyö
Joulukuu 2010
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto

KESKINEN, KIMMO: Verkkosovellusten tietoturva

Opinnäytetyö 58 s.
Joulukuu 2010

Tämä opinnäytetyö tehtiin perehdytyksenä verkkosovelluksien tietoturvaan ja siinä käytettiin lähtökohtana PHP-sovelluskehitystä MySQL-tietokantaratkaisulla. Työn tarkoituksena oli vastata verkkosovellusten tietoturvallista kehittämistä koskevaan tutkimuskysymykseen verkkosovellusten tietoturvan eri osa-alueista kirja- ja verkkolähdekatsauksen pohjalta. Tässä tutkielmassa pyrittiin aiemman, tiukemmin rajatun tutkimuksen jatkoksi huomioimaan kokonaisuuksia suositeltavissa ratkaisuissa.

Keskeisiä tutkimusalueita olivat muun muassa salausmenetelmät ja niiden käyttö välitettävän tiedon luottamuksellisuuden, eheyden ja kiistämättömyyden tukemiseksi. Salatun yhteyden muodostaminen käyttäjän selaimen ja verkkosovelluspalvelimen välille osoittautui myös ratkaisevaksi tekijäksi useiden eri tietoturvaratkaisujen suhteen. verkkosovellusten tietoturvan kokonaisuuteen vaikuttavat säilytettävän tiedon määrä ja luotettavuuden aste, tietosyötteiden käsittely hyökkäysten varalta ja sovelluksen käytettävyyden turvaaminen suoritustehokkuutta parantamalla. Työssä esiteltiin ratkaisuja evästeiden ja istuntojen vahventamiseen, HTTPS-yhteyden käyttöönottoon ja merkitykseen, käyttäjäsyötteiden käsittelyyn sekä asiakas- että palvelinpuolen hyökkäystekniikoiden torjumiseksi, luottamukselliseen salasanojen hallintaan ja sekä verkkosovelluksen että PHP-tulkin asetustiedostojen tietoturvaan.

Yksittäiset ongelmat ja ratkaisut tietoturvallisuudessa eivät osoittautuneet teorialtaan tai käytännön ratkaisuiltaan erityisen haastaviksi. Koko tutkimuksen ajan painottuikin se, kuinka verkkosovelluksen eri osa-alueet ja haavoittuvuudet vaikuttavat toisiinsa ja kuinka useimpien ratkaisujen tulee heijastaa tätä kokonaistietoturvan näkemystä. Vaikuttaisi hälyttävältä, kuinka yksinkertaisia monet verkkosovelluksia koskevista ongelmista ovat, ja kuinka silti verkkokehityksen perusosaamiseen näyttäisi nykyään kuuluvan vain hyvin osittainen tuntemus tietoturvan kokonaisosaamisesta.

Asiasanat: Tietoturva, verkkoohjelmointi, PHP, tietokannat, salaus.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Engineering

KESKINEN, KIMMO: Data security of web applications

Bachelor's thesis 58 pages
December 2010

This thesis work was made as a study into web application data security. PHP and MySQL were chosen as easily understood language and database solutions for the platform and subject restrictions of research. Web applications are thriving even well after the initial Web 2.0 boom, but data security in their development is hazy and often lacking as the subject field keeps changing both technologically and due to the evolution of the internet. This bachelor's thesis made research into the field of data security in web development, attempting to understand it widely and sum up inter-related solutions for specified security threats to counter web application attacks and risks thereof.

The methodology of this thesis was largely related to literature study in related publications and online sources in cases where more concrete, up-to-date and practical discussion was better available. The results pointed to most solutions being simple in nature, and that the data security problematic was in the interconnection of multiple different technologies and solutions. A single solution may not be able to provide much needed data security at all unless it's supported by other solutions to invalidate other attack methods, errors and vulnerabilities. Determinining the required level of security for the application, securing data by crypting it into unreadable form while in-transmission, better hiding of secret configurations and related data and sanitization of user input were critical to achieving confidentiality, integrity and non-repudiation of message transmission over the web and in storing data securely.

News about online attacks and failures of data security in the past few years make it obvious how important a wide knowledge of security in web development is. There is much to do in educating the wide society as well as improving the overall safe computing solutions, but especially the web development society. Related education should teach a more thorough curriculum of data security theory and security techniques.

Key words: Data security, web development, PHP, database applications, cryptography, MySQL, Internet.

SISÄLLYS

1. JOHDANTO

Tässä luvussa on johdanto opinnäytetyön sisältöön sekä esittelyä tutkimuksen taustoista. Aihealueen ja tekniikoiden taustoja esitellään omissa alaluvuissaan 1.1-1.4. Tämä on on Tampereen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman ja ohjelmistotuotannon suuntautumisvaihtoehdon puolesta joulukuussa 2010 valmistunut opinnäytetyö. Opinnäytetyöohjaajana työlle on toiminut Jussi Pohjolainen.

verkkotekniikat ja standardit kehittyvät jatkuvasti, samoin kuin verkkosovelluksiin kohdistuvat hyökkäysriskit. Johtuen sekä alan jatkuvasta kehityksestä että tietoturvan laajuudesta oppiaiheena tietoturvaa ei sisällytetä ohjelmistotuotannon koulutukseen Tampereen ammattikorkeakoulussa kovin kattavasti. Opiskeltuani ja harjoiteltuani verkkokehitystä kiinnostuin tämän tärkeän osaamisalueen omatoimisesta täydentämisestä. Myöskin hiljattain julkaistu, Harvardin yliopiston opiskelijoiden, Bonneau & Preibuschin (2010) tekemä tutkielma osoittaa tietoturvan huutavan pulan – 150 eri verkkosivuston arviointi ja vertailu antoi ilmi merkittäviä määriä eritasoisia heikkouksia tietoturvaa koskevissa käytännöissä. Tämä opinnäytetyö onkin itsenäinen tutkielma verkkosovellusten tietoturvasta ohjelmistokehityksen näkökulmasta, ammattiosaamisen sekä henkilökohtaisen tietoturvan ymmärtämisen ja kehittämisen vuoksi myös verkkosovellusten käyttäjänä. Tutkielman tarkoitus on vastata tutkimuskysymykseen ”miten kehittää verkkosovelluksia tietoturvallisesti”

Työ on rajattu siten, että tietoturvaa käsitellään verkkosovellusten ohjelmistokehityksellisistä näkökulmista, koskien niiden suunnittelua ja toteutusta, esitellen verkkotekniikoiden haavoittuvuuksia ja ohjeita näiden käsittelyyn dynaamisessa verkkokehityksessä. Työn aihetta ei ole rajattu mihinkään palvelinpuolen tekniikkaan, mutta aihetta käsitellään verkkokehitykselle tavanomaisen asiakas-palvelin-tietokanta -arkkitehtuurin näkökulmasta ja esimerkkitarkoituksessa käsitellään dynaamisuudesta vastaavana palvelinpuolen ohjelmointikielenä PHP:a ja tietokantana MySQL:a. Työn ei ole tarkoitus keskittyä HTTP-palvelinohjelmistoon, mutta samoin kuin aiemmat tekniset rajaukset, voidaan teoreettisen kehitystyön olettaa toimivan Apache-palvelimen 2.2.11-version päällä ellei muuta mainita. Esimerkkiekniikoiden valinnat perustuvat niiden laajaan käyttöön ja tunnettuuteen, joten niiden tulisi toimia hyvänä lähestymispohjana aihealueeseen.

Aiemmat verkkosovellusten tietoturvaan liittyvät tutkimukset ovat pääasiassa tarkkoja ja kattavia rajauksia hyvin kapeaan ongelmaan (Liu, Huang & Gouda 2005; Sotirov ym. 2008). Vaihtoehtoisesti taas aiemmat opinnäytetyöt ovat käytännönläheisiä, mutta käsittelevät tietoturvaa esimerkiksi käyttöjärjestelmien (Salminen 2003), langattomien verkkojen (Tuominen 2005), yritysten (Sampajoki & Sihvo 2006; Jarmas 2007) tai julkaisualustojen (Haapamäki 2010) näkökulman ja ongelmien kautta. Vaikka Haapamäen (2010) tietoturva-arvio Joomla!-julkaisualustasta käsittelee kyllä osaltaan myös verkkosovelluksen tietoturvaa, tämän opinnäytetyön on tarkoitus olla tietystä julkaisualustasta riippumaton tutkielma. Tämän tutkielman tulokset antavat paremman pohjakäsityksen tietoturvan teoriaan ja ongelmiin, joiden ymmärtäminen auttaa ratkaisemaan tietoturvakysymyksiä myös eri julkaisualustoilla tai dynaamisen verkkokehityksen kielillä, johtuen monien ongelmien yhteydestä verkkosovelluksien toimintaa mahdollistaviin asiakaspuolen ja tietoliikenteen standardeihin ennemmin kuin suoraan PHP- ja MySQL-tekniikoiden toimintaan itsessään.

2. TAUSTAT

Tässä luvussa esitellään työhön ja sen aihepiiriin kuuluvia taustoja koskien tietoturvaa (luvussa 2.1 alalukuineen), verkkosovelluksia (2.2) ja tarvittaessa esimerkeissä käytettyjä verkkosovellustekniikoita: PHP:a (2.3) ja MySQL:a (2.4). Tietoturvan käsitteistöä, taustoja ja määrittelyä koskevat luvut ovat suurin painopiste taustoissa, koska niissä avataan ja määritetään termistöä, joka määrittää tutkimuskysymyksiä ja johdattelee tutkimustuloksiin.

2.1 Tietoturva

Tietoturva on riskien ennakointia ja niiden toteutumisen ehkäisyä yhtä lailla kuin tapahtuneen vahingon minimointia ja hallintaakin. Sekä tästä syystä että tietämättömyyden vuoksi tietoturva jää helposti ylenkatsotuksi tai unohdetuksi osaksi kattavaa ohjelmistokehitysprosessia. Se nähdään liian helposti ylimääräisenä kustannuksena ja turhana työnä, josta ei ole selkeää lisäarvoa palvelulle sen normaalin toiminnan aikana. Tietoturvan arvo näkyykin vasta silloin kun se on puutteellista ja siitä seuraa vahinkoa verkkopalvelun omistajalle, ylläpitävälle taholle tai käyttäjille, kun jokin väheksytty tai tiedostamaton riski toteutuu ylipäätänsä tai vakavampana kuin tietoturvan ollessa kunnossa. Tämän vuoksi on tärkeää eritellä, mitä ajatuksia tai periaatteita tietoturvan kokonaisuuteen määritellään, ja mitä uhkia verkkopalveluun voi kohdistua, ja huomioida näitä tekijöitä riittävällä varmuudella verkkopalvelua kehittäessä. (Hakala, Vainio & Vuorinen 2006, 3-6.)

Hakala, Vainio ja Vuorinen (2006) esittelevät tietoturvalle sekä klassisen että laajennetun määritelmän. Klassinen määritelmä nojautuu tiedon arvoon, jolloin sen keskeiset osatekijät ovat tiedon luottamuksellisuus, käytettävyys ja eheys. Laajennettu määritelmä käsittää näiden lisäksi kiistämättömyyden ja pääsynvalvonnan osatekijöissään. Luottamuksellisuus tarkoittaa tiedon olevan vain siihen oikeutettujen henkilöiden pääsyn siihen ja että tieto pysyy ainoastaan näiden oikeutettujen tahojen tiedossa. Käytettävyys tarkoittaa tiedon riittävän nopeaa ja oikean muotoista saatavuutta aina tarvittaessa. Eheys käsittää tietojen virheettömyyden ja paikkansapitävyyden. Kiistämättömyys on

eri järjestelmissä käyttäjien syöttämään ja muokkaamaan tietoon liittyvä ominaisuus, jossa varmistetaan että tunnistettu käyttäjä on kiistämättömästi vastuussa jostain tiedosta, sen muokkaamisesta tai käsittelystä. Pääsynvalvonta vuorostaan on joukko tietoteknisiä menetelmiä, joilla rajoitetaan sekä järjestelmien ulkopuolisten käyttäjien, että järjestelmän sisällä olevien mutta tiettyjen tietojen kannalta oikeuttamattomien käyttäjien pääsyä näihin tietoihin ja resursseihin. (Hakala, Vainio & Vuorinen 2006, 4-5.)

Samassa teoksessa tietoturvaa käsitellään laajana kokonaisuutena, erityisesti yritysten ja organisaatioiden näkökannalta. Tietoturvan käsitteen yhteydessä esitellään tietoturvapolitiikan ja tietoturvasuunnitelman muodostamisen ohjeistusta mahdollisimman kattavasti. Tietoturvallisuus itsessään jaotellaan monenlaisiin osa-alueisiin, sen käsittelyn ja paremman tietoturvan suunnittelun ositteluksi ja järkeistämiseksi. Hallinnollinen turvallisuus, fyysinen turvallisuus, ympäristöturvallisuus, henkilöstöturvallisuus, tietoaineistoturvallisuus, ohjelmistoturvallisuus, laitteistoturvallisuus, sekä tietoliikenneturvallisuus ovat kaikki tärkeitä osakokonaisuuksia modernin organisaation kokonaistietoturvassa. Näissä tietoturvaa käsitellään niin toimintaohjeiden, laitteistovalintojen, työpaikkasäännösten, ohjelmointi- ja muiden erilaisten ratkaisujen taholta. (Hakala, Vainio & Vuorinen 2006, 10-12, 304-314.)

Hakala, Vainio ja Vuorinen (2006) painottavat vielä että edellinen jaottelu on keinotekoinen, ja että eri kategorioiden välillä on vuorovaikutuksia keskenään. Tässä opinnäytetyössä keskitytään edellisestä listauksesta enimmäkseen ohjelmistoturvallisuutta vastaavaan osa-alueeseen verkkopalvelujen suunnittelun ja kehityksen näkökulmista.

Tietoturvan osa-alueet näkyvät verkkopalveluissa eri näkökulmina tietoturvaan. Luottamuksellisuus, käytettävyys, eheys, kiistämättömyys ja pääsynvalvonta näkyvät eri puolilla verkkopalvelun kehityksen ja toiminnan osa-alueita. Näistä osatekijöistä luottamuksellisuus, käytettävyys ja eheys kuuluvat klassisen tiedon arvoon perustuvaan määritelmään tietoturvallisuudesta. Nykyiseen laajennettuun tietoturvallisuuden määritelmään kuuluvat myös tietojärjestelmien tarkkailu- ja säätelyominaisuuksiin pohjaavat kiistämättömyys ja pääsynvalvonta. Myös tiedon säilytykseen tai tietojärjestelmien laitteistopuoleen liittyvä fyysinen turvallisuus voidaan katsoa osatekijäksi kokonaistietoturvallisuutta, vaikka se kuuluukin tämän tutkielman rajauksen ulkopuolel-

le, koskiessaan enemmän fyysisiä ja organisaatiosuunnitelmallisia tekijöitä kuin ohjelmistokehitystä. Lisäksi myös todentaminen tunnetaan tietoturvallisuuden osa-alueena, tarkoittaen käyttäjän tai muun entiteetin luotettavaa tunnistamista tietojärjestelmässä; kuitenkin johtuen sen kiinteästä yhteenkuuluvuudesta luottamuksellisuuden ja kiistämättömyyden kanssa, käsitellään todentamista tässä tutkimuksessa pääasiassa kahden edellisen kautta. (Hakala, Vainio & Vuorinen 2006, 4-6; Benantar 2008, 3-6.)

Klassisen määritelmän mukaiset tietoturvan osa-alueet ovat käsiteltävissä ja ymmärrettävissä hyvin itsenäisinä tekijöinä, mutta kiistämättömyyttä ja pääsynvalvontaa voidaan käsitellä luottamuksellisuuteen ja eheyteen liittyvinä johdannaispiirteinä. Pääsynvalvonta kuvaa enemmänkin menetelmiä ja ratkaisuja, joilla saavutetaan varsinainen kiistämättömyys tietoja käsiteltäessä. Kiistämättömyys taas on eheyttä tukeva tekijä. Kun eheydessä varmistetaan tiedon tahallinen tai tahaton muuttumattomuus ja virheettömyys, kiistämättömyydessä taas tuetaan tiedon eheyttä esimerkiksi pitämällä kirjaa muokkausten yhdistettävyydestä varmasti tunnistettuun käyttäjään (Hakala, Vainio & Vuorinen 2006, 5).

2.1.1 Luottamuksellisuus ja yksityisyys

Yksinkertaisimmillaan yksityisyydellä tarkoitetaan verkkopalvelujen käyttäjien kannalta sitä, että he saavat itse vaikuttaa luovuttamaansa henkilökohtaiseen tietonsa käsittelyyn. Käsittely tarkoittaa miten tätä tietoa kerätään, talletetaan, lähetetään, käytetään tai jaetaan muiden kanssa, milloin, miten ja kuinka mittavasti näin tehdään. (Cannon 2004, 9.)

Verkkopalvelun kannalta tietoturvallinen luottamuksellisuus perustuu siihen kuinka hyvin tämä palvelee käyttäjiensä yksityisyyttä, sekä siihen kuinka hyvin yksityisen tiedon turvallisuus on palvelussa taattu. Käyttäjän yksityisyyden palveleminen perustuu edellä mainitun tavoin siihen kuinka käyttäjä on tietoinen ja vaikutusvaltainen palvelun keräämien tietojen suhteen, ja missä laajuudessa tai millä tavoin palvelu kerää ja käyttää käyttäjän yksityistä tietoa. Verkkopalvelu on vastuussa käyttäjän tietojen turvallisesta käyttämisestä ja säilyttämisestä siten, ettei tieto vuoda tarkoituksettomien ja sitä väärin käyttävien tahojen käsiin tai ettei talletuksessa olevaa tietoa kadoteta tai vääristetä verkkopalvelun sisällä vahingossa tai tahallisesti. (Cannon 2004, 9, 12-13.)

Yksityisyyden käsitteistöstä palvelun käyttäjään voidaan viitata termillä entiteetti. Entiteetti voi olla henkilön lisäksi myös tosimaailman organisaatio tai käyttäjäryhmä. Sovelluksissa ja verkkopalveluissa entiteetillä on identiteetti, jonka pohjalle palveluun tallentuvat tiedot, tapahtumat ja henkilökohtainen sisältö yhdistetään. Palvelukohtaista identiteetin mallinnusta vastaa profiili. Profiilia voi kutsua verkkopalvelussa myös käyttäjätiliksi. Entiteetin tunnistamista sovelluksissa tai verkkopalveluissa kutsutaan todentamiseksi, jonka tehtävä toimenpiteenä on todentaa että tunnistettava entiteetti vastaa todella pyytämäänsä entiteettiä esimerkiksi kirjautuessaan palveluprofiiliinsa. Tässä toimenpiteessä luottamuksellista tietoa sisältävä profiili mahdollistaa tunnistamisen pääsynvalvonnan ja kiistämättömyydenkin puolesta. (Benantar 2006, vii-viii, 1, 3, 10-11.)

2.1.2 Käytettävyys

verkkosovellukselle käytettävyys tarkoittaa sen virheetöntä, riittävän nopeaa toimintaa, joka antaa käyttäjälle tarvittavat, tarkoitetut tiedot riittävän viiveettömästi. verkkosovelluksen toiminnassa tähän kuuluu muun muassa palvelimen ja verkkoyhteyden toimivuus, koodin ja tietokannan optimoitu suorituskyky, sekä käyttökelpoisten tieto- ja tallennusrakenteiden käyttäminen, jotta ne olisivat hyödyllisiä käyttäjälle ja uudelleenkäytettävissä eri sovelluksissa tai sovelluksen osissa tehokkaasti. (Hakala, Vainio & Vuorinen 2006, 4, 336-337.)

2.1.3 Eheys

Eheydellä tarkoitetaan tiedon virheettömyyttä, tarkoituksellista muuttumattomuutta ja paikkansapitävyyttä sekä tallennuksessa että tiedonsiirrossa palvelimen ja tietokannan, sovelluksen tietorakenteiden tai palvelimen ja asiakkaan välillä. Eheyteen voidaan vaikuttaa muutamilla laitteistotason ratkaisuilla, kuten virheenkorjaavilla muisteilla, tai verkkoliikenneratkaisuilla hyödyntämällä virheenkorjaavia tiedonsiirtoprotokollia ja reitityslaitteita. Pääasiassa kuitenkin eheyteen vaikutetaan ohjelmointiteknisesti eli käyttäjäsyötteiden rajoitteilla ja tarkastelulla, tiivisteisiin perustuvilla muuttumattomuustarkastuksilla ja virheenkäsittelytoiminnoilla. (Hakala, Vainio & Vuorinen 2006, 4-5.)

Kun eheyttä halutaan parantaa ohjelmointiteknisillä virheenkorjausmetodeilla, tulee huomioida, että on suorituskyvyllisesti paljon kevyempää laskea muuttumattomuustarkistuksia kuin tehdä automaattista virheenkorjausta. Tästä syystä virheenkorjaavia tarkistusarvoalgoritmeja käytetään lähinnä vain kriittisissä sovelluksissa, ja yleisemmin tulisi nojata verkko- ja laiteinfrastruktuurin suomaan virheenkorjaavuuteen, ja varmuuskopioinnin ja tarkistussummien yhteiskäyttöön. (Hakala, Vainio & Vuorinen 2006, 322-336.)

2.1.4 Kiistämättömyys

Kiistämättömyys on käyttäjien luotettavaa tunnistamista ja heidän tietojensa tallentamista järjestelmässä. Tähän pyritään jotta käyttäjien toimista riippuvaisen tiedon alkuperä voitaisiin tunnistaa luotettavasti, ja jotta voidaan kiistämättömästi osoittaa käyttäjän osallisuus, mikäli tämän katsotaan rikkoneen järjestelmän käyttöehtoja tai toimineen muuten haitallisesti. (Hakala, Vainio & Vuorinen 2006, 5-6.)

Laillisesti sitova kiistämättömyys on hyvin vaikea todistaa, johtuen muun muassa monien entiteetin tunnistavien tietojen, kuten IP-osoitetietojen väärennettävyydestä. Kuitenkin julkisen avaimen salaukseen perustuvat digitaaliset allekirjoitukset voidaan katsoa hyvin varmoiksi, sitoviksi tekijöiksi tarvittaessa myös oikeudelliseen sovellettavuuteen. (Hakala, Vainio & Vuorinen 2006, 86; Benantar 2008, 5.)

2.1.5 Pääsynvalvonta

Pääsynvalvonta käsittää tietojenkäsittelyinfrastruktuurin suojaamista luvattomalta käytöltä. Tavanomaisesti pääsynvalvonta on kiistämätöntä entiteetin tunnistamista ja tälle sallittujen laite-, verkko ja tietoresurssien rajaamista ja rajauksen hallintaa keskitetysti, esimerkiksi rajoittaen kaikkien yrityksen työntekijöiden pääsyä mihin tahansa verkkosivustoille työaikana. Tämä osa-alue on tietoturvan käsitteelle nykyaikana hyvin keskeinen erityisesti monien yhtenevien tietojenkäsittelyjärjestelmien ja langattomien yhteyksien yleistymisen johdosta. (Benantar 2008, 3-4; Hakala, Vainio & Vuorinen 2006, 4-6, 85-86.)

Muut osa-alueet edellyttävät pääsynvalvontaa tietojärjestelmässä (Hakala, Vainio & Vuorinen 2006, 85)

2.1.6 Verkkosovellusten uhkat

Tämän luvun tarkoitus on kuvata uhkia, joilla tässä tutkielmassa tarkoitetaan tietoturvan puutteen tai epäonnistumisen seurauksia. Tätä varten pyritään erittelemään seurauksia, joihin verkkosovelluksen haavoittuvuudet ja erinäiset näitä hyödyntävät hyökkäykset voivat johtaa. STRIDE on alkujaan Microsoftin kehittämä uhkien tai riskien määrittämiseen tarkoitettu listaus ja muistisääntö sekä ohjelmistokehitykseen että riskimääritykseen, joka esittää valmiiksi joukon yksinkertaistettuja uhkia, jotka pätevät hyvin verkkosovelluksiin. (Open Web Application Security Project 2010a; MSDN Magazine November 2006; MSDN Library 2005; Microsoft Corporation 2005.)

STRIDE-nimitys on muistisääntö, joka pohjautuu listattujen riskien englanninkielisiin alkukirjaimiin (taulukko 1). Nämä yleistetyt riskit ovat identiteettihuijaus, tiedon muokkaus, kiistäminen, tietovuoto, palvelunesto ja käyttöoikeuden korottaminen. Tutkimuksen aikana käsitellyt hyökkäykset ja heikkoudet käyvät yhteen STRIDE:n kanssa ja niiden voidaan katsoa johtavan vastaavien riskien toteutumiseen hyökkäysten tai väärinkäytösten onnistuessa.

Alkuperäiskielinen riski	**Riski (käännös)**	**Tietoturva-alue, johon kohdistuu**
Spoofing identity	Identiteettihuijaus	Todentaminen
Tampering with data	Tiedon muokkaus	Eheys
Repudiation	Kiistäminen	Kiistämättömyys
Information disclosure	Tietojen paljastuminen	Luottamuksellisuus
Denial of service	Palvelunesto	Käytettävyys
Elevation of privilege	Oikeuksien korottaminen	Pääsynvalvonta

TAULUKKO 1. STRIDE:n mukaiset tietoturvauhkatja niissä uhatut tietoturvaperiaatteet

Identiteettihuijauksesta seuraa todentamisen ja luotettavan tunnistamisen horjuminen (MSDN Magazine 2006). Jos hyökkääjä pystyy esiintymään toisena käyttäjänä, kaikki muutkin todentamiseen nojaavat tietoturvan osa-alueet vaarantuvat (Benantar 2006, 3-4). Valheellisesti toisena käyttäjän esiintyvä hyökkääjä uhkaa pääsynvalvonnan toimin-

taa sovelluksessa, käytetyn käyttäjäprofiilin sekä muiden tälle avoimien tietojen luottamuksellisuutta ja riippuen sovelluksen toiminnasta hyökkääjälle voi myös olla mahdollista muokata väärän profiilin turvin tietoja, jolloin eheys vaarantuu. Riippuen sovelluksen teknisestä toteutuksesta käyttäjän toimien kirjaamisesta luotettavasti tunnistettuun tekijään, myös kiistämättömyys voi kärsiä, mikäli suoritetut muutokset yhdistetään ainoastaan väärinkäytettyyn käyttäjäprofiiliin.

Tietojen muokkaus eli sovellustietojen muuttaminen ja vääristäminen käyttäjän tahallisin tai tahattomin toimin voi tapahtua lukuisilla eri tavoilla. HTTP:n evästeet, GET-, POST- ja header- ynnä muut kentät voivat tulla hyökkääjän muokkaamiksi ja tulla palautetuiksi väärillä, kelvottomilla tai suoraan vahinkoa aiheuttavilla sisällöillä (Open Web Application Security Project 2010a). Tällöin tallennetun ja siirrettävän tiedon eheys sovelluksessa on uhattuna (MSDN Magazine 2006), ja seurauksena voi olla virheellisen tai vajaan tiedon tallentamista, haitallisten ja sovelluksen toiminnan ulkopuolisten komentojen ajamista muokattujen tietokenttien palatessa sovelluksen käsittelyyn, tai pahimmillaan evästeiden tai istuntojen kaappaus hyökkääjän toimesta, joka johtaa edellä esiteltyyn identiteettihuijaukseen ja sen mahdollisiin seurauksiin.

Sovelluksen sisäisten tapahtumien ja toimintojen kieltäminen on riskinä, kun kiistämättömyyttä ei turvata riittävillä menetelmillä tai kiistämättömyyden ei voida katsoa olevan luotettavaa lailliseen sitovuuteen. Tästä voi seurata sovelluksen luonteesta ja ominaisuuksista riippuen merkittävääkin vahinkoa, kuten kaupallisten sovellusten rahasiirtoja, joiden tapahtuminen kielletään väärinkäyttäjän toimesta ja vaaditaan jo menetettyjen rahojen uudelleen takaisin maksamista käyttäjälle. Mikäli sovellus ei takaa sitovaa kiistämättömyyttä, tällaiset tapaukset jäävät herkästi sovellusta ylläpitävän palvelun tappioiksi. Tästä johtuen lokeihin ja tapahtumahistoriaan tähtäävä kiistämättömyysmenettely on erityisen tärkeää sovelluskehityksessä etenkin verkkosovelluksissa, joissa on tässä luvussa esiteltyjen riskien mukaisesti lukuisia mahdollisuuksia yksittäisten tunnistustekijöiden hämärtymiseen. (Open Web Application Security Project 2010a.)

Tietojen paljastuminen on riski sovelluksen ja sen tietojen luottamuksellisuudelle. Sovelluksesta voi eri tavoin paljastua käyttäjien yksityisiä tietoja tai sovelluksen toiminnalle keskeisiä, luottamuksellisia tietoja. Käyttäjien yksityisyyden menetys tietenkin vahingoittaa jo yksittäistapauksina käyttäjien luottamusta sovellukseen palveluna ja

tämän maineeseen. Sovelluksen toiminnalle keskeisen, luottamuksellisen tiedon julkistuminen suunnittelu- tai sovellusvirheen tai hyökkäyksen seurauksena taas voi mahdollistaa muidenkin riskien toteutumista hyökkääjän käyttäessä saamiaan tietoja uusiin hyökkäyksiin. (Open Web Application Security Project 2010a.)

Palvelunesto on riski, joka koskee nykyään kaikkia verkkosovelluksia. Sovelluksen alustana toimivalla laitteistolla, ohjelmointikielivalinnoilla ja palvelinohjelmistoilla liitännäisineen on kaikilla omat vaikutuksensa tämän riskin todennäköisyyteen, mutta sovelluskehityksessä on myös paljon vaikutusvaraa; suunnitellessa ja toteutettaessa verkkosovellusta tulee kiinnittää huomiota kaikkiin suorituskyvyn pullonkauloihin, jotka sattumanvaraisessa käyttöhuipussa, säännöllisessä rasituksessa tai järjestelmällisen hyökkäyksen johdosta saa sovelluksen pysähtymään rasituksesta. (Open Web Application Security Project 2010a.)

Oikeuksien korottaminen on uhka, jossa tiettyihin resursseihin ja toimiin pääsynvalvonnallisesti rajoitettu käyttäjä saa nostettua omaa käyttöoikeusluokitusta, päästen käsiksi luottamukselliseen tietoon. Verkkosovelluksissa yleisenä pidetty haavoittuvuus on käyttöoikeusrajattujen toimintojen tarjoaminen siten, että vain niihin johtavien linkkien näyttäminen on suojattu, kun todellinen ratkaisu on tarkastaa käyttöoikeudet ja tunnistautuminen jokaisessa luottamuksellisella verkkosivulla erikseen. (Open Web Application Security Project 2010a.)

2.2 Salausmenetelmät

Kryptaus- eli salausmenetelmillä tarkoitetaan menetelmiä selkokielisen sisällön muokkaamiseksi salattuun muotoon. Keskeisimmät käytöt salaustekniikoille ovat tiedon salaaminen kahden eri tahon välillä siten, että tieto hyökkääjän kaappaamana olisi merkityksetöntä ja tiedonsiirrossa hyökkääjän tekemät muutokset voitaisiin tunnistaa ja hylätä epäkelvollisina. Tämä on verkkopalvelujen kannalta olennaista erityisesti salasanoja tai muuta luottamuksellista tunnistautumistietoa käsiteltäessä, sekä asiakkaan ja palvelimen välisen tiedonkulun suojaamisessa.

Selkokielinen teksti pystytään muuttamaan peruuntumattomaksi tiivisteeksi niin kutsutuilla tiivistealgoritmeilla. Tällaista tiivistettä ei tulisi pystyä palauttamaan selkotekstiksi

millään käänteistoimenpiteellä. Käytännössä tämä tarkoittaa, että salausmenetelmiin käytettäviä tiivistefunktioita on hyvin merkittävästi tehottomampi kääntää takaisinpäin, sillä aidosti yksisuuntaisten funktioiden olemassaolosta ei ole todisteita (Buchmann 2004, 237). Lisäksi hyvän tiivisteen tulisi muuttua koko mitaltaan merkittävästi pienistäkin muutoksissa alkuperäistekstiin, jotta tiiviste olisi luotettava eikä sen alkuperäissisältö olisi pääteltävissä. Tiivisteet ovat kiinteän mittaisia, niiden mitta vaihtelee algoritmista riippuen, ja niistä ei voida tunnistaa tai tulkita alkuperäistä, selkotekstistä sisältöä. Eri selkoteksteistä ei saa muodostua samanlaisia tiivisteitä. Tiivisteitä käytetään tyypillisesti tiedostojen tai viestisisältöjen muuttumattomuuden osoittamiseen ja todennettavien, luottamuksellisten tietojen tallentamiseen. Tällainen muuttumattomuuden varmistaminen tukee tiedon eheyttä, ja on ns. yksinkertainen tarkiste – tiivistealgoritmeja ei voida peruuttamattomuutensa vuoksi käyttää virheenkorjaukseen, jossa viestisisällön osoittautuessa muuttuneeksi pitäisi se pystyä korjaamaan tiivisteestä alkuperäistietoa purkamalla. (Schneier 2003, 30-31; Hakala, Vainio & Vuorinen 2006, 325.)

Tiivisteiden lisäksi selkotekstiä voidaan salata salatekstiksi erilaisilla algoritmeilla myös peruutettavasti, jolloin oikealla salausavaimella ne voidaan palauttaa alkuperäiseksi selkotekstiksi (Buchmann 2004, 71-114). Salausmenetelmät jakautuvat useisiin eri lajeihin, riippuen ominaisuuksistaan. Kuvaavat termit salausmenetelmien jaoissa toisiinsa nähden ovat symmetrinen, epäsymmetrinen, yksisuuntainen ja kaksisuuntainen. Symmetriset salausalgoritmit ovat algoritmeja, joissa salausavain on sama sekä salatessa että salausta purkaessa, tai purkuavain on helposti johdettavissa salausavaimesta. Modernit symmetriset algoritmit perustuvat selkotekstin bittien tietokoneistettuun sekoittamiseen. Epäsymmetrisissä salausalgoritmeissa on oma avaimensa molemmille toimenpiteille, eikä niitä voida laskennallisesti suoraan johtaa toisistaan. Epäsymmetriset algoritmit perustuvat matemaattisen laskentaan, joka voi toiminnaltaan olla hyvinkin yksinkertaista, ja siksi laskuissa käytettävien lukujen on oltava hyvin suuria; seurauksena myös salausavaimilta vaaditaan suurempia pituuksia epäsymmetrisissä menetelmissä. (Järvinen 2003, 77, 131-132; Buchmann 2004, 235-242.)

Suola on salasanatiivisteen muodostamisessa käytetty satunnaisjono. Suolat suojaavat samaa salasanaa useissa eri järjestelmissä käyttäviä henkilöitä. Siinä missä salasanojen tallentaminen tiivisteinä selkotekstin sijaan suojaa salasanatietokantaa epärehellisiltä ylläpitäjiltä ja tekee tietokantamurron tapauksessa salasanat lähes käyttökelvottomiksi,

suolan käyttäminen parantaa tallennettujen tiivisteiden turvaa jälkimmäisessä tapauksessa entisestään heikentämällä ratkaisevasti sanakirjahyökkäyksien ja esilaskennan toimivuutta hyökkäysmetodina. Esilaskennallinen sanakirjahyökkäys perustuu kokonaisen mahdollisesti salasanoina käytettyjen sanakirjojen listaamiseen esilaskettuina tiivisteinä. Jos hyökkääjä saa järjestelmän salasanatiivisteet haltuunsa, eikä tiivisteissä käytetä mitään ylimääräisiä suojamenetelmiä, voi hyökkääjä vain verrata sanakirjasta laskemiansa tiivisteitä salasanatiivisteisiin ja siten vertaamalla tunnistaa mitä salasanoja tietokannassa on. Mikäli kaapatussa salasanatietokannassa on käytetty suoloja, kasvaa salasanahyökkäyksen esilaskettu tiivistelista mahdollisten suola-tiiviste -yhdistelmien monikerraksi, ja hyökkäys jää helposti tehottomaksi. (Schneier 1996, 52-53; Järvinen 2006, 245-247.)

Sanakirjahyökkäyksen ohella toinen keskeinen hyökkäys salauksia vastaan on salausavaimen selvittäminen niin kutsutusti raa'alla voimalla, eli kokeilemalla kaikkia mahdollisia yhdistelmiä, kunnes oikea avain löytyy. Tätä vastaan salausmenetelmien tulisi yleisesti ottaen käyttää pidempiä salausavaimia, jotka koostuvat mahdollisimman laajasta valikoimasta erilaisia merkkejä.

Julkisen avaimen salaus, ”public key encryption”, on epäsymmetrinen tekniikka, jossa julkinen ja yksityinen avain ovat keskenään erilaisia (Järvinen 2003, 132). Avaimet ovat kuitenkin keskenään symmetrisiä siten, että ne toimivat molempiin suuntiin, eli kumpikin avain voi salata viestin ja purkaa toisen salaaman viestin. Tätä symmetrisyyttä käytetään hyväksi digitaalisissa allekirjoituksissa, jossa allekirjoittajan yksityinen avain salaa allekirjoitustiivisteen tiedosta, ja allekirjoituksen tarkistaja voi purkaa tiivisteen vertailtavaksi viestisisältöön allekirjoittajan julkisella avaimella. (Järvinen 2003, 154.)

2.3 Verkkopalvelut

Tavanomainen, staattinen verkkosivu on muuttumaton ja itsenäinen HTML- tai XHTML-dokumentti. Tämä tarkoittaa, että sen näyttämä sisältö on aina sama kaikille käyttäjille. Tällainen sivu ei täytä dynaamisuuden määritelmää, johon kuuluu sisällön mukautuvuus käyttäjään, ulkoiseen syötteeseen tai muihin ehtoihin; samalla staattisten sivujen sisällön päivittämisen tehokkuus, muu hallinnointi ja toiminnallisuus saavuttavat vain murto-osan dynaamisen sivun mahdollisuuksista. Staattinen HTML-sivu ei

itsessään riitä täyttämään verkkopalvelun kriteerejä. Jotta voitaisiin oikein puhua verkkopalvelusta, sivuston tulee olla jollain tavoin dynaaminen. Tämä tarkoittaa sivuston sisällön mukautuvuutta HTTP-pyynnön ominaisuuksiin kuten selainversioon tai käyttäjäsyötteeseen (X)HTML-lomakkeiden tai osoiterivin parametrien kautta. Tavanomainen piirre on myös muisti, useimmiten tietokannassa, joka mahdollistaa käyttäjäsisällön, tehtyjen valintojen tai suoritettujen toimintojen tallentamisen lokissa tai historiana joka voidaan palauttaa tarkasteltavaksi tai muokattavaksi myöhemmin. Tietokantapohjaisuus verkkopalveluissa mahdollistaa myös yhä enemmän hyödynnetyn sisällönkäsittelyn, päivittämisen ja ylläpidon palveluun kuuluvien ylläpitotyökalujen kautta. verkkokäyttöliittymälliset ylläpitotyökalut verkkopalvelujen osana helpottavat merkittävästi edellämainittuja toimenpiteitä sisällöstä ja ylläpidosta vastaaville henkilöille. (Ullman 2008, ix-x.)

Koska dynaamisissa sivustoissa palvelun henkilökohtaistaminen, profiilit, toimintahistorian seuraaminen, käyttäjien luoma sisältö, merkinnät ja tilaukset kaikki vaativat jonkinlaista käyttäjätietokantaa, ovat kirjautumistoimenpiteiden suunnittelu ja toteutus tässä työssä keskeisiä. Moni tietoturvasuositus ja ohjelmistokehityksen erityishuomio keskittyy johonkin kirjautumisen tai käyttäjänhallinnan osaan.

2.4 PHP

PHP:n kotisivuilla kieli määritellään seuraavasti: PHP on laajasti käytetty yleiskäyttöinen komentosarjakieli joka on erityisesti soveltuva verkkokehitykseen ja on upotettavissa HTML:n joukkoon. PHP:n julkaisuversio kirjoitushetkellä on 22.7.2010 julkaistu PHP 5.3.3. (The PHP Group 2010c, 2010e.)

Artikkelissaan Lynn Greiner (2008) viittaa Evans Datan tutkimukseen, joka osoittaa PHP:n laajaa markkinaosuutta Internetin dynaamisesta sisällöstä (Greiner 2008). Myös SecuritySpacen (2009) selvitys määritti PHP:n kolmanneksi Apache HTTP-palvelimen käytetyimmistä moduuleista (Apache Survey: Apache Module Report 2009).

PHP on alustariippumaton ja palvelimen puolella toimiva teknologia. Tällä viitataan PHP:n toimintaperiaatteeseen, jossa PHP-tulkki toimii HTTP-palvelimen yhteydessä palvelintietokoneella. Kun PHP vastaanottaa HTTP-sivupyynnön, palvelin ajaa PHP-

-tulkin lävitse pyynnön kohteeseen sisältyvän PHP-tiedoston dynaamisen sisällön, ja palauttaa tästä syntyvän HTTP-vastauksen HTML- tai XHTML-sisällöllä. PHP tukee olio-ohjelmointimenetelmiä 13.7.2004 julkaistusta PHP 5-versiosta asti, jolloin sen toiminnalle keskeinen komentosarjamoottori Zend Engine päivittyi toiseen versioonsa, Zend Engine 2:een (Zend Technologies 2010; The PHP Group 2010). (Ullman 2008, x.)

Ullman (2008) väittää koskien PHP:n toimintaa, että kaikki PHP-sisältö on tavoitettavissa ainoastaan URL:ien ja siten HTTP-palvelimen kautta, joka ajaa PHP-tulkkia (Ullman 2008, x). Tämä ei kuitenkaan pidä paikkaansa, sillä PHP-koodia sisältävän tiedoston voi ajaa myös suoraan komentoriviltä PHP-tulkin kautta esimerkiksi testaustarkoituksessa.(Fuecks 2004; The PHP Group 2010Q).

Komentosarjakielen ja varsinaisen ohjelmointikielen välinen rajanveto on häilyvä. komentosarjakieli on usein jonkin toisen ohjelman hallintaan käytettävä yksinkertainen ohjelmointikieli, jota ajetaan tulkitsevan ohjelman eli tulkin kautta joko suoraan lähdekoodista tai tavukoodiksi käännettynä. Ohjelmatulkki itse on yleensä käännettynä ajoympäristöriippuvaiseen, konekieliseen muotoon. Yleiskäyttöisellä komentosarjakieleksi kutsutaan usein esimerkiksi Perliä muistuttavaa tai sen tavoin komentosarjakielestä ohjelmointikieleksi kehitettyä kieltä, vaikka näitä käytettäisiinkin enemmän sovelluskehitykseen kuin toisten sovellusten tehtäviä ohjaaviin skripteihin.

2.5 MySQL

MySQL on Oracle Corporationin omistama RDBMS eli "Relational Database Management System", relaationaalinen tietokantahallintajärjestelmä. Relaationaalinen tietokanta tallettaa tietoa useissa tietokantatauluissa, joiden välillä voi olla relaationaalisia yhteyksiä keskenään. Sen etuina ovat luotettavuus, parempi haettavuus ja tuki samanaikaisille käyttäjille. (Ullman 2008, xiv.)

MySQL on maailman suosituin avoimen lähdekoodin RDBMS (Oracle Corporation, 2010a). Sen lisensointivaihtoehdot ovat kahdenlaiset, riippuen käyttötavasta – MySQL ja sen asiakaskirjastot ovat vapaasti käytettävissä GPL 2.0 -lisenssin mukaisessa tuotteessa, mutta mikäli kaupallinen käyttäjä ei halua lisensoida MySQL:n sisällyttävää tuotetta tai palveluaan GPL:n alaisuuteen, myy Oracle Corporation myös kaupallisia li-

senssejä MySQL:n käyttöön (Oracle Corporation 2010c). Poikkeuksena tähän on pelkkien MySQL-asiakaskirjastojen käyttö, jolloin on mahdollista levittää tuotettua sovellusta myös Oracle Corporationin määrittämän FOSS-poikkeussäännön ("Free and Open Source Software") mukaisesti minkä tahansa FOSS-lisenssin mukaisesti (Oracle Corporation 2010d).

MySQL-tuoteperheeseen kuuluu MySQL-palvelimen lisäksi Enterprise-tason yrityspaketti ohjelmistoineen ja teknisine tukineen, graafinen hallintatyökaluohjelmisto MySQL Workbench, sekä hajautettu tietokantaratkaisu MySQL Cluster (Oracle Corporation 2010e). MySQL:n nykyinen versio on kirjoittamisen hetkellä tuotantoversio 5.1, MySQL 5.5 ja 5.6 ollessa kehitysversioita (Oracle Corporation 2010b, 2010k).

MySQL:n lähdekoodin avoimuus on edistänyt monien ohjelmistojen hyvää MySQL-tukea. Esimerkkinä PHP 5-versiossa paranneltu laajennus MySQL Improved Extensionista, PHP-rajapinnasta joka tukee aiempaa rajapintaa monipuolisemmin MySQL:n modernimpia ominaisuuksia. (Oracle Corporation 2010a, 2010c; The PHP Group 2010a; Ullman 2008, xv.)

MySQL:n nimessä esiintyvä SQL (”Structured Query Language”) puolestaan on tietyistä avainsanoista, funktioista ja syntaksista koostuva kieli, jonka avulla voidaan tehdä monipuolisia hakuja, lisäyksiä ja muokkauksia relaatiotietokannoissa olevaan tietoon. MySQL-palvelimella toimivaa tietokantaa käsitellään mukautetulla SQL-standardilla. (Oracle Corporation 2010g; Ullman 2008, 123.)

3. TUTKIMUSKYSYMYS

Tässä luvussa esitellään tutkimuskysymys, johon tutkielmassa haetaan vastausta tai vastauksia; haettaessa parempaa tietoturvaa verkkosovelluskehityksessä, kysytään keskeinen tutkimuskysymys: miten kehittää verkkosovelluksia tietoturvallisesti? Taustoissa on jo valmiiksi alustettu tätä kysymystä perehtymällä tietoturvan käsitteeseen, osa-alueisiin ja uhkiin. Osa-alueet ovatkin eräänlaisia tietoturvan itseisarvoisia periaatteita, uhkat näiden periaatteiden puutetta, vahinkoa sovellukselle ja sen käyttäjille – itse tutkimuskysymyksessä perehdytään siihen millä tavoin nämä uhkat voivat toteutua verkkosovelluksissa, ja miten niihin voidaan vaikuttaa suojellakseen haluttuja tietoturvan osa-alueita kehitystyössä vähentämällä uhkien toteutumisen mahdollisuutta tai vaikutusta.

Tämän kysymyksen tarkoitus on esitellä verkkosovellusten hyviä tietoturvakäytäntöjä, haavoittuvuuksia ja vastatoimenpiteitä haavoittuvuuksien torjuntaan ja hallintaan. Täten tämän kysymyksen vastaukset perustuvat taustoissa annettuun tietoturvan määritykseen, ja tarjoavat toivottavasti käytännöllisimmin teoriaa ja ohjeita verkkosovelluksen tietoturvan suunnitteluun ja toteutukseen. Kysymykseen vastataan luvussa 5. ja sen alaluvuissa 5.1-5.8.

4. TUTKIMUSMENETTELY

Työn tavoitteena on paikallistaa ja oppia keskitetysti ratkomaan verkkopalveluiden tietoturvaan liittyviä ongelmakohtia kokonaisuutena. Työmenetelmänä on yksinkertaisesti kirjakatsaus ja harkittujen nettilähteiden hyväksikäyttäminen sen tueksi. Tutkielman vastauksissa ja esimerkeissä viitataan oletusarvoisesti PHP- ja MySQL-pohjaisiin ratkaisuihin perustuen näiden tekniikoiden taustoissa esiteltyyn tunnettuuteen ja helppoon käytettävyyteen, henkilökohtaiseen osaamispohjaan, sekä oletukseen, jonka mukaan löydettyjen ratkaisujen teoriaa voidaan soveltaa myös muihin tekniikoihin perustuvissa kokonaisuuksissa. Oletuksen perusteluna on verkkosovellusten yleistetty toimintamalli, jonka mukaan verkkosovellusten dynaamisuudesta vastaavat tekniikat tuottavat toiminnastaan riippumatta lopulta samankaltaisia (X)HTML-dokumentteja, ja se, että yleiset verkkosovellushaavoittuvuudet pohjaavat enemmän yleiseen sovelluslogiikkaan, (X)HTML:n ja HTTP-protokollan toimintaan, kuin itse dokumentteja tuottavaan tekniikkaan.

Aikaisempaa kokonaisvaltaista tutkimusta on huomattavasti vaikeampi löytää kuin erityisesti tapauskohtaisia suunnittelu- ja toteutustöitä. Useimmat aikaisemmat tietoturvaa käsittelevät opinnäytetyöt myös keskittyvät erityisesti tietoverkko-, laitehierarkia ja palomuuri- sekä pääsynhallintaratkaisuihin. Tutkielmassa käsitellään ensisijaisesti lähdekirjallisuudesta (Schneier 1996; Cronkhite & McCullough 2001; Buchmann 2004; Cannon 2004; Alshanetsky 2005, 73-85; Benantar 2006; Hakala, Vainio & Vuorinen 2006; Ullman 2008) löytyneitä haavoittuvuuksia ja ohjeita. Tutkielman kirjalähteet jakautuvat seuraavasti: tietoturvallisuuden taustojen ja olemuksen selvittämisen kannalta tärkeät lähteet, salausmenetelmien teoriasta ja käytännön sovellutuksista myös laajemmin tietoa sisältävät lähteet (Scheiner 1991; Järvinen 2004; Buchmann 2006) ja täydentävät verkkokehitystä koskevat lähteet (Cronkhite & McCullough 2001; Ullman 2008). Tapauskohtaisesti ja tarkemmin suositusten aiheisiin ja haavoittuvuuksiin on edellä mainittujen ohella haettu aineistoa paljolti (Alshanetsky 2005, 73-85) verkkolähteistä ja tutkimusjulkaisuista (Fu, Sit, Smith. & Feamster 2001; Liu, Huang. & Gouda 2005; Stevens, Lenstra & de Weger 2007; Sotirov, Stevens, Appelbaum, Lenstra, Molnar, Osvik & de Weger 2008). OWASP:n tuore TOP 10 -listaus (Open Web Application Security Project 2010c) kuvassa 1 osoittaa kuitenkin alan olevan jatkuvan muutoksen alaisena, johtuen teknisen kehityksen myötä myös uusien haavoittuvuuksien löytymises-

tä. Uusien haavoittuuvuuksien lisäksi tunnettujen haavoittuvuuksien globaalit painoarvot muuttuvat (Open Web Application Security Project 2007; 2010c) toisiinsa nähden riippuen siitä kuinka ne yleisesti ottaen verkkosovelluskehityksessä huomioidaan eli kuinka suurta vahinkoa ne voivat saada aikaan ja kuinka laajalti esiintyviä ne ovat.

5. TIETOTURVAA VERKKOSOVELLUSKEHITYKSEEN

Tässä luvussa pyritään esittelemään verkkosovellusten tietoturvaa, sitä koskevia haavoittuvuuksia ja ratkaisuja. Alaluvut 5.1-5.8 pyrkivät tällä tavoin vastaamaan luvussa 2. esitettyyn tutkimuskysymykseen käsitellen tunnettuja haavoittuvuuksia ja verkkosovelluskehityksen hyviä käytäntöjä mahdollisuuksien mukaisessa laajuudessa. Haavoittuvuuksia on kerätty useista tutkimusmenetelmissäkin mainituista lähteistä, joista kuvaavimpana lienee Open Web Application Security Projectin (2010c) julkaisema Top 10 -listaus (kuva 1) verkkosovellusten nyt vakavimpina pidetyistä haavoittuvuuksista; tutkielmaan on pyritty sisällyttämään käsittelyä kaikista tärkeimmistä tietoturvan periaatteita koskevista haavoittuvuuksista, joista on löydetty tietoa kattavasti useammista lähteistä – täten poissulkien listan kohdat A4 ja A9.

OWASP Top 10 – 2010 (New)
A1 – Injection
A2 – Cross-Site Scripting (XSS)
A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object References
A5 – Cross-Site Request Forgery (CSRF)
A6 – Security Misconfiguration (NEW)
A7 – Insecure Cryptographic Storage
A8 – Failure to Restrict URL Access
A9 – Insufficient Transport Layer Protection
A10 – Unvalidated Redirects and Forwards (NEW)

KUVA 1. OWASP:n verkkosovellusten 10 pahimpana pidettyä haavoittuvuutta vuonna 2010 (Open Web Application Security Project 2010c).

PHP:n tai muun vastaavan dynaamisesta sisällöstä vastaavan palvelinpuolen ohjelmointikielen kanssa työskennellessä tulee huomioida palvelimelle asennetun PHP:n, tai muun vastaavan komentojonotulkkiohjelmiston, versio jo kehitystyössä. Verkkosovelluksen ja käytettyjen tekniikoiden asetusten vaikuttaessa PHP:n käytettävissä oleviin ominaisuuksiin ja turvallisuuteen, myös uudempien versioiden paremmuus tietoturvan kannalta on aina huomioitava. Komentotulkin uudemmissa versioissa on korjattu suorituskykyyn ja vakauteen liittyviä virheitä, jotka parantavat palvelun keskimääräistä saatavuutta ja siten käytettävyyttä (The PHP Group 2010i). Tämän lisäksi erityisen tärkeää on huomata, että korjauksiin sisältyy myös muistivuotojen ja muiden tietoturvaongelmien korjauksia, ja näin ollen kehitysalustaksi tulee aina suosia tuoreinta versiota niin PHP:sta kuin muistakin ohjelmistoista ja tekniikoista. Joskus kuitenkin on projektista nousevien vaatimusten mukaista käyttää vanhempaa PHP:n versiota, jotta sovellus olisi yhteensopiva muiden sen kanssa toimivien sovellusten kanssa. Vanhemman PHP-version koodisyntaksilla ja komennoilla kirjoitetut verkkosovellukset eivät välttämättä ole yhteensopivia tietoturvallisempien, päivitettyjen PHP-komentotulkkien kanssa – tällaisissa tilanteissa tulee huomioida PHP:n dokumentaation muutto-ohjeistukset versioiden välillä, esimerkkinä kirjoittamishetkellä tuorein 5.2-kehityshaarasta 5.3-versioihin päivittämisen ohjeistus (The PHP Group 2010f).

Alunperin heikosta suunnittelusta seuraa vääjäämättä vältettävissä olevia riskejä. Verkkopalvelun (X)HTML-lomakkeisiin sisällytettyihin piilokenttiin voi tarvittaessa sisällyttää ohjaustietoja lomakkeen toiminnalle. Sovelluksen turvallisuuden kannalta luottamuksellisen tiedon syöttäminen lomakkeen piilokenttiin tai muuallekaan (X)HTML-tulosteen joukkoon on kuitenkin tietoturvariski, sillä (X)HTML-muotoinen verkkosivun lähdekoodi on kenen tahansa luettavissa. Näin ollen palvelun ominaisuuksia kannattaa suunnitella tarkoin, ja karsia salattavien tietojen välittämistä näkyvän lähdekoodin puolella. PHP:n lähdekoodi itsessään on suojattu ulkoiselta tarkastelulta, sillä kaikki pyynnöt PHP-tiedostoihin palvelimella ajetaan PHP-tulkin läpi, joka palauttaa ainoastaan PHP-komentosarjan ulostulon HTTP-vastauksen mukana.

Keskeisimmät PHP:n mahdollistamat hyökkäyskohdat verkkopalveluissa koostuvat asiakkaan puolesta annettuihin syötteisiin, sekä tiedonvälitykseen asiakkaan ja palvelimen välillä. Tämä näkyy myös OWASP:n (Open Web Application Security Project 2010c) listasta (kuva 1), jossa vakavin haavoittuvuus on myös myöhempänä käsitellyt SQL- ja XSS-injektiot käsittävä injektion yleiskäsite. Syötteiden välittäminen asiakkaal-

ta palvelimelle on mahdollista lomakkeiden kautta GET- ja POST-tietokentissä. Tämän lisäksi käyttäjän valintoja ja käyttäjäkohtaisesti mukautettua palvelutilannetta voidaan pitää tallennettuna evästeisiin tai istuntoihin. Evästeitä ja istuntoja käytetään yleisesti tiettyjen muuttujien, kuten käyttäjän kirjautumistilanteen, muistissa pitämiseksi sivuston sivujen välillä siirtymisten välillä. Myös istunnot hyödyntävät asiakaspuolella talletettavaa evästettä, joka sisältää pelkästään sessiotunnisteen, joka osoittaa palvelimen puolessa tallennetun, asiakassessiokohtaisen sessiosisällön.

Verkkosovelluksen kehityksessä on tärkeää ymmärtää GET- ja POST-metodien erot. W3C:n HTML 4.01 -määritelmää (1999) koskevan suosituksen (luettu 2010) mukaan URL:ien osana näkyvien GET-metodin mukaisten tietojen käyttö tulisi rajoittaa ns. idempotentteihin sovellutuksiin. Tällä tarkoitetaan pääasiassa erilaisia sisältöhakuja ynnä muita sovellutuksia, joissa metodin nimen mukaisesti haetaan tietoa, ilman että palvelimen puolessa tapahtuu mitään pysyviä muutoksia. Lisäyksiä, muokkauksia tai poistoja sovelluksen tietokantaan tekevät tiedot tulee välittää POST-metodilla. GET-metodin väärä käyttö lomakesyötteissä voi johtaa CSRF-hyökkäyksiin, joissa käyttäjä avaa dokumentin, jonka sisässä käyttäjältä huomaamatta ladataan linkki myös toiseen verkkosovellukseen. Tässä mainittu toinen verkkosovellus on altis CSRF-hyökkäykselle, mikäli se käyttää GET-metodia ei-idempotenteissa lomakekäsittelyissä ja huijausdokumentin avannut käyttäjä on mahdollisesti tunnistautunut ja yhä voimassaolevassa istunnossa alttiin sovelluksen kanssa. CSRF-hyökkäyksen uhriksi jäävässä sovelluksessa voidaan siis suorittaa toimia huijatun käyttäjän nimissä, rikkoen kiistämättömyyden ja todentamisen periaatteita. (Open Web Application Security Project 2010b; World Wide Web Consortium 2010a.)

5.1 Syötteenkäsittely ja URL-osoitteet

Syötteenkäsittely ("input sanitization") on ehdottoman tärkeää kaikilla dynaamisilla verkkosivuilla, jotka vastaanottavat ja käsittelevät POST- ja GET-dataa, ja tekevät tähän perustuen tietokantahakuja tai muodostavat tämän pohjalta osia jonkin toisen sivun tulosteesta. XSS- ja SQL-injektiohyökkäykset ovat keskeisimpiä uhkia, jotka kohdistuvat ulkoisen syötteen haavoittuvaisuuksiin. XSS- eli Cross-Site Scripting -hyökkäys perustuu asiakaspuolella tulkittavan komentojonon, kuten JavaScript-komentojen, syöttämiseen verkkopalvelulle siten, että verkkopalvelu tulostaa syötteen muuttumatto-

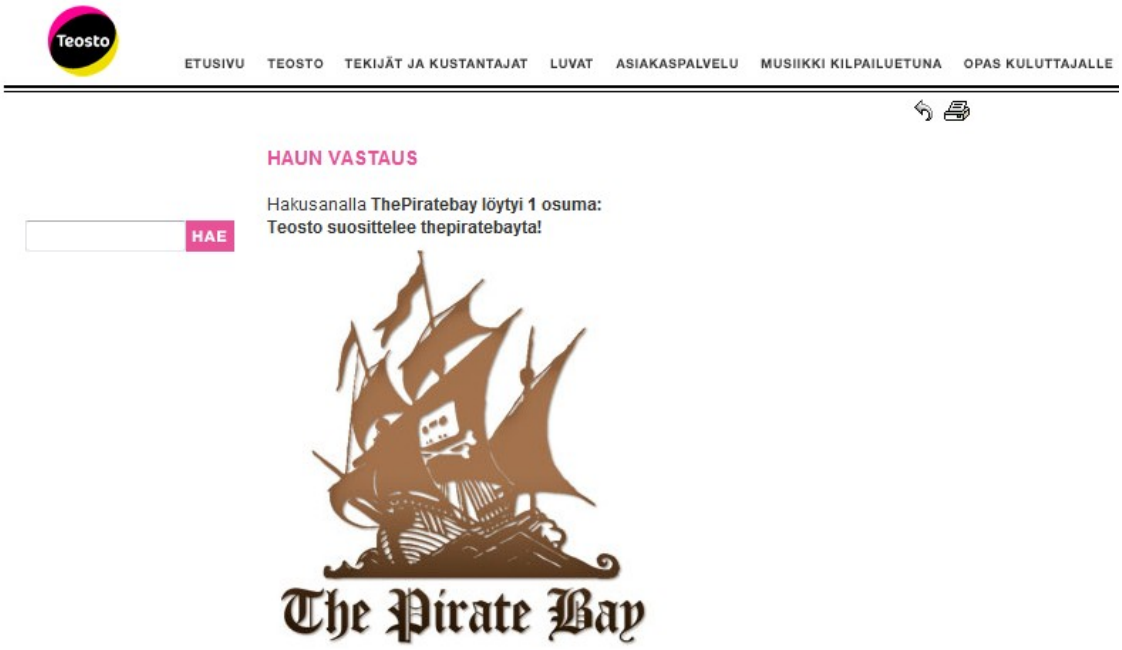
mana. Tällöin haittakoodi eli syötteeseen ujutettu komentojono ajetaan asiakaspuolella selaimessa, josta taas voi seurata useita tietoturvariskejä evästeiden kaappaamisesta sivuston toiminnan tai ulkoasun sotkemiseen. (Ullman 2008, 374.)

XSS-injektiohyökkäykset pyrkivät jättämään käyttäjäsyötteeseen upotettuja komentosarjoja, kuten <script></script>-tagien sisällä olevia JavaScript-ohjelmia. Myös monet <link />-tagit voivat ladata haittakoodia tai pelkät -linkit voivat ohjata käyttäjiä palvelun sisältä hyökkäyssivustoille. Täten kaikki käyttäjäsyötteistä muodostettavat, sivujen (X)HTML:ksi tulostettavat syötteet tulee käydä läpi tapauskohtaisesti riippuen toivotun syötteen tyypistä ja käyttötarkoituksesta. Olennaista tällöin on karsia tai muuten suojata syötteessä esiintyvät ohjausmerkit – käsiteltävät ohjausmerkit riippuvat siitä mihin syötteen on mahdollista päätyä: verkkosivujen osaksi missään vaiheessa tulostettavat syötteet vaativat kaikkien asiakaspuolen eli selainpohjaisten <script>-upotteiden ja (X)HTML-elementtien käsittelyn, tietokantakutsuihin käytettävistä syötteistä on ehdottomasti karsittava käytetyn tietokannan käsittelemät ohjausmerkit tietokantariippuvaisesti. Tietokantasyötteiden käsittelystä kerrotaan tarkemmin myöhemmässä luvussa. (Ullman 2008, 374-376.)

Ohessa on kuvakaappauksilla esitetty esimerkki XSS-injektiohyökkäyskoodista (kuva 2) ja sen vaikutuksesta Teoston verkkosivuilla marraskuulta (kuva 3), ja Teoston verkkosovelluksen XSS-haavoittuvuuden korjauksen jälkeen joulukuulta 2010 (kuva 4). Kyseinen hyökkäys oli enemmänkin pilkkaisku sekä Teoston vastustamaa piraattitoimintaa näiden sivustolla esittäen että näpäytys verkkosovelluksen kehittäjien huolimattomuudelle, osoituksena paikkaamattomasta haavoittuvuudesta. Samalla periaatteella toimivassa hyökkäyksessä voitaisiin eri kohteessa käyttää huomattavasti vahingollisempaa komentojonoa esimerkiksi salattujen evästeiden lukemiseen – josta lisää luvuissa 5.3 ja 5.5.

```
http://www.teosto.fi/teosto/websivut.nsf/SearchShowDocuments?SearchView&
Query=ThePiratebay%20l%F6ytyi%201%20osuma:%3Cbr%20/%3ETeosto%20suosittelee%20thepiratebayta!
%3Cbr%20/%3E%3Ca%20href=%22http://www.thepiratebay.org%22%3E%3Cimg%20border=%220%22
%20src=%22http://static.thepiratebay.org/img/tpb.jpg%22/%3E%3C/a%3E%3C!--&SearchFuzzy=1&lng=Suomi
```

KUVA 2. Marraskuussa netinkäyttäjien keskuudessa esitelty, keskusteltu ja kierrätetty XSS-injektio URL:n GET-parametriin syötettynä. GET-parametri "Query" on saanut arvokseen hakulomakkeeseen syötetyn injektiokoodin (Teosto r.y. 2010)



KUVA 3. Hakulomakkeessa käytetty hakulause tulostetaan osana hakutuloksia, jolloin käsittelemätön injektiokoodi suoritetaan kuin se olisi tarkoituksenmukainen osa sivustoa (Teosto r.y. 2010)

KUVA 4. Joulukuussa 9.12. sivuston ylläpito oli jo korjannut sovelluksen, eikä XSS-injektiohyökkäys mennyt enää läpi käsittelemättömänä (Teosto r.y. 2010)

PHP:ssa on valmiiksi hyviä funktioita tekstisyötteen käsittelyyn, kuten esimerkiksi XSS-injektion torjumiseksi käytännöllinen htmlspecialchars(), tai monikäyttöinen, suodatinparametrien mukaisia suodatuksia tekevä filter_var(). Funktio urlencode() taas on hyödyllinen työkalu kun halutaan varmistaa dynaamisesti muodostettujen linkkien toimivuus URL-osoitteiden merkistörajoitteista huolimatta. Käytettäessä urlencode()-funktiota (X)HTML-entiteetit voivat kuitenkin mennä yhä sekaisin samankaltaisesti nimettyjen muuttujien kanssa, varoitetaan PHP:n dokumentaatiossa (urlencode 2010). Samassa dokumentaatiossa suositellaan käyttämään &-entiteettimerkintää (kuva 5) pelkän &-merkin, eli ampersandin sijasta, kun muodostetaan dynaamisesti sivulle tulostettavien linkkien URL-osoitteita, ja käsitellä ne urlencode()-funktion sijaan

htmlspecialchars()-funktiolla tai vastaavalla filter_var()-funktion suodattimella, FILTER_SANITIZE_SPECIAL_CHARS. &-merkin suora käyttö tulostettavien linkkien URL-osoitteissa rikkoo (X)HTML:n validiutta HTML4 ja myöhemmissä standardeissa (kuva 6). Tämän vuoksi World Wide Web Consortium (2010b), suosittelee käsittelemään ampersandit URL-osoitteissa joko edelläkin mainittuun muotoon $amp; tai $#38. (Ullman 2008, 374; The PHP Group 2010p.)

```
Väärin <a href="http://host/?x=1&y=2">linkkiteksti</a>
Oikein <a href="http://host/?x=1&#38;y=2">linkkiteksti</a>
Oikein <a href="http://host/?x=1&amp;y=2">linkkiteksti</a>
```

KUVA 5. XHTML 1.0 -esimerkki &-merkin virheellisestä ja oikeasta käytöstä linkki-URI:ssa

Validation Output: 2 Errors

- Line 13, Column 27: **cannot generate system identifier for general entity "y"**
- Line 13, Column 27: **general entity "y" not defined and no default entity**
- Line 13, Column 28: **reference not terminated by REFC delimiter**
- Line 13, Column 28: **reference to external entity in attribute value**
- Line 13, Column 28: **reference to entity "y" for which no system identifier could be generated**
- Line 13, Column 26: entity was defined here

KUVA 6. Tiivistetty validator.w3.org:n XHTML 1.0 Strict -palaute väärästä linkki-URI-käytöstä &-merkille

Silloinkin kun käyttäjältä saatu lomakesyöte ei suoraan ohjaa dynaamisia SQL-hakuja, voi syöte yhä olla väärää tyyppiä, kirjoitettu tai muodostettu väärin riippuen lomakkeen tarkoituksesta ja lomaketiedon käytöstä ohjelmakoodissa. Mikäli PHP:n puolella odotetaan numeroarvoja joilla pyritään tekemään laskutoimituksia tai käyttämään näitä erikseen muissa toiminnoissa, voi tekstiarvon syöttäminen tahallaan tai vahingossa lomakkeeseen keskeyttää PHP-komentosarjan ajamisen ja dynaamisen sivusisällön palautuksen palvelimelta. Riippuen verkkopalveluohjelman rakenteesta, koko komentosarja voi suorittua loppuun arvaamattomilla lopputuloksilla, keskeytyä ja palauttaa vain aiemman osan (X)HTML-sivusta jota käsiteltiin, jättää sivun kokonaan palauttamatta; mikäli kehittäjille tarkoitettu debug-tila ja virhetulosteasetus on jäänyt päälle PHP:n asetuksista sovellus voi tällöin myös näyttää PHP:n virheilmoituksia käyttäjille. Näin ollen lähes kaikki käyttäjäsyöte lomakkeista tai URL:n mukaisista GET-parametreista muodostaa riskejä verkkopalvelun käytettävyydelle, tietokantatiedon eheydelle ja luotta-

muksellisuudelle, mikäli hyökkääjä pystyy saamaan sovelluksen toiminnasta tarkempia tietoja huolimattoman virheraportoinnin ja tahallisesti aiheutettujen virheiden vuoksi. Virheraportoinnin asetuksista ja virheidenkäsittelyn tietoturvasta omassa luvussaan 5.8.

GET-kentässä välitettyihin tietoihin pääsee kuka tahansa käsiksi asiakaspuolessa lukemalla ja muokkaamalla selaimen osoiteriviä ja siinä näkyviä GET-parametreja, jotka näkyvät osana URL:a. GET-parametrijono erotetaan URL:sta kysymysmerkillä (?) ja jonon sisällä eri parametrit toisistaan ampersandilla (&). Parametrit koostuvat avain-arvoparista, jossa avain toimii nimitunnisteena arvon osoittamalle tiedolle. Myös POST-metodilla välitettyyn tietoon on mahdollista päästä käsiksi erinäisillä työkaluilla. Esimerkkinä tutkimus- ja verkkokehityskäyttöön tarkoitettu Firefox-liitännäinen Tamper Data (Tamper Data 2010). Syötteiden käsittely näiden kenttien sisällön puolesta koskee enimmäkseen suoraan asiakaspuolella tehtävää haitantekoa, erotuksena myöhempien alalukujen käsittelemiin yhteyden suojaamisen ongelmiin.

Verkkopalveluja uhkaavat tahot voivat erinäisillä ratkaisuilla (Tamper Data 2010; Live HTTP Headers 2010) muokata HTTP-pyyntöä, joka sisältää myös POST-kentässä välitetyt parametrit. Hyökkääjä voi muokata HTTP-pyyntöä mielensä mukaan ja välittää haitallisella datalla varustetun pyynnön palvelimelle. Haitallista dataa voi olla mikä tahansa ohjelman normaalia toimintaa sotkeva tai PHP:n kautta muodostettuun, dynaamiseen SQL-lauseeseen päätyvä ja sen toimintaa estävä tai muokkaava data.

5.2 SQL-injektiot

SQL-injektio on syötteenkäsittelyn erityisongelma, joka koskee lomakkeita, joilla kerätään tietoa mukautettuihin SQL-lauseisiin. Tällaisia lomakekenttiä ovat esimerkiksi erilaiset hakukentät ja kirjautumislomakkeet, joihin syötetty tieto liitetään osaksi ohjelmakoodissa etukäteen muotoiltua SQL-lauserunkoa. SQL-injektiohyökkäys perustuu aina lomakesyötteen puutteelliseen tai olemattomaan käsittelyyn verkkopalvelussa. Kun lomakesyötettä ei tarkasteta asianmukaisesti, voi hyökkääjä lähettää SQL-koodia lomakesyötteen seassa. Tämä koodi päätyy osaksi sovelluksen suorittamaa SQL-lauserunkoa ja ajetaan tietokannassa mikäli kokonaislauseen syntaksi on yhä eheä. Oikeanlaisilla SQL-injektioilla hyökkääjä voi vaiheittain päätellä sovelluksen tietokantarakenteen sekä ajaa omia komentojaan tämän tietämyksen pohjalta. Pahimmassa tapauksessa SQL-in-

jektio voi esimerkiksi ajaa MSSQL Server -tietokantapalvelimesta löytyviä käyttöliittymäkomentoja SQL-lauseita palvelimen toimiessa riittävillä käyttöoikeuksilla ja vaikuttaa siten suoraan tietokannan alla toimivaan käyttöjärjestelmään. (Alshanetsky 2005, 73-74; The PHP Group 2010p.)

Esimerkiksi kirjautumislomakkeen tapauksessa PHP:iin ohjelmoitu kirjautumistoiminto odottaisi todennäköisesti kirjautumislomakkeesta esimerkiksi käyttäjänimeä ja salasanaa erillisinä parametreinaan. Yksinkertainen kirjautumistarkastuksen SQL-lause voisi olla seuraavanlainen:

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

PHP lähettää SQL-lauseen MySQL-palvelimelle, ja odottaa vastaukseksi käyttäjätietorivin palautusta hakua vastaavalta käyttäjältä; mikäli tietokannasta löytyy näitä tietoja vastaava käyttäjä, tunnistautuminen on onnistunut. Tyhjä palautus tarkoittaa ettei annettuja ehtoja vastaavaa käyttäjää löydy users-taulusta, jolloin jompikumpi hakuehdoista, käyttäjänimi ("admin") tai salasana ("salasana"), on annettu väärin syötteessä.
SQL-injektiossa, hyökkääjä etsisi tämänkaltaista kirjautumisratkaisua, ja syöttäisi kirjautumislomakkeeseen dataa, joka sisältää SQL:n ohjausmerkkejä. (Alshanetsky 2005, 73.)

```
admin' AND 1 = 1; --
```

Ylläoleva esimerkkidata syötettynä kirjautumislomakkeen käyttäjätunnukseksi muodostaisi osan aiemman esimerkin SQL-lausetta palautuessaan lomakkeesta PHP-käsittelyyn. Tällöin muodostuva SQL-hakulauseke näyttäisi seuraavalta:

```
SELECT * FROM users
WHERE username = 'admin' AND 1 = 1; --'
AND password = 'salasana'
```

Tässä SQL-injektiohyökkäyksessä käytetyt ratkaisevat ohjausmerkit ovat "admin"-käyttäjätunnuksen perään lisätty yksinkertainen lainausmerkki sekä lopussa oleva puolipiste

";" ja kommenttimerkintä "--". Toinen MySQL:n hyväksymä SQL-kommentaatiomerkintä on "#". Edellä oleva esimerkki toteuttaa SQL-injektioiden keskeisintä mallia, jossa lomakkeesta tuleva syöte esittää yhtä odotetuista parametreista, mutta sisältää SQL--ehtolauseen vertailtavaa stringmuuttujaa sulkevan lainausmerkin. Tätä seuraa uusi ehtolause AND- tai OR-koodisanalla, jonka jälkeen voidaan esittää uusi vertailulause ja saada kokonaisvertailu joko onnistumaan tai epäonnistumaan hallitusti. SQL-lauseen katkaisevan puolipisteen sekä kommenttimerkinnän syötteen loppuun sisällyttäminen mahdollistaa alkuperäisen SQL-lauseen loppuosan hylkäämisen tai uuden SQL-lauseen syöttämisen aikaisemman perään. Tällä tavalla voidaan suorittaa muokatun alkuperäislauseen lisäksi kokonaan itse rakennettuja SQL-muokkauksia tai -poistoja, koska SQL-injektioita käyttämällä voidaan päätellä yksittäisen tietokantataulun tai koko tietokannan rakenne haavoittuvissa sovelluksissa. (Alshanetsky 2005, 79-80.)

Verkkosovelluksen haavoittuvuus SQL-injektiolle määrittyy tietokantaratkaisun lisäksi pitkälti myös ohjelmakoodin käyttämästä tietokantarajapinnasta. Esimerkiksi PHP:n mysql_query()-funktio, joka lähettää SQL-komentolauseen yhdistetylle MySQL-palvelimelle, ei hyväksy peräkkäisiä SQL-lauseita, ja on siten teoriassa immuuni puolipisteeseen perustuviin, haittakoodilla katkaistua syötettä jatkaville hyökkäyksille. Saman funktion kehittyneempi seuraaja kuitenkin mahdollistaa samassa tekstiparametrissa useampien SQL-lauseiden syöttämisen MySQL-palvelimelle mahdollisessa optimointitarkoituksessa, mutta altistaa sovelluksen tälle tavanomaiselle SQL-injektiolle. Molempia funktioita käytettäessä AND- ja OR-koodisanat voivat kuitenkin jatkaa täysin tarkoitetunkaltaista SQL-lausetta joissain toteutuksissa, kuten ylläolevassa tunnistautumisesimerkissä. Ei siis pidä luottautua mysql_query():n antamaan näennäiseen turvaan, vaan aina toteutuskohtaisesti sekä perehtyä käytettyjen ratkaisujen ominaisuuksiin ja haavoittuvaisuuksiin että pitää huolellisesti kiinni hyvistä tietoturvakäytännöistä kuten syötteenkäsittelystä. (Alshanetsky 2005, 74.)

MySQL-tietokantakutsuihin käytettäviä lomakekenttiä, kuten rekisteröitymislomakkeita, kirjautumislomakkeita ja erilaisia ohjauslomakkeita verkkopalvelun toiminnoille täytyy siis suojella SQL-injektioilta. Selkein tapa tähän on käsitellä kaikki ei-numeeriset syötteet PHP:n mysql_real_escape_string-funktiolla. Tämä funktio on nimenomaan MySQL:n erityismerkityksiä omaaville merkeille, kuten lainausmerkeille, kehitetty ratkaisu, joka suojaa nämä ongelmalliset, erityismerkitykselliset merkit. Tämä ratkaisu estää lainausmerkkeihin perustuvan syötteen katkaisun, muttei suojaa injektioilta hauis-

sa, joissa käyttäjäsyöte tulee SQL-lauseen lainausmerkittömään numeroarvoon. Tällöin tulee käyttää PHP:n tyyppimuunnoksia ja muuntaa odotetusti numeroarvoinen käyttäjäsyöte varmasti numeroksi, jolloin numeroalkuisesta SQL-injektiosta tippuisi hyökkäyskomennollinen loppuosa pois, ja se toimisi ainoastaan numeroarvona. (Alshanetsky 2005, 77-78; Ullman 2008, 373; 2010n.)

5.3 Tunnistautumisen, istuntojen ja evästeiden haavoittuvuuksia

Käyttäjän tunnistautuminen verkkopalveluun on usein välttämätön ominaisuus sekä tallennettujen tietojen luottamuksellisuuden että käyttäjän toimien kiistämättömyyden puolesta. Kirjautumisjärjestelmän kautta tunnistautunut käyttäjä voi ylläpitää ainoastaan itselleen tai oman rajauksensa mukaan näkyviä tietoja luottamuksellisesti rajaten niiden julkisuutta. Kuten mainittu taustaluvussa 2.1.6 Evästeiden tai istuntojen välttämättömyys seuraa HTTP:n tilattomuudesta; asiakasselain tai palvelin eivät ilman evästeitä tai istuntoja ja tietokantoja tiedä mikä tieto on jo esitettyä ja kenelle, tai mitä valintoja on tehty. Evästeillä ja istunnoilla voi toki toteuttaa vähemmänkin tietoturvakriittisiä tilamuistitoimintoja kuin kirjautumisen, mutta se on yksi konkreettinen ja keskeinen esimerkki tietoturvallisuudesta puhuttaessa. (Ullman 2008, 327.)

Tunnistautumisen ongelmista puhuttaessa voidaan myös huomioida, että joskus käyttäjät itse ovat riski omien tietojensa luottamuksellisuudelle eri palveluissa ja niiden välillä. Käyttäjät saattavat valita heikkoja salasanoja, jotka ovat alttiita sanakirjahyökkäyksille tai raa'alle voimalle, ja joskus käyttäjät valitsevat saman salasanan useisiin verkkosovelluksiin – tällöin altistaen tunnistautumismielessä jokaisen palvelun kokonaistietoturvallisuuden omalta osaltaan heikoimman toteutuksen tasolle. Tämän vuoksi käyttäjien vahvojen salasanojen valinta kuuluu myös vastuullisille kehittäjille; kehittäjien tulee ohjeistaa ja vaatia vahvojen salasanojen käyttöä toteutettaessa tunnuksien luomista verkkosovellukseen; vahvojen salasanojen ehdottaminen käyttäjille ei riitä, vaan rekisteröitymistoiminnoissa tulisi olla tarkistus, ja vain vahvat salasanat tulisi hyväksyä (Jianxin, Blackwell, Anderson & Alastair 2000). Vahva salasana tulisi olla mahdoton murtaa taustoissa mainituilla sanakirjahyökkäyksillä, ja riittävän monimutkainen, jotta sen arvaaminen raa'an voiman toistuvilla arvauksilla ei olisi ajankäytöllisesti järkevää tietokoneistetustikaan. Lähteestä vaihdellen vahvojen salasanojen suosituspituudeksi annetaan 10-15 merkkiä, joiden tulisi koostua sekä isoista että

pienistä kirjaimista, numeroista ja erikoismerkeistä. Kirjaimien ja muiden merkkien tulisi olla sekaisin keskenään, eikä vain esimerkiksi niin että muutenkin vältettävän sanamuotoisen kirjainsarjan perään on lisätty numeroja. (Jianxin, Blackwell, Anderson & Alastair 2000; Linja-Aho 2010.)

Tunnistautuminen voi myös koskea palvelimen tunnistautumista asiakkaalle kun halutaan todentaa että palvelimen vastaus asiakkaan pyyntöön on edelleen peräisin luotettavasti aidosta lähteestä, eikä tiedonsiirron väliin asettuneelta hyökkääjältä. Tunnistautumisesta vastaavien kirjautumisjärjestelmien osina toimivat tunnistautumistilan tallentava eväste tai sessio, ja itse tunnistautumisprosessi jossa mitataan tunnistettavan entiteetin luotettavuutta. Tunnistautumisprosessiin kuuluvat mahdolliset hyökkääjien väliintuloa estävät tai vaikeuttavat toimenpiteet eli salausmenetelmät. Tästä aiheesta lisää luvussa 5.4 ja 5.5.

Evästeet ovat palvelimelta asiakkaan koneelle HTTP-vastauksen mukana tallennettuja tiedostoja, joiden avulla saadaan paikattua HTTP:n ja PHP:n asiakaspuolen "muistittomuus", eli tilattomuus. Asiakas lähettää joka pyynnön mukana evästeensä palvelimelle jonka yhteyteen eväste on merkitty kuuluvaksi, ja näin voidaan seurata asiakaspuolen muuttuvaa tilannetta palvelun eri toiminnoissa, mukauttaa dynaamista sisältöä evästeisiin tallennettujen valintojen perusteella. HTTP-pyyntöjen mukana asiakas lähettää myös uudet toimenpiteensä sivulla ja riippuen pyytämistään URL:ista tai lomakevalinnoistaan, palvelin päivittää evästeen tiedot PHP:ssa ja lähettää muokatun evästeen jälleen asiakkaalle. Evästeiden heikkouksien vuoksi niitä ei tule koskaan käyttää todentamiseen, identiteetin tai käyttöoikeuksien muistissa pitämiseen elleivät ne ole salattuja; salatut ja turvalliset evästeprotokollatkaan eivät silti korvaa haavoittuvuutta, joka seuraa salasanalomakkeen lähettämisestä palvelimelle selkokielisenä, kun asiakas-palvelin -yhteys ei ole salattu. Turvallisista evästeprotokollista myöhemmin tässä luvussa, ja salatuista yhteyksistä luvussa 5.5. (Benantar 2008, 127-128.)

Istunnot toimivat muutoin samalla tavalla, mutta niiden tietosisältö säilytetään koko ajan palvelimen evästetietokannassa. PHP:ssa on itsessään istuntokäsittelijä, joka huolehtii istuntotietojen tallentamisen, lukemisen ja nimeämisen näiden omaan tietokantaan. Tämän lisäksi ohjelmistokehittäjä voi myös rekisteröidä joukon omia funktioitaan mukauttaakseen istuntoja käyttämään vaikkapa MySQL-tietokantaa paremman tehokkuuden, hallinnan tai tietoturvan vuoksi. Istunnot tuottavat kuitenkin

asiakkaan säilytettäväksi evästeen, jossa säilytetään ainoastaan tunnisteavain jonka perusteella asiakas yhdistetään istuntotietokannassa olevaan, oikeaan istuntoon. Tästedes evästeestä puhuttaessa tarkoitetaan yhtäläisesti sekä istuntoevästettä että tavanomaista evästettä, koska huolimatta näiden luomiseen käytettävistä erillisistä funktioista, käytetyt asetukset ovat samankaltaiset ja molemmat vaativat evästetiedoston olemassaolon asiakaspuolella.

Tietoturvallisuuden puolesta istunnoilla on merkittäviä etuja tavanomaisiin evästeisiin: istuntotietoja säilytetään palvelimella, joten niitä ei voi muokata asiakaspuolelta käsin, niihin saa tallennettua suuremman tietomäärän ja lisäksi ne vaativat vähemmän tiedonsiirtoyhteydeltä, sillä kaikkia tietoja ei tarvitse tavallisen evästeen tavoin lähetellä jokaisen HTTP-kutsun ja -vastauksen mukana (Ullman 2008, 349). Evästeen tietoturvallisuuteen voi vaikuttaa PHP:ssa samalla kun se luodaan, määrittämällä sen ominaisuudet. Evästeen tietoturvan kannalta PHP:n funktio setcookie() voi asettaa rajoituksia evästeen eliniälle, polulle, domain-osoitteelle, evästeen siirrettävyydelle HTTP:n ja HTTPS:n välillä, sekä evästeen luettavuudelle muutoin kuin HTTP:n kautta. Elinikä eli vanhentumisajankohta määrittää milloin asiakasohjelma voi automaattisesti poistaa evästeen. Polkuasetuksella voidaan määrittää eväste lähetettäväksi ainoastaan tietyssä polussa ja sen kaikissa alaosioissa tapahtuvissa pyynnöissä ja vastauksissa. Polku ilmoitetaan suhteellisesti palvelimen www-juureen nähden. Domain-osoitteella voidaan rajata evästeen käyttöä alasivustoilla ja eritellä esimerkiksi foorumit.esimerkki.fi -sivun evästeet kauppa.esimerkki.fi -sivun evästeistä, luomalla molemmille omat evästeensä vastaavine domain-määritteineen. (The PHP Group 2010l, 2010k.)

Istuntoevästeille vastaavat asetukset määritetään session_set_cookie_params()-funktiolla ennen session_start-funktiota() tai PHP:n oletusasetuksista php.ini-tiedostossa. Tämän lisäksi istunnot välittävät istuntotunnuksen URL-kentän mukana sivuilta toisille siirryttäessä, mikäli asiakas ei ole hyväksynyt evästeiden käyttöä sivustolla. URL:n mukana näkyvä selkotekstinen tunniste voi näkyä muille sivustoille linkkien kautta siirryttäessä HTTP-kutsun referrer-kentässä, ja on lisäksi selkotekstinä on altis HTTP--liikenteen kaappaamiselle hyökkääjien toimesta. Tämän vuoksi istunnoille tulisi erikseen pakottaa istuntoevästeen käyttö ilman mahdollisuutta istuntotunnisteen välittämiseen URL:n yhteydessä, ja istuntoja tulisi käyttää HTTPS-suojatun yhteyden kanssa. Istuntoevästeen käytön pakottaminen asiakasohjelmalle onnistuu session.use_only_cookies-asetuksen avulla PHP:n oletusasetuksista php.ini-tiedostosta. Tämän asetuksen

ollessa voimassa istunnot eivät toimi lainkaan mikäli asiakas kieltäytyy evästeistä verkkopalvelun sivustolta, joten tässäkin on huomioitava tietoturvan kustannukset toimivuuden rajautumisena tietyissä tilanteissa suhteessa lisättyyn tietoturvallisuuteen. (The PHP Group 2010k.)

Kaksi viimeistä asetusta, secure ja HttpOnly suojaavat evästettä erilaisilta kaappaushyökkäyksiltä. Evästeen secure-ominaisuuden asettaminen true-arvoon rajoittaa evästeen käytön ainoastaan HTTPS-protokollan mukaisiin sivuosoitteisiin, joissa osoiterivin alussa näkyy https:// tavallisen http:// -protokollamerkinnän sijaan. Tällöin, mikäli sivustolla on suojatun yhteyden turvaamia HTTPS-osia sekä suojaamattomia HTTP--osia, eväste siirtyy asiakkaan ja palvelimen välillä ainoastaan HTTPS-osioihin liittyvissä pyynnöissä ja vastauksissa, eikä niin sanotusti "vuoda" suojaamattomien HTTP-pyyntöjen ja vastausten mukana. HttpOnly-asetuksen määrittäminen true-arvoon on tarkoitus estää muita tekniikoita kuten JavaScriptiä lukemasta evästettä ja vuotamalla JavaScriptin kautta haavoittuvaiseksi samaan tapaan kuin secure-arvon oletusasetuksella HTTPS/HTTP-protokollia sekaisin käyttävässä verkkopalvelussa. Evästeiden vuotaminen näillä tavoilla mahdollistaa vastaavasti sekä istunnonkaappaus että "Cross-Site Scripting" eli XSS-hyökkäykset. (Hafner 2009a, 2009b; The PHP Group 2010l.)

Evästeiden turvalliseen käyttöön on esitetty erilaisia protokollia, joilla oikaista evästeiden käytön heikkouksia sellaisenaan (Fu, Sit, Smith & Feamster 2001; Liu, Kovacs, Huang & Gouda 2005; Benantar 2008, 127-128). Liu, Huang & Gouda (2005) esittelevät parannellun version Fun, ym. (2001) evästeprotokollasta, ratkaisten kiistämättömyyden, luottamuksellisuuden ja eheyden ongelmia evästeiden käytössä sekä edellisen protokollaesityksen heikkouksia. Paranneltu protokolla on täten suositeltu ratkaisu evästeiden ja istuntojen käytössä, vaatimatta ylimääräistä nonce-arvojen hallinnointijärjestelmää ja toimien kohtuullisen kevyesti palvelinpuolella käsiteltynä. Suosituksen tietorakenne on esiteltynä alla, siten, että siihen sisällytetyt tiedot ja varmentavat tiivisteet on kuvattu eriteltyinä tietokenttinään. (Liu, Kovacs, Huang & Gouda 2005, 336.)

käyttäjätunnus|päättymisaika|(data)k|HMAC(käyttäjätunnus| päättymisaika|data|istuntoavain, k)

… jossa:

(data)k = data salattuna avaimella k
salausavain k = HMAC(käyttäjätunnus|päättymisaika, sk)
sk = palvelimen salainen avain
HMAC = Hash-based Message Authentication Code, tiivistepohjainen viestitunniste
istuntoavain = SSL-istunnon avain
HMAC(data, sk) = HMAC, joka koostuu datasta salattuna avaimella sk
(Liu, Kovacs, Huang & Gouda 2005, 336.)

5.4 Salausmenetelmät

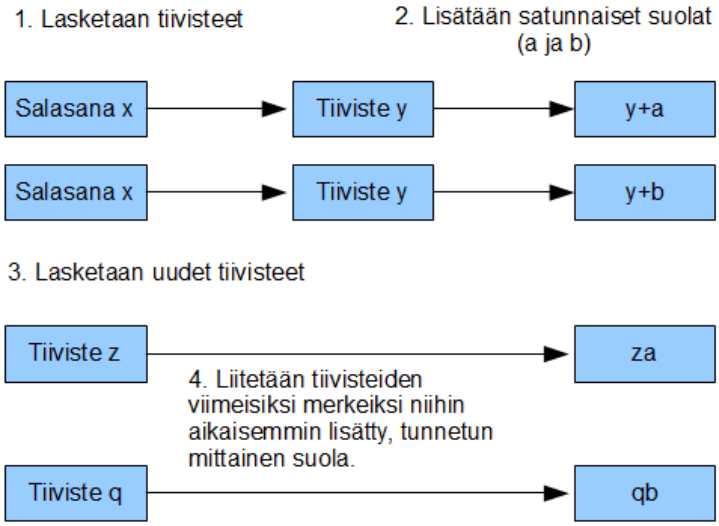
Esimerkiksi tiivisteiden avulla verkkopalvelu voi vahvistaa luottamuksellisuuttansa siten, ettei sen tarvitse tietää suoranaisesti mikä on käyttäjän salasana, vaan ainoastaan tunnistaa onko käyttäjän kirjautuessa antama salasana oikein vai väärin. Tämä onnistuu tallentamalla käyttäjätietoihin ainoastaan salasanasta laskettu tiiviste, ja laskemalla kirjautumissyötteen salasanasta samalla menetelmällä tiiviste, jota verrataan tallennettuun arvoon. (Schneier 1996, 52.)

MD5 on usein sovelluskehityksen opetuksesta tuttu tiivistealgoritmi verkkopalveluja ohjelmoineille. Algoritmin suosio perustuu sen nopeuteen ja yleisyyteen, sen ollessa iäkkäämpänä algoritmina tuettu funktioina useissa eri ohjelmointi- ja komentosarjakielissä. Kyseinen algoritmi ja sen sovellukset ovat kuitenkin käyttökelvollisuuteen nähden oleellisesti murrettu lukuisia haavoittuvuuksia käyttäen jo vuoteen 2008 mennessä (Xiaoyun, Dengguo, Xuejia & Hongbo 2004; Black, Cochran & Highland 2006; Stevens, Lenstra & de Weger 2007; Sotirov, Stevens, Appelbaum, Lenstra, Molnar, Osvik & de Weger 2008; US-CERT 2008). Yhdysvaltojen tietoturvan asiantuntijaelin United States Computer Emergency Readiness Team, lyhyesti US-CERT (2008), suosittelee etteivät ohjelmistokehittäjät, verkkosivustojen omistajat tai tietoturvavarmenteiden myöntäjät käyttäisi MD5-algoritmiä enää missään laajuudessa.

Suositeltavia salaus- ja tiivistealgoritmeja ovat vahvemmat, pidempiin salausavaimiin ja tiivisteisiin perustuvat algoritmit. Tiivistealgoritmeista käyttökelpoisempia ovat 256- ja 512-bittisiin tiivistearvoihin perustuvat SHA-algoritmin vahvemmat toteutukset. Salausalgoritmeista suositeltavampia ovat symmetrinen AES ja epäsymmetrinen RSA. PHP:ssa voi valita eri tiivistealgoritmeja funktion hash parametrina. Funktio ottaa en-

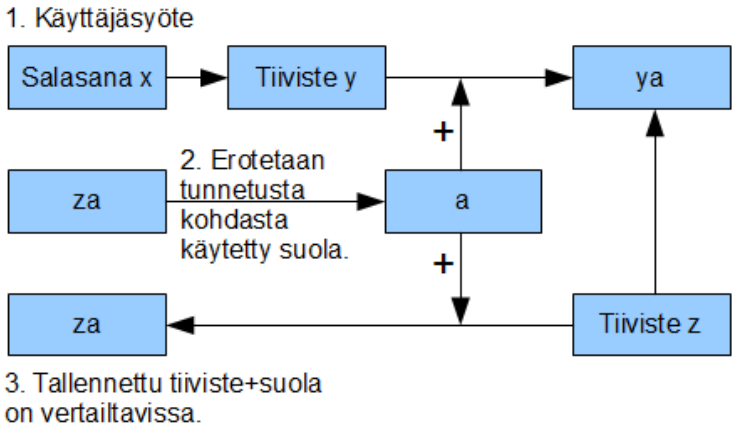
simmäisenä parametrinaan algoritmin tunnuksen numerona tai tekstitunnuksena, ja toisena parametrinaan tiivistettävän lähdetekstin. PHP:n versiosta 5.3 alkaen hash-funktion käytettävissä on ollut myös 224-bittinen SHA-algoritmi, kompromissina uhatumman 160-bittisen SHA-1 ja raskaampien, pidempien tiivisteiden välillä. (Järvinen 2003, 123-124; The PHP Group 2010D.)

Taustoissa esitelty suola vahvistaa tiivisteiden tietoturvaa myös siten, ettei salasanatietokannan käyttöönsä saanut hyökkääjä tai väärin toimiva järjestelmänvalvoja pysty päättelemään usein toistuvista tiivisteistä, että tietyillä käyttäjillä on samoja salasanoja. Tällainen tieto voi johdatella käsitykseen, että samaa salasanaa käyttävillä käyttäjillä on jokin yksinkertainen ja yleisluontoinen salasana, joka voi tulla kenelle tahansa mieleen, jolloin nämä tunnukset olisivat ensisijaisia kohteita sanakirja- ja raa'an voiman hyökkäyksille (brute force attack). Käytetty suola voi olla tallennettuna selväkielisenä salasanan osaksi upotettuna, koska hyökkääjä ei voi tietää mikä osa tietokannassa olevasta salasanatiivisteestä on suolaa ja mikä alkuperäistä tiivistettä. Täten eräs sanakirjahyökkäyksiltä suojaava suolausratkaisu olisi esimerkiksi käyttää jo itsessään melko turvallista SHA-1-tiivistefunktiota, tai sen vahvempia toteutuksia, salasanaan ja lisätä tiivisteen kanssa samanlaisia merkkejä sisältävä satunnainen suola tähän ensimmäiseen tiivisteeseen. Tästä salasanan tiivisteen ja suolan yhdistelmästä laskettaisiin uusi SHA-1-tiiviste, johon vielä liitettäisiin äsken käytetty suolamerkkijono ennalta valittuun kohtaan. Kirjautumisjärjestelmä osaisi sovitusti valitun kohdan mukaan yhdistää syötteenä annettuun salasanaan oikean käyttäjäkohtaisesti satunnaistetun suolan. Kuvissa 4-5 on havainnollistava esitys tiivisteiden ja suolan yhteiskäytöstä. Vaikka hyökkääjä saisi tallennetut tiivisteet käsiinsä, ei suolatuista tiivisteistä voi päätellä useammilla käyttäjillä toistuvia, erityisen heikkoja salasanoja löytyvän tietokannasta, eikä sitä, millä käyttäjillä tällaisia salasanoja on käytössä (kuvio 1). Tiivisteeseen käytetty suola (tiivisteen z tapauksessa a) tallennetaan tunnettuun sijaintiin tiivistemerkkijonoa, josta se voidaan poimia käytettäväksi kirjautumissyötteeseen vertailua varten (syöte+a, tiiviste+a, vertailu). (Schneier 1996, 52-53; Järvinen 2006, 245-247.)



KUVIO 1. Toistuvien tiivisteiden suolaaminen ja suolan merkitseminen uuteen tiivisteeseen

Käyttäjän syöttämän salasanan vertailu tallennettuun ja suolalla vahvistettuun tiivisteeseen perustuu tiivisteen yhteydessä tunnetulla tavalla tallennettuun suolaan, joka ei hyökkääjälle erotu tiivisteestä (kuvio 2). Vaikka hyökkääjä erottaisikin suolatusta tiivisteestä siinä käytetyn suolan arvon, joutuisi hyökkääjä silti yrittäessään arvata käyttäjien salasanoja tutkimaan jokaisen mahdollisen suolatun tiivisteen käyttäjäkohtaisesti, siinä missä ilman suolan käyttöä hyökkääjä voi tehdä esilaskennallisen sanakirjahyökkäyksen tietokantakohtaisesti. Esilaskennalla tarkoitetaan sanakirjahyökkäyksen tulosten tiivistämistä valmiiksi taulukoiksi, jolloin hyökkääjä voi vertailla tietokannassa näkyviä tiivisteitä esilaskettuihin sanakirjahyökkäystiivisteisiin ja löytää siten kaikki sanakirjahyökkäykselle alttiit, heikot salasanat vähimmäisvaivalla. (Järvinen 2006, 246-247.)



KUVIO 2. Salasanasyötteen vertaaminen suolattuun tiivisteeseen kirjautuessa

Salasanojen tallentaminen suolan kanssa ei suojaa verkkopalvelua siltä, että hyökkääjä kokeilisi sanakirjahyökkäyksen tapaan eri salasanoja peräkkäin verkkopalvelun omaan kirjautumislomakkeeseen, sillä tämä käyttäisi palvelun omaa kirjautumisjärjestelmää, joka osaisi käyttää oikeaa suolaa salasanan ja tiivistearvon vertaamiseen. Tällaiseen brute force -hyökkäykseen tehoaa ainoastaan kirjautumisjärjestelmään ja käyttäjätietokantaan liitetty toiminto, joka lukitsee käyttäjätunnuksen väliaikaisesti tai pysyvästi liian monen väärän yrityksen jälkeen. (Järvinen 2006, 245-246; Ullman 2008, 388.)

Tiedonsiirrossa kahden osapuolen välillä tietosisällöstä ja jostain salaisesta, molempien osapuolien tuntemasta arvosta voidaan laskea tiiviste, jolla voidaan varmentaa viestin eheys eli muuntumattomuus kolmansien osapuolien toimesta (Järvinen 2006, 122-123). Silti hyväkin salaus tiedonsiirtoyhteydessä vaatii osapuolien keskinäistä todentamista, jotta tiedon alkuperään, eli lähettäjään, ja muuttamattomuuteen yhteyden ääripäiden välillä voidaan luottaa. "Ketju on juuri niin vahva kuin sen heikoin lenkki", muistuttaa Järvinen (2003, 122) tiivistäen tietoturvan kokonaisuutta sananlaskuun.

5.5 Palvelinyhteyden suojaaminen, HTTPS ja X.509

Teoriassa myös jokainen tiedonsiirtoväli asiakkaan selaimen, PHP-palvelimen ja mahdollisesti erillään olevan MySQL-palvelimen välillä muodostaa oman tietoturvariskinsä. Mikäli ulkopuolinen taho pystyy kaappaamaan HTTP-pyynnön esimerkiksi asiakkaan

kirjautumislomakkeen lähetyksen ja PHP-palvelimen väliltä, saa hyökkääjä käyttäjän kirjautumistiedot haltuunsa. Mikäli hyökkääjä saa PHP-palvelimen vastauksen kaapattua esimerkiksi kirjautumislomakkeen vastaanoton jälkeen, hyökkääjä voi saada käyttäjän selaimelle tarkoitetun evästeen tai istuntoevästeen haltuunsa ja täten pääsyn palveluun kirjautuneena käyttäjänä, aidon käyttäjän identiteetillä. Tämä sortaa käyttäjän yksityisyyden puolesta luottamuksellisuutta ja verkkopalvelun kannalta kiistämättömyyttä, ja lisäksi kirjautumisen varastanut hyökkääjä voi mahdollisesti aiheuttaa vakavia lisähaittoja palvelun sisältä käsin, esimerkiksi poistamalla tietoa, lisäämällä valheellista tietoa tai varastamalla käyttäjän identiteettiin kuuluvia tietoja.

Pelkästään palvelinpuolen ohjelmointiin perustuva PHP-pohjainen verkkopalvelu ei pysty käsittelemään verkkosivun lomakkeisiin syötettyä luottamuksellista dataa ennen kuin se kulkee verkkoyhteyden yli asiakkaalta palvelimelle. Verkkosivun lomaketiedot kulkevat asiakkaalta palvelimelle raakatekstimuodossa. Tällä välillä data on haavoittuvaista ja kaapattavissa hyökkääjien toimesta. Toisin sanottuna mikäli verkkopalvelun ja asiakasselaimen välillä ei ole suojattua, eli salattua, tiedonsiirtoprotokollaa, kulkee data näiden tahojen välillä aina selkokielisenä. Salaamattoman yhteyden kautta välitetty data on kaikkien samassa verkossa asioivien hyökkääjien kaapattavissa, ja selkokielisenä kaikki arvokas tieto on kaiken lisäksi suoraan käyttökelpoista.

Esimerkkinä sessioiden haavoittuvuudesta selkotekstisen HTTP-tiedonsiirron käytössä ovat suojaamattomien tai osittain suojattujen verkkopalveluiden ja avoimien langattomien verkkojen tunnettujen uhkien uudet esiintymismuodot. Tunnetut uhkat muodostuvat verkkopalvelujen kirjautumisjärjestelmien yleisestä heikkoudesta ilman koko palvelun kattavaa TLS-protokollan suojaa tai vain osassa verkkopalvelujen sivuista käytettyä TLS-protokollan suojaa yhdistettynä evästeisiin, joita ei ole erikseen pakotettu turvallisiksi luvun 4.2.3 osoittamilla tavoilla. Turvattomat evästeet ja istuntoevästeet "vuotavat" vain osittain HTTPS:nä tarjottujen sivustojen HTTP-kutsujen yhteydessä salaamattomaan, selkokieliseen verkkoliikenteeseen.

Firesheep on hyvin yksinkertaisella käyttöliittymällä varustettu Firefox-liitännäinen, joka mahdollistaa HTTP-pakettien, ja näistä paketeista löytyvien istuntoevästeiden kaappaamisen kertaklikkauksella, kaappaamisen verkosta, johon hyökkääjällä on pääsy. Uhka koskee erityisesti avoimien langattomien verkkojen käyttäjiä, koska näissä verkoissa hyökkääjä pääsee erityisen vaivattomasti käyttäjän ja palvelimen väliseen

liikenteeseen käsiksi. Liitännäinen julkaistiin 24.10.2010 ja toimii Firefoxin 3.x-sarjan versioissa, muttei vielä kirjoitushetkellä beetakehityksessä olevassa 4.x-sarjassa. Liitännäisen tarkoitus on tekijänsä Mark Butlerin (2010) mukaan tuoda esille kauan vähäteltyä haavoittuvuutta ja lisätä ihmisten tietoisuutta HTTPS-turvallisuudesta, sillä liitännäisen käyttämät hyökkäystekniikat ovat olleet jo pitkään käytössä, mutta ongelmia on vähätelty, eikä HTTPS edelleenkään ole kattavassa käytössä verkkopalveluissa. Windows-käyttäjien tulee asentaa WinPCap-ohjelmisto ennen Firesheepin liittämistä Firefoxiinsa, mutta muuten liitännäinen on hyvin maallikkotajuinen ja laajasti levinnyt keino varastaa kirjautuneiden käyttäjien istuntoja ja profiileja eri verkkosovelluksiin. Firesheep-liitännäistä kehitetään avoimesti, ja sen kykyä varastaa eri palvelujen istuntoevästeitä kasvatetaan (kuva 7). (Hyppönen 2010; Butler 2010; Garreffa 2010)

Palvelun nimi	Domain
Amazon	amazon.com
Basecamp	basecamphq.com
bit.ly	bit.ly
Enom	enom.com
FaceBook	facebook.com
FourSquare	foursquare.com
Github	github.com
Google	google.com
Hacker News	news.ycombinator.com
Harvest	harvestapp.com
The New York Times	nytimes.com
Pivotal Tracker	pivotaltracker.com
Twitter	twitter.com
ToorCon: San Diego	sandiego.toorcon.org
Evernote	evernote.com
Dropbox	dropbox.com
Windows Live	live.com/bing.com
Cisco	cco.cisco.com
Slicehost	manage.slicehost.com
Gowalla	gowalla.com
Flickr	flickr.com

KUVA 7. Lista Firesheepille alttiista palveluista 7.12.2010 (Firesheep Github Wiki 2010)

Suojaamattoman HTTP:n heikkouksia paikkaamaan tarkoitettuja, suojattuja tietoliikenneprotokollia ovat verkkopalvelujen kannalta hyödynnettävät Secure Sockets Layer, SSL, ja sen nykyinen, kehittyneempi versio on nimeltään Transport Layer Security, TLS. Nykyisin nimiä käytetään lähes verrannollisesti keskenään, tai yhdistetyssä muodossa SSL/TLS, korostaakseen että TLS:ssa kyseessä on käytännössä sama asia kuin monille aiemmin tutussa SSL-tekniikassa. HTTP:n heikkouksiin pureutuva HTTPS ei ole itsenäinen protokolla, vaan viittaa HTTP-tiedonsiirtoon joka toteutetaan SSL- tai TLS-suojauksella.

HTTPS-protokollan käyttö näkyy käyttäjän selaimen osoiterivillä https://-alkuisena URL-merkintänä ja selaimen alapalkissa lukon kuvana useimmissa selaimissa. Palvelimen puolelta HTTPS:n käyttöönotto vaatii sarjan toimenpiteitä. Ensinnäkin verkkopalvelun täytyy pystyä luotettavasti tunnistautumaan sertifiointiauktoriteetille (CA), joka varmentaa palvelun identiteetin, yleensä perustuen sähköpostiosoitteeseen ja verkkopalvelun domain-nimeen. Useimmat varmenteet ovat maksullisia, mutta nykyään Start Commercial Limited -yritys tarjoaa StartCom Certification Authorityn nimellä matalan luottamustason varmenteita vuodeksi kerrallaan täysin ilmaiseksi. Samalta yritykseltä on maksua vastaan saatavilla tarkempiin identiteettitarkistuksiin perustuvia korkeamman luottamuksen varmenteita. Nämä luottamustasot perustuvat identiteetin tarkastuksen tarkkuuteen, jossa korkeammilla tasoilla identiteetin todentaminen perustetaan useampiin ja virallisempiin ominaisuuksiin, kuten yritystietoihin, verrattuna alimman tason sähköposti- ja domain-tarkistukseen.

PKI eli Public Key Infrastructure on järjestelmä joka koostuu luottamukseen perustuvaa tietoturvaa mahdollistavien varmenteiden jakeluvaiheista. Tekijöinä tässä ovat erityisesti ohjelmistot, ihmiset, politiikat ja toimitavat, joiden avulla identiteetteihin sidottavia, luottamusta osoittavia varmenteita jaellaan. varmenteet perustuvat luottamukseen suoraan tai ulkoisestetusti eri tahojen välillä.

HTTPS:n käytössä tietoturva perustuu X.509-varmenteisiin siten, että varmentajan antama varmenne verkkopalvelulle osoittaa varmentajan luottamusta verkkopalveluun. Asiakaspuolella selaimeen on asennettu juurivarmenteita, jotka osoittavat selainkehittäjän luottamusta näiden juurivarmenteiden varmentajiin. Näin verkkopalvelun käyttäjä, jolla on selaimessaan juurivarmenne, voi varmistua siitä että HTTPS-protokollan kautta

tarjottu sivu kuuluu sille taholle joka varmenteessa lukee, eikä ole hyökkääjän tekemä huijauskopio alkuperäisestä sivusta.

Kuka tahansa pystyy tuottamaan varmenteita, joten on tärkeää että varmenteen on allekirjoittanut ja myöntänyt luotettava taho, eikä se ole osa huijausyritystä, jonka kaltaisia varmennejärjestelyn on nimenomaan tarkoitus pyrkiä estämään. Jotta HTTPS:n voisi katsoa olevan tietoturvallinen selaimen käyttäjän tulee palveluun luottaakseen pystyä luottamaan sekä selainkehittäjän näkemykseen, eli juurivarmenteiden luotettavuuteen, että verkkopalvelun varmenteen myöntäneeseen varmentajaan. Lisäksi varmenteen salausalgoritmin on oltava riittävän vahva, ettei sitä pystytä purkamaan ja sen yksityistä avainta väärinkäyttämään huijauskopioiden tekemiseen. Tämä luottamuksen vaatimus on yksi tekijä joka kannustaa varmentajia aidosti huolehtimaan myönnettävien varmenteiden taustatarkastuksista, sillä heidän palvelunsa luotettavuus on heidän toimintansa elinehto.

Yksinkertaistettuna varmenne koostuu x:n julkisesta avaimesta, varmentajan allekirjoituksesta ja varmennetta täsmentävistä, varmentajan takaamista tarkentavista tiedoista. Allekirjoitus on viestin tiiviste salattuna varmentajan salaisella avaimella, joten asiakasselain pystyy käyttämään juurivarmenteessa olevaa varmentajan julkista avainta todentamaan sekä varmenteen allekirjoittajan aitouden että sisällön muuttumattomuuden vertaamalla purettua tiivistettä itse laskettuun tiivisteeseen.

5.6 Tietokannan tietoturva

Vaikka PHP:n lähdekoodi itsessään on suojattu ulkoiselta tarkastelulta, eikä ohjelman sisäinen rakenne tätä kautta voi paljastua hyökkääjille, on silti hyvän ohjelmointitavan mukaista ja jatkokehitykselle hyödyllistä pitää sovelluksen asetukset koottuna erillään toiminnallisesta koodista. Tämän vuoksi erilliset asetustiedostot ovat suositeltavia. Tällaisia asetuksia ovat erityisesti verkkopalvelusovelluksen käyttämät tunnistautumistiedot PHP:ssa sen hyödyntämille muille palvelimille tai palveluille kuten MySQL-tietokantaan. MySQL-palvelimen tunnusten suojaaminen on ratkaisevan tärkeää, joten kehityksessä tulee lähtökohtaisesti karsia mahdollisuus tietokantatunnusten vuotamisesta PHP:n virhepalautteiden tai tulosteiden mukana verkkopalvelussa. Näin ollen PHP:n puolella toteutetaan paljon tietoturvaa - josta esimerkkinä ovat myös SQL-injektioiden

karsimiset syötteenkäsittelyssä - joka on suoraan yhteydessä MySQL:n tietoturvaan. Tästä huolimatta tietokantapuolen tietoturvaa tulee vahvistaa myös suoraan muilla menetelmillä. Tietokantaan yhdistämiseksi tarvittavien tunnuksien luottamuksellisesta säilyttämisestä ja käytöstä luvussa 5.8.

Verkkopalvelun käytettävyyden kannalta on tärkeää että SQL-lauseet, jotka ajetaan MySQL:ssa ovat kirjoitettu mahdollisimman tehokkaan eli tässä tapauksessa nopean tietokantakäsittelyn mukaisiksi. Mitä lyhyemmän ajan MySQL joutuu käyttämään tuottaessaan vastauksia saamilleen SQL-lauseille, sitä pienempi on PHP:n ja MySQL:n välinen pullonkaula verkkopalvelun suorituskyvyssä, ja sitä käytettävämpi palvelu on. Parempi tehokkuus jättää myös enemmän varaa suuremmalle käyttäjäkuormitukselle, jolloin palvelu pysyy toimintakykyisenä suuremmassa kertakuormituksessa.

Vaikka verkkosovelluksen tietoturva itsessään olisi kunnossa, tietokannan ja tietokannan väli olisi suojattu, on silti huomioitava, että tietokantapalvelimen tietoihin voi päästä käsiksi varkauden tai häikäilemättömän järjestelmänvalvonnan kautta. Verkkosovelluskehittäjän tulisi täten suojella käyttäjien yksityisyyttä ja kehityksen luottamuksellisuutta myös itseltään. Mitään ratkaisevalla tavalla yksityisiä tietoja, kuten salasanoja, ei saisi koskaan säilyttää yksityisyyttä haavoittavalla tavalla. Tämä tulee huomioida myös tietokantojen tallentamia tietoja suunnitellessa. Kirjautumistoiminnallisilla verkkosovelluksilla on joskus tapana tarjota toiminto, jolla unohtuneen salasanan saa palautettua muistutuksena sähköpostiin – tämä on kuitenkin kestämätön ratkaisu jo tietokantapuolen luottamuksellisuuden puolesta. Samaan ongelmaan paljon parempi ratkaisu on palauttaa sähköpostiin käyttäjätilille asetettu uusi, satunnainen ja väliaikainen salasana. Käyttäjätietoja talletettaessa tietokantaan tulee aina tallentaa kirjautumissyötteisiin ainoastaan vertailtavissa oleva tiiviste, ei selkokielistä salasanaa. Tarkempi kuvaus tiivisteiden käytöstä, sekä niiden vahvistamisesta ”suolalla” löytyy salausmenetelmiä koskevasta luvusta. (Cannon 2004, 258-260; Atwood 2007; Ullman 2008, 383-388.)

5.7 Eheyden varmistaminen transaktioilla

Kun tietokantaan lisätään tietokokonaisuuksia, jotka kattavat toisiinsa olennaisesti liittyen useampia pöytiä, tulee huomioida keinoja varmentaa tallennetun ja ylläpidetyn

tiedon eheyttä. Jos tietokannassa on tallennettuna kokonaisuus, jonka osaa säilytetään yhdessä taulussa ja muita osia toisissa tauluissa loogisina osakokonaisuuksina, tulee välttää ettei kokonaisuuden osia voi lisätä ilman samaan kokonaisuuteen kuuluvia muita osia, eikä kokonaisuuden osia voida poistaa yksittäin. Toisin sanottuna tällöin tulee varjella kokonaisuuksien eheyttä.

Mainitun kaltaisia ongelmia voi seurata sekä kokonaisuuksia lisätessä että poistaessa. Mikäli vain osa tauluista päivittyy ja yhden tai useamman osakokonaisuuden tauluihinsa tallentaminen, tai poistaminen, epäonnistuu, jää kokonaisuus vajaaksi eli epäeheäksi. Kaikkien taulujen onnistuneen ja samanaikaisen päivittymisen varmistaminen PHP:n puolelta tarkastellen on hyvin vaikeaa ja vaatii todennäköisesti erityisjärjestelyjä myös taulujen rakenteelta. Helpompi tapa valvoa tietokokonaisuuksien eheyttä ovat MySQL:n tukemat transaktiot.

MySQL-transaktio koostuu aloituskomennosta, suoritettavista SQL-lauseista, sekä transaktion lopetuskomennosta, jotka suoritetaan MySQL:ssa. Aloituskomentoa seuraavat SQL-lauseet ajetaan "väliaikaisesti", ja mikäli jokainen lause suoritetaan onnistuneesti, ilman virheitä, ja myös transaktion lopetuskomento suoritetaan jonon lopuksi, muutokset hyväksytään pysyvinä (kuva 8). Tämä eheyttä varmentava menetelmä edellyttää että transaktion sisältämä SQL-komentojoukko ajetaan PHP:n mysqli_multi_query()-funktiolla, sillä tavanomainen mysql_query()-funktio ei tue useamman SQL-lauseen peräkkäistä suorittamista samassa komennossa. On huomattava, että tällöin korostuvat myös luvussa 5.2 käsitellyt SQL-injektiot ja niiden käsittelyyn on kiinnitettävä yhä tarkempaa huomiota. Mikäli useamman SQL-lauseen transaktiossa tapahtuu virhe, joka havaitaan PHP:n puolella mysqli-kyselyn tuloksen tarkastuksesta, voidaan ajaa uusi mysqli-kysely, jonka sisältönä on SQL:n ROLLBACK-komento. Myös mysqli_rollback() ajaa saman asian. Tämä palauttaa tietokantataulut tilaan ennen transaktioyrityksen aloittamista ilman että vain osa lauseista olisi suoritettu onnistuneesti ja tauluissa olisi epäeheää tietoa. Transaktiot toimivat MySQL:n InnoDB-tyyppisiksi määritetyissä tauluissa, mutta ei ISAM- tai MyISAM-tauluissa. (The PHP Group 2010a, 2010h, 2010g.)

```php
<?php

$mysqli  = mysqli_connect( "localhost", "owner", "pass", "db", 3306, "/var/lib/mysql/mysql.sock" );

$QUERY = <<<EOT
START TRANSACTION;
SELECT @lng:=IF( STRCMP(`main_lang`,'de'), 'en', 'de' )
FROM `main_data` WHERE  ( `main_activ` LIKE 1 ) ORDER BY `main_id` ASC;
SELECT `main_id`, `main_type`, `main_title`, `main_body`, `main_modified`, `main_posted`
FROM `main_data`
WHERE ( `main_type` RLIKE "news|about" AND `main_lang` LIKE @lng AND `main_activ` LIKE 1 )
ORDER BY `main_type` ASC;
COMMIT;
EOT;

$query = mysqli_multi_query( $mysqli, $QUERY ) or die( mysqli_error( $mysqli ) );

if( $query )
{
  do {
    if( $result = mysqli_store_result( $mysqli ) )
    {
      $subresult = mysqli_fetch_assoc( $result );
      if( ! isset( $subresult['main_id'] ) )
        continue;

      foreach( $subresult AS $k => $v )
      {
        var_dump( $k , $v );
      }
    }
  } while ( mysqli_next_result( $mysqli ) );
}

mysqli_close( $mysqli );

?>
```

KUVA 8. PHP.net-käyttäjän undefined esimerkki transaktioiden käytöstä (The PHP Group 2010h)

5.8 Huolimattomat sovellusasetukset avaavat haavoittuvuuksia

Verkkosovelluskehityksessä määritetään eri ohjelmistojen ja itse sovelluksen asetuksia monella yhtä lailla tärkeällä tasolla; hyvin keskeisiä tekijöitä ovat virheraportoinnin asetukset ja toteutus, tietokantaan yhdistämistä varten tallennettujen tunnusten salassapito (Alshanetsky 2005, 80-83), HTTPS-suojauksesta vastaavat palvelimen sertifikaattiasetukset ja sertifikaatin turvallinen säilyttäminen (Fleishman 2009).

SSI-tiedosto on palvelinohjelmiston puolella sisällytettävä tiedosto (Server Side Include file), erotuksena komentojonotulkin, kuten PHP:n, koodin itse sisällyttämä jatkokoodi toisesta tiedostosta. Esimerkiksi Apache HTTP-palvelin pystyy SSI-tiedostojen avulla hallinnoimaan osaa tietoturvavastuusta säilytettäessä luottamuksellisia tunnistetietoja. SSI-järjestelyn verkkosovelluksille ehkä yleiskäyttöisimpänä, tietoturvallistavana tarkoituksena on tallettaa tietokantayhteyden luottamukselliset tunnukset SSI-tiedostoon,

joka on ainoastaan HTTP-palvelimen, kuten Apache HTTP Serverin, luettavissa tämän käynnistymisen ajan. SSI-tiedostoon kirjoitetaan komennot, joilla sen lukeva palvelinohjelmisto ajaa käynnistyessään luottamukselliset tiedot kerran PHP:n MySQL-yhteyden oletusarvoiksi – käynnistystoimenpiteidensä jälkeen Apache menettää korkeammat käyttöoikeutensa myös SSI-tiedostoon, ja PHP:n tietokantayhteysrajapinnan oletusmuuttujat eivät ole suoraan ja selkokielisinä enää luettavissa edes palvelimen PHP-sovelluksille, jotka saattaisivat niitä vuotaa julkisiksi esimerkiksi virheraportoinnin puutteiden ja huolimattomuuden seurauksena. (Alshanetsky 2005, 79-83; Apache HTTP Server Project 2010; The PHP Group 2010b.)

Apache-palvelimen käyttöohjeiden mukaan asetustiedoston httpd.conf ja käytettävän SSI-tiedoston tulee olla suojattu muilta käyttäjiltä ja ryhmiltä palvelinkäyttöjärjestelmän pääsynvalvontamentelmillä. Verkkosovelluksen tiedostosijaintiin HTTP-palvelimella asetettavan .htaccess-tiedoston tulee sisältää sisällytyskomento käyttöoikeuksilta rajatussa sijainnissa suojatulle SSI-tiedostolle. Apachen dokumentaatio suosittelee käyttämään IncludeNOEXEC-määrettä sisällytykseen .htaccess-tiedostossa, jolloin ei ole vaaraa haittakoodin ajamisesta, mikäli joku onnistuisi korvaamaan sisällytettävän tiedoston vahingollisella näköistiedostolla. SSI-tiedoston sisältöön tässä käsiteltyyn PHP:n MySQL-palvelinyhteysoletusten asettamiseksi tarvitaan ainoastaan PHP:n Apache-moduulin komentoa php_admin_value. Koko prosessista on tiivistetty esimerkki kuvissa: .htaccess-tiedosto verkkosovelluskansiossa (kuva 9) ja SSI-tiedosto MyDBdefaults.cnf (kuva 10). .htaccess-tiedostossa olennainen määre on sisällytystoiminto Include, mutta sitä edeltää tietoturvaa parantava määrite, joka kieltää ajamasta ohjelmakoodia sisällytettävistä tiedostoista. Includen määreenä on Apachen palvelinjuurelle suhteellinen tiedostopolku suojattavaan esimerkki-SSI-asetustiedostoon MyAppDBdefaults.cnf. SSI-esimerkkitiedostossa MyAppDBdefaults.cnf (kuva 10) asetetaan PHP:n MySQL-asetuksien oletusarvoiksi yhdistäminen samalla palvelimella toimivaan MySQL-palvelimeen käyttäjätunnuksella ”LimitedUser” ja tämän tunnuksen salasanalla ”SomeStrongPassWord”. Tällä tavoin SSI-tiedostossa oletusarvoiksi asetettuina näitä luottamuksellisia tietoja ei ole PHP:n tiedossa, eikä siten sovelluskoodin virheistäkään vahingossa vuodettavissa. Onnistuneen SSI-järjestelyn jälkeen PHP:n koodissa olevat MySQL-yhteysfunktio mysql_connect() toimii ilman parametrien syöttämistä, jolloin ei tarvita luottamuksellisten tietojen sisällyttämistä lähdekoodiin. (Alshanetsky 2005, 79-83; Apache HTTP Server Project 2010.)

```
Options IncludesNOEXEC

Include [tiedostopolku suhteessa Apachen ServerRoot-polkuun]/MyAppDBdefaults.cnf
```

KUVA 9. Esimerkkitiedosto .htaccess

```
php_admin_value mysql.default_host 127.0.0.1
php_admin_value mysql.default_user LimitedUser
php admin value mysql.default password SomeStrongPassWord
```

KUVA 10. Esimerkkitiedosto MyAppDBdefaults.cnf

Aiemmassa luvussa käsiteltyyn SQL-injektiohyökkäykseen liittyen verkkosovelluksia suunnitellessa tulee pitää huolta, etteivät syötelomakkeiden kenttien nimet (X)HTML--lähdekoodissa vastaa suoraan tietokannan taulujen ja sarakkeiden nimiä. Sovellus ei myöskään muilla tavoin koskaan saa tulostaa käyttäjälle asetus- ja rakennetietojaan – vastaavanlaisista yhtenevien nimikkeiden paljastuksista voi muodostua merkittävä etulyöntiasema tietokantarakennetta hyödyntävälle injektiohyökkääjälle (The PHP Group 2010m). Tässäkin luvussa jo mainitun luottamuksellisten tietojen vuotamisen estämiseksi toinen menetelmä luottamuksellisen tiedon edellä esitetyn suojaamisen lisäksi on suunnitelmallinen ja hyvin toteutettu virheiden käsittely ja lokiin kirjaaminen. Tällä tavoin voidaan minimoida tietovuotojen mahdollisuutta, vuodettavissa olevan tiedon määrän lisäksi. (Alshanetsky 2005, 79-80.)

Kun verkkosovellusta kehitetään – aina siten, ettei siihen ole käyttäjillä tai hyökkääjillä julkisesta verkosta avointa pääsyä – PHP:n virheraportointi on hyödyllinen apukeino paikannettaessa ja poistaessa virheitä. Valmis verkkosovellus taas toimii tuotantoympäristössä, jossa sen oletetaan olevan käytössä, ja käytettävyyden vuoksi sovellusvirheiden korjaamiseksi tarvitaan joka tapauksessa useimmiten huoltokatkoksia, eikä kehittäjä tällöin ole nojaamassa virheraportointiin. Tällöin virheraportointi voi tuoda virheilmoituksia näkyviin vahingossa tai tahalleen aiheuttaneille käyttäjille tai hyökkääjille, ja siten paljastaa luottamuksellista tietoa sovelluksen rakenteesta ja toiminnasta. Toteutusympäristön virheilmoitusten tulisi olla yleisluontoisia ja yhteneviä. Tarkemmat virheet tulee tallettaa vain kehittäjän luettavissa olevaan lokitiedostoon.

Verkkosovellukseen ei myöskään kannata missään nimessä toteuttaa mitään ulkoisesti aktivoitavaa kytkintä virheraportoinnin hallintaan, eli virheraportoinnin käyttöaste tulisi

olla hallittu ainoastaan lähdekoodista tai PHP-asetuksista käsin, eikä missään nimessä GET-tyyppisesti URL-rakenteessa, josta hyökkääjä voi arvata virheraportointitilan käynnistävän komennon joko pääteltävissä olevana nimenä tai jopa raa'an voiman menetelmällä. (Ullman 2008, 208-209; The PHP Group 2010b.)

Ullman (2008) varoittaa huolimattomasta virheraportoinnista ja sen vaarallisuudesta verkkosovelluksen ollessa käytössä toteutusympäristössä. PHP:n dokumentaatiosta (2010b) löytyy kuitenkin joukko asetuksia (The PHP Group 2010j), ja suosituksia niille tietoturvan vahvistamiseksi. PHP-julkaisujen mukana tulee myös kaksi erillistä versiota komentojonotulkin oletusasetuksille, php.ini-development ja php.ini-production, sekä kehitysympäristöä että turvallisempaa toteutusympäristöä varten. Pelkästään virheraportoinnin poiskytkemiseksi tuotantoympäristössä riittävät PHP-dokumentaation lyhyesti esittämät asetukset (kuva 11). (Ullman 2008, 208-209; The PHP Group 2010b.)

```
display_startup_errors  false
display_errors          false
log_errors              true
error_log               [virhelokitiedoston nimi]
```

KUVA 11. PHP:n dokumentoinnissa suosituksiksi annetaan seuraavien asetuksien käytöstä tuotantoympäristössä (The PHP Group 2010b)

6. JOHTOPÄÄTÖKSET JA POHDINTA

Tässä luvussa on tutkielmaa seuranneita johtopäätöksiä, kehitysideoita tämän tutkielman jatkokehitykseen ja mahdollisille tuleville tutkielmille, sekä yleistä itsearviointia opinnäytetyön onnistumisesta. Nämä ovat jaettu omiksi alaluvuikseen 6.1, 6.2 ja 6.3.

6.1 Johtopäätökset

Työn tuloksista löytyy useita hyvin yksinkertaisia toimenpiteitä ja ratkaisuja, joiden hyöty todelliselle tietoturvalle toteutuu vasta niiden käytössä kokonaisuutena. Osa toimenpiteistä tukee toisiaan, kun taas osalle toiset niistä tuntuvat lähinnä päällekkäisiltä vararatkaisuilta mikäli ensimmäinen puuttuu tai pettää. Esimerkiksi HTTPS suojaa itsessään verkkoliikenteen, mutta tämä on turhaa mikäli evästeet pääsevät vuotamaan HTTP-kutsujen myötä. Toisaalta työssä esitelty turvallinen evästeprotokolla mitätöi evästeiden osalta tarpeen HTTPS:lle tai voi toimia korvaavana ratkaisuna mikäli HTTPS:n käyttöönotto ei ole esimerkiksi palvelinrajoitteilla mahdollista tai luotettavien salausvarmenteiden puolesta taloudellisesti mahdollista. Tässä on kuitenkin pidettävä mielessä, että korvaavuus koskee vain evästeiden suojaa, ja HTTPS säilyy edelleen välttämättömänä todella tietoturvalliselle sovellukselle, sillä tunnistautumistoimenpiteiden aikana hyökkääjän on mahdollista kuunnella HTTP-liikenteestä käyttäjätunnuksia ja salasanoja sekä muuta luottamuksellista lomaketietoa, mikä ohittaa evästeiden tarjoamaa näennäistä tietoturvaa.

Tutkimustyön aikana esiintyi jatkuvasti uusia aihealueita, tietoturva-aukkoja tai hyökkäyskeinoja ja niiden ratkaisuja. Esimerkiksi tietoturvan kustannuksien arvioinnissa palvelin- ja tietokantasuorituskyvyn kannalta olisi syytä vielä jatkotutkimukselle. Samoin olisi tarpeen tarkastella käyttäjien vastuuta verkkopalveluiden käyttäjinä ja yhteiskunnan osana kokonaistietoturvalle esimerkiksi salasanojen paremman käytön ja jonkinlaisen tietoturvalukutaidon puolesta. Nykyään ihmiset käyttävät edelleen hyvin tietämättömästi ja varomattomasti lukuisia palveluja, välinpitämättömänä omalle yksityisyydelleen tai sille mihin palveluun milloinkin antavat suostumuksia tai mitä linkkejä klikkailevat. Sekä sovelluskehittäjien että sovellusten käyttäjien kannalta tulisi perehtyä myös turvallisen tietojenkäsittelyn alustoihin, eli teoriaan ja ohjeistukseen oman tieto-

jenkäsittelyjärjestelmänsä ajankohtaisuudesta ja tietoturvallisuudesta. Toisaalta, tietojenkäsittelyn ja ohjelmistotuotannon opiskelijana pidän omituisena että koulutusohjelmassamme käyttäjäpuolen tietoturvaa tunnutaan käsittelevän melkeinpä enemmän kuin sovelluskehityksen näkökulmaa tietoturvariskeihin ja hyökkäyskeinoihin.

6.2 Kehitysideoita

Lähteissä ja dokumentaatioissa esiintyvät ratkaisut tekevät vaihdellen hyvinkin vapaasti oletuksia tekniikoiden käyttöympäristöstä palvelinkäyttöjärjestelmän tai palvelinohjelmistojen puolesta. Kuitenkin todella kattavan ohjeistuksen luomiseksi olisi toivottavaa nähdä esimerkiksi tässä työssä kerättyjä tekniikoita ja ratkaisuehdotuksia esiteltynä ja käsiteltynä huomattavasti nykyistä laajemmin. Parempien ja tarkemmin vaihe-vaiheelta etenevien ohjeiden ja suositusten tuottaminen parantaisi tämän tutkielman arvoa vertaislukijoille, joiden parhaat edut eivät tämän työn itseoppimismäärityksen ohella täyttyneet ihanteellisesti. Myös omien opinnäytetöidensä arvoisina aiheina olisi vertailla enemmän vaihtoehtoja tässäkin työssä esitellyille tekniikoille, ja arvioida eri palvelin- ja käyttöjärjestelmäratkaisujen hyötyjä ja haittoja, tai esimerkiksi useampien ilmaisten tietokantaratkaisujen tehokkuutta eri tyyppisessä käytössä suhteessa toisiinsa.

Tässä työssä käsittelemättömiä aiheita, kuten PHP-komentojonotulkin yhteisölähtöistä Suhosin-tietoturvapaikkausprojektia ja useiden sovellusten keskeisiä yhteiskirjautumisjärjestelmiä sekä OAuth-tyyppisiä todentamisen ja käyttöoikeuksien hallinnoinnin tekniikoita käsittelemällä voisi hyvin jatkaa tämän tutkielman laajentamista. Myös verkkopankkien tai muiden vastaavien tietoturvakriittisten kohteiden ratkaisuista tehtyjä tutkielmia olisi niiden löytyessä voinut käyttää hyvin perusteltuina ratkaisuvalintoina ja menettelyinä esiteltäväksi tässä tai samoissa jalanjäljissä seuraavissa opinnäytetöissä.

Tietoturvan kokonaisuutena toimimisen vuoksi on hyvin vaikea rajata pois osioita ja keskittyä puhtaasti ohjelmistokehityksen näkökulmaan tai rajata palvelinympäristön käsittelyä kokonaan pois. Selkeämpi lähestymistapa tähän tutkimusalueeseen voisi olla käytännön ratkaisupaketti, jossa tuotetaan esimerkki kokonaistietoturvallisesta verkkosovelluksesta, ja silti kuitenkin hieman tiukennetaan rajausta. Tämä kuitenkin vaatisi huomattavasti enemmän aikaa ja olisi tullut huomioida paljon varhaisemmassa opiske-

lun vaiheessa. Toteutustehtävän valmistelu parityönä voisi myös jakauttaa laajaa aihepiiriä ja pohjatietojen vaatimuksia, etenkin koska esimerkkitoteutus tulisi vaatimaan enemmän myös palvelinpuolen ympäristöriippuvaista ylläpito-osaamista.

6.3 Itsearviointi

Koen työn onnistuneen erinomaisesti perustuen omaan oppimiseeni ja perehdytykseen tietoturvan eri osa-alueissa, mikä olikin pääasiallinen tavoite opinnäytetyössä. Toisaalta tutkielman aikana on selvästi osoittautunut, että aihealue on hyvin monipuolinen ja laaja, ja riittävän selkeää aiherajausta on vaikea tehdä. Tämä saattaa olla johtanut hieman epäselvään rakenteeseen työssäni. Vaikka aihepiiri osoittautui tämän tutkielman myötä aivan liian laajaksi opinnäytetyötason tutkielman käsittelyyn, katsoin silti ammatillisen kehitykseni kannalta riittävän hyödylliseksi ottaa tämän laajan yleiskatsauksen tietoturvaan ja sen tekniikoihin. Tämän yleiskatsauksen pohjalta voin myöhemmän tarpeen mukaan jo aika hyvin virkistää muistiani tai laajentaa osaamistani nopeasti aukkojen kohdalta, kun tiedän jo yleisesti niiden luonnon ja merkityksen. Uskon myös, että työhöni kiinnostusta osoittaneet kanssaopiskelijat pystyvät tutkielman ja lähdeluettelon perusteella löytämään joitain aiemmin tiedostamattomia, merkityksellisiä aiheita ja tekniikoita oman osaamisensa tueksi huomattavasti vähemmällä vaivalla kuin tekemällä vastaavaa tutkimustyötä itsenäisesti opintojensa ja töidensä ohella.

Merkittävimpinä heikkouksina näen tässä työssä olevan aiheen rajauksen ja kysymystenasettelun ongelmat. Alusta loppuun opinnäytetyön tekeminen tuntui pitkälti käsittelevän työtä itseään metatasolla, työn aiheen ja rakenteen jatkuvaa rajausta ja uudelleenmäärittelyä. Tästäkin syystä sisältö ei aina valitettavasti saanut ansaitsemaansa huomiota ja täsmennystä.

Tutkielman luotettavuus voi vaikuttaa vaihtelevalta, koska aivan kaikkea aihealueen tietoa on vaikea tai mahdoton löytää parhaiten luotettavista tutkimuslähteistä. Tietoturva kuitenkin on alan tekijöiden ja työläisten toimesta jatkuvasti keskusteltu aihealue, eikä sen nopeasta muuttuvuudesta johtuen tule väheksyä työnsä puolesta asiantuntevien henkilöiden blogikirjoituksia ja artikkeleita, joissa he esittelevät omia ratkaisujaan, löytämiään uusia heikkouksia tai koostavat näitä käytännönläheisesti. Myös työssä esitellyt salausmenetelmät ovat vain oletetusti ja toistaiseksi turvallisia. Mitä pidempään

salausmenetelmä, kuten AES tai SHA-tiivistealgoritmit, ovat olleet käytössä, sitä paremmin niiden vahvuudet tunnetaan. Kuitenkin pidetään mahdollisena, että jonain päivänä näihin tai muihin algoritmeihin keksitään suoria matemaattisia ratkaisuja, jolloin näiden luotettavuus ja käyttöarvo menetetään lähes välittömästi. Edelleen tästäkin syystä pidän enemmänkin hyvänä asiana, että tutkimukseen on sisältynyt myös enemmän alan harrastajien ja ammattilaisten blogi- ja artikkelimuotoista keskustelua lähdemateriaalina. Joihinkin osa-alueisiin kuten esimerkiksi sovelluksen käytettävyyteen olisi kyllä hyvä löytää selkeästi käytettävyysnäkökulman tutkimuksia, joista saisi parempia arvioita esimerkiksi siihen kuinka tasapainottaa tietoturvallisuus ja käytettävyys valitessa määrä- ja kestorajoituksia käyttäjätunnusten lukittumiselle vääristä kirjautumisyrityksistä.

LÄHTEET

Alshanetsky, I. 2005. php|architect's Guide to PHP Security. Toronto: Marco Tabini & Associates, Inc.

Apache HTTP Server Project. 2010. Security Tips. [Apache-dokumentaatio]. Luettu 9.12.2010. http://httpd.apache.org/docs/2.0/misc/security_tips.html

Atwood, J. 2007. You're probably storing passwords incorrectly. [Verkkoartikkeli]. Luettu 30.9.2010. http://www.codinghorror.com/blog/2007/09/youre-probably-storing-passwords-incorrectly.html

Benantar, M. 2006. Access Control Systems. Security, Identity Management and Trust Models. New York: Springer Science+Business Media, Inc.

Black, J., Cochran, M. & Highland, T. 2006. A Study of the MD5 Attacks: Insights and Improvements. [Tutkimus]. Luettu 26.10.2010. http://www.cs.colorado.edu/~jrblack/papers/md5e-full.pdf

Bonneau, J. & Preibusch, S. 2010. The password thicket: technical and market failures in human authentication on the web. The Ninth Workshop on the Economics of Information Security. WEIS 2010. Harvard University, Massachusetts.

Buchmann, J. 2004. Introduction to Cryptography. Second Edition. New York: Springer-Verlag New York, Inc.

Butler, E. 2010. Firesheep. [Kehittäjän blogi]. Luettu 7.12.2010. http://codebutler.com/firesheep

Cannon, J. 2004. Privacy. First printing. Boston: Pearson Education, Inc.

Cronkhite, C. & McCullough, J. 2001. Access Denied. The Complete Guide to Protecting Your Business Online. Berkeley: Osborne/McGraw-Hill.

Firesheep Github Wiki. 2010. Handlers. [Github-projektisivu]. Luettu 7.12.2010. https://github.com/codebutler/firesheep/wiki/Handlers

Fleishman, G. 2009. How to obtain and install an SSL/TLS certificate, for free. [Blogiartikkeli]. Julkaistu 20.12.2009. Luettu 10.12.2010. http://arstechnica.com/security/news/2009/12/how-to-get-set-with-a-secure-sertificate-for-free.ars

Fu, K., Sit, E., Smith, K. & Feamster, N. 2001. Dos and Don'ts of Client Authentication on the Web. 10th USENIX Security Symposium 11.-13.8.2001. Washington.

Fuecks, H. 2004. PHP on the Command Line - Part 1. [Blogiartikkeli/opas]. Luettu 19.10.2010. http://articles.sitepoint.com/article/php-command-line-1

Greiner, L. 2008. PHP, JavaScript, Ruby, Perl, Python, and Tcl Today: The State of the Scripting Universe. [verkkoartikkeli]. Julkaistu 29.8.2008. Luettu 18.10.2010. [Ensisijainen lähde ei ole avoimessa levityksessä].

http://www.cio.com/article/446829/PHP_JavaScript_Ruby_Perl_Python_and_Tcl_Today_The_State_of_the_Scripting_Universe

Haapamäki, J. 2010. Joomla!:n tietoturva.Tampereen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma.

Hafner, R. 2009a. How to Create Totally Secure Cookies. [Kehittäjäblogi]. Julkaistu 24.8.2009. Luettu 9.12.2010. http://thinkvitamin.com/code/how-to-create-totally-secure-cookies/

Hafner, R. 2009b. How to Create Bullet-Proof Sessions. [Kehittäjäblogi]. Julkaistu 1.9.2009. Luettu 9.12.2010. http://thinkvitamin.com/code/how-to-create-bulletproof-sessions/

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

Jarmas, T. 2007. Pienyrityksen tietoturva. Tampereen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Tietoverkkopalveluiden suuntautumisvaihtoehto.

Jianxin, Y., Blackwell, A., Anderson, R. & Alastair, G. 2000. The memorability & security of passwords – some empirical results. University of Cambridge. Computer Laboratory. Tech Report.

Järvinen, P. 2003. Salausmenetelmät. 1. painos. Jyväskylä: Docendo Finland Oy.

Linja-Aho, V. 2010. Salasanavuoto paljasti järkyttävän tietoturvaosaamisen? Käytä vahvempia salasanoja! [Verkkoartikkeli Mikro-PC:ssä]. Julkaistu 23.3.2010. Luettu 9.12.2010. http://www.mikropc.net/kaikki_uutiset/article387792.ece

Liu, A., Huang, C. & Gouda, M. 2005. A Secure Cookie Protocol. 14th International Conference on Computer Communications and Networks 17.-19.10.2005. San Diego.

Live HTTP Headers. 2010. mozdev.org: Live HTTP Headers. [Avoimen lähdekoodin projektisivusto]. Luettu 9.12.2010. http://livehttpheaders.mozdev.org/

Microsoft Corporation. 2005. Microsoft Developer Network Library. The STRIDE Threat Model. [Dokumentaatio]. Luettu 6.12.2010. http://msdn.microsoft.com/en-us/library/ee823878%28CS.20%29.aspx

Open Web Application Security Project. 2007. OWASP Top 10. [Tietoturvaraportti]. Luettu 9.12.2010. http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf

Open Web Application Security Project. 2010a. Threat Risk Modeling. [Projekti-wiki]. Luettu 6.12.2010. http://www.owasp.org/index.php/Threat_Risk_Modeling#STRIDE

Open Web Application Security Project. 2010b. Cross-Site Request Forgery (CSRF). [Projekti-wiki]. Luettu 7.12.2010. http://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29

Open Web Application Security Project. 2010c. OWASP Top 10. [Tietoturvaraportti]. Luettu 9.12.2010. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

Oracle Corporation. 2010a. About MySQL. [Verkkosivu]. Luettu 19.10.2010. http://www.mysql.com/about/

Oracle Corporation. 2010b. MySQL Documentation. [Virallinen dokumentaatio]. Luettu 13.10.2010. http://dev.mysql.com/doc/

Oracle Corporation. 2010c. Commercial License for OEMs, ISVs and VARs. [Lisensointi-info]. Luettu 19.10.2010. http://www.mysql.com/about/legal/licensing/oem/

Oracle Corporation. 2010d. FOSS License Exception. [Lisensointi-info]. Luettu 19.10.2010. http://www.mysql.com/about/legal/licensing/foss-exception/

Oracle Corporation. 2010e. MySQL Downloads. [Ohjelmistolataussivu]. Luettu 19.10.2010. http://www.mysql.com/downloads/

Oracle Corporation. 2010f. Choosing Which Version of MySQL to Install. [Dokumentaatiosivu]. Luettu 19.10.2010. http://dev.mysql.com/doc/refman/5.1/en/choosing-version.html

Oracle Corporation. 2010g. SQL Statement Syntax. [Dokumentaatiosivu]. Luettu 21.10.2010. http://dev.mysql.com/doc/refman/5.1/en/sql-syntax.html

Sampajoki, I. & Sihvo, J. 2006. Tietoturvaohjeistus. Case: Vapaa Valinta. Tampereen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma.

Schneier, B. 1996. Applied Cryptography: protocols, algorithms, and source code in C. Second Edition. New York: John Wiley & Sons, Incorporated.

Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D. & de Weger, B. 2008. MD5 considered harmful today: Creating a rogue CA certificate. Päivitetty 13.1.2010. [Tutkimus]. Luettu 26.10.2010. http://www.win.tue.nl/hashclash/rogue-ca/

Stevens, M., Lenstra, A. & de Weger, B. 2007. Vulnerability of software integrity and code signing applications to chosen-prefix collisions for MD5. Päivitetty 1.1.2009. [Tutkimus]. Luettu 26.10.2010. http://www.win.tue.nl/hashclash/SoftIntCodeSign/

Tamper Data. 2010. mozdev.org: Tamper Data. [Avoimen lähdekoodin projektisivusto]. Luettu 9.12.2010. http://tamperdata.mozdev.org/

Teosto r.y. 2010. Haun vastaus. [Verkkosivu]. [2010 Teoston sivuille tehty XSS-hyökkäysdemonstraation URL-osoite]. Luettu 13.12.2010. http://www.teosto.fi/teosto/websivut.nsf/SearchShowDocuments?SearchView&Query=ThePiratebay%20l%F6ytyi%201%20osuma:%3Cbr%20/%3ETeosto%20suosittelee%20thepiratebayta!%3Cbr%20/%3E%3Ca%20href=%22http://www.thepiratebay.org%22%3E%3Cimg%20border=%220%22%20src=%22http://static.thepiratebay.org/img/tpb.jpg%22/%3E%3C/a%3E%3C!--&SearchFuzzy=1&lng=Suomi

The PHP Group. 2010a. MySQL Improved Extension Overview. [PHP:n laajennusdokumentaatio]. Luettu 19.10.2010. http://fi2.php.net/manual/en/mysqli.overview.php

The PHP Group. 2010b. PHP: Error reporting. [PHP:n dokumentaatio]. Luettu 8.12.2010. http://fi2.php.net/manual/en/security.errors.php

The PHP Group. 2010c. PHP: General Information. [Usein Kysytyt Kysymykset -sivu]. Luettu 18.10.2010. http://fi2.php.net/manual/en/faq.general.php

The PHP Group. 2010d. PHP: hash. [PHP:n dokumentaatio]. Luettu 29.11.2010. http://www.php.net/manual/en/function.hash.php

The PHP Group. 2010e. PHP: Hypertext Preprocessor. [Virallinen kotisivu ja dokumentaatio]. Luettu 13.10.2010. http://www.php.net

The PHP Group. 2010f. PHP: Migrating from PHP 5.2.x to PHP 5.3.x. [PHP:n dokumentaatio]. Luettu 9.12.2010. http://fi2.php.net/migration53

The PHP Group. 2010g. PHP: mysqli::autocommit. [PHP-dokumentaatio]. Luettu 10.12.2010. http://fi2.php.net/manual/en/mysqli.autocommit.php

The PHP Group. 2010h. PHP: mysqli::multi_query. [PHP-dokumentaatio]. Luettu 10.12.2010. http://php.net/manual/en/mysqli.multi-query.php

The PHP Group. 2010i. PHP: PHP 5 Changelog. [PHP:n dokumentaatio]. Luettu 9.12.2010. http://www.php.net/ChangeLog-5.php

The PHP Group. 2010j. PHP: Runtime configuration. [PHP:n dokumentaatio]. Luettu 8.12.2010. http://fi2.php.net/manual/en/errorfunc.configuration.php

The PHP Group. 2010k. PHP: Sessions. [PHP:n dokumentaatio]. Luettu 7.11.2010. http://fi2.php.net/manual/en/book.session.php

The PHP Group. 2010l. PHP: setcookie. [PHP:n dokumentaatio]. Luettu 7.11.2010. http://php.net/manual/en/function.setcookie.php

The PHP Group. 2010m. PHP: SQL Injection. [PHP:n dokumentaatio]. Luettu 8.12.2010. http://fi2.php.net/manual/en/security.database.sql-injection.php

The PHP Group. 2010n. PHP: Type Juggling. [PHP:n dokumentaatio]. Luettu 7.11.2010. http://php.net/manual/en/language.types.type-juggling.php

The PHP Group. 2010o. PHP: urlencode. [PHP:n dokumentaatio]. Luettu 7.12.2010. http://www.php.net/manual/en/function.urlencode.php

The PHP Group. 2010p. Using PHP from the command line. [Verkkodokumentaatio]. Luettu 19.10.2010. http://fi2.php.net/manual/en/features.commandline.php

Tuominen, T. 2005. WLAN tietoturva. Tampereen ammattikorkeakoulu. Tietotekniikan koulutusohjelma. Tietoliikennetekniikan suuntautumisvaihtoehto. Opinnäytetyö.

Ullman, L. 2008. PHP 6 and MySQL 5 for Dynamic Web Sites. Berkeley: Peachpit Press.

US-CERT. 2008. Vulnerability Note VU#836068 MD5 vulnerable to collision attacks. [Haavoittuvaisuusjulkaisu]. Luettu 4.11.2010. https://www.kb.cert.org/vuls/id/836068

World Wide Web Consortium. 2010a. Forms in HTML documents. [HTML 4.01-standardin kuvausdokumentti]. Luettu 8.11.2010. http://www.w3.org/TR/html4/interact/forms.html#submit-format

World Wide Web Consortium. 2010b. Appendix B: Perfomance, Implementation and Design Notes. [HTML 4.01-standardin liite]. Luettu 7.12.2010. http://www.w3.org/TR/html4/appendix/notes.html#ampersands-in-uris

Xiaoyun, W., Dengguo, F., Xuejia, L. & Hongbo Y. 2004. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. [Tutkimus]. Luettu 26.10.2010. http://eprint.iacr.org/2004/199.pdf

Zend Technologies. 2010. PHP and Zend Engine – Zend contributions to PHP. [Esittelysivu]. Luettu 19.10.2010. http://www.zend.com/en/community/php