TIETOKANTALÄHTÖISEN PHP-OHJELMOINNIN TIETOTURVAOPTIMOINTI

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulututkinnon opinnäytetyö

Mediatekniikan koulutusohjelma

Riihimäen toimipiste 17.2.2010

Martti Pelttari

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Mediatekniikan koulutusohjelma
Kaartokatu 2
11100 Riihimäki

Työn nimi Tietokantalähtöisen PHP-ohjelmoinnin tietoturvaoptimointi

Tekijä Martti Pelttari

Ohjaava opettaja Kauko Ojanen

Hyväksytty _____._____.20_____

Hyväksyjä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

RIIHIMÄEN TOIMIPISTE
Mediatekniikan Koulutusohjelma

Tekijä Martti Pelttari **Vuosi** 2008

Työn nimi Tietokantalähtöisen PHP-ohjelmoinnin tietoturvaoptimointi

Työn säilytyspaikka HAMK, Riihimäen toimipiste

TIIVISTELMÄ

Toteutin opinnäytetyön Hämeen ammattikorkeakoulun Teknologiateollisuuden koulutus- ja tutkimuskeskuksen palkkalistoilla kesällä 2008. Työn aihe, tietokantalähtöisen PHP-ohjelmoinnin tietoturvaoptimointi lähti liikkeelle kahdesta tietokantavetoisesta projektista, jotka toteutettiin kesän aikana. Tämä työ ei niinkään ole sidoksissa näiden projektien lopputulokseen, vaan tarkoituksena oli kiinnittää erityistä huomiota tietoturvallisuuteen tämänkaltaisissa projekteissa.

Työn teoriaosuuden on tarkoitus olla yleispätevää koskien kaikkia PHP:lla toteutettuja tietokantasovelluksia. Työssä on käsitelty huomionarvoisia seikkoja tietoturvallisen sovellutuksen toteuttamisesta sekä vertailtu erilaisia työvälineitä ja toteuttamistapoja. Varsinainen suoritettu tutkimus alkoi vasta, kun kesälle määritellyt projektit olivat käyttövalmiina.

Suoritin tutkimusosion kokeilemalla erilaisia lähestymistapoja SQL-injektio, XSS ja CSRF -hyökkäyksiin. Tarkoituksena oli löytää konkreettisesti aukkoja toteuttamieni sivustojen tietoturvasta. Sen lisäksi, että työn tekemisen yhteydessä oppi uusia asioita tietoturvallisesta ohjelmoinnista, niin samalla kertyi huomattava määrä ymmärrystä siitä, miksi joitain asioita tehdään aina tietyllä tavalla.

Tämä työ on ennen kaikkea läpileikkaus Web-ohjelmoinnin tietoturvaratkaisuista ja vasta toiseksi ohje nimenomaiseen PHP -ohjelmointiin.

Avainsanat Tietoturva, Internet, HTML, PHP, SQL

Sivut 49 s. + liitteet 2 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Unit of Riihimäki
Name of degree programme: Media Technology

Author Martti Pelttari **Year** 2008

Subject Security Optimization of Database-Driven PHP-Programming

Archives HAMK University of Applied Sciences,
Riihimäki

ABSTRACT

This thesis was created while I was working in the Technological Research and Education Center of HAMK University of Applied Sciences in summer 2008. The Subject of this work, Security Opimization of Database Driven PHP Programming, started from two data-base driven projects that were ongoing during that summer. This work is not tied to the outcomes of these projects, but instead it concentrates on the security of these kinds of projects in general.

The main purpose of the theory of this work is to be valid for the implementations of all database driven applications written with PHP. Different ways of creating secure online applications and some tools and methods to achieve them are handled in this thesis. The real research started after the majority of programming was done and the applications were allready operationals.

The Research was performed by trying out different approaches of SQL injection, XSS- and CSRF- attacks, with the aim of discovering vulnerabilities in the applications' programming codes. Besides valuable information concerning the security of different web systems, a firm understanding of the reasons why some things are always done in certain ways was also obtained

Even though the title of this thesis appears to be only about PHP, the purpose was to create a roadmap of secure web-programming, valid with any programming language.

Keywords Information security, Internet, HTML, PHP, SQL.

Pages 49 p. + appendices 2 p.

SISÄLLYS

TERMIEN SELITYKSET

AJAX
Web 2.0 liittyvä tekniikka, jossa selainpuolella ajettava ohjelmakoodi kutsuu suoraan tietoja tietokannasta tai tallentaa niitä. Mahdollistaa parempaa interaktiota. [6 s.381.].

Apache
Unixissa toimiva Web palvelin ja proxy ohjelma [9 s.33.].

Bottiverkko
Joukko koneita jotka on valjastettu niiden omistajien tietämättä muuhun käyttöön. Suurimmat tavatut bottiverkot ovat olleet yli sadantuhannen koneen laajuisia. Yleensä bottiverkkojen palveluita tarjotaan rikollisille [1 s.88-89.].

CGI
Nimitys tulee sanoista “Common Gateway Interface”. Kyseessä on yleisesti netissä käytetty tekniikka jossa HTML sivulla olevat lomakkeet ja muut syötteet ohjataan erilliseen ohjelmaan joka sitten generoi vastaukseksi tulevat sivut. Käytössä mm. Perl ja Python ohjelmoinnin yhteydessä. [9 s.84–85.]. Myös PHP:tä voidaan ajaa CGI:nä. Erilliset ohjelmat ajetaan silloin käyttäjäkohtaisesti ja tästä saataisiin lisäturvaa palvelinpuolella toisilta käyttäjiltä. Haittapuolena on huomattava tehonmenetys, pahimmillaan 40 %. [4 s.145]

Charset
*Katso **merkkityyppi**.*

CSRF
Cross-site request forgeries lyhennetään yleensä CSRF tai XSRF. Tämä on yleisnimike joukolle erilaisia hyökkäyksiä jotka suoritetaan metodeilla joilla käyttäjän selain saadaan yhdellä sivulla ottamaan yhteyttä toiseen sivustoon ja suorittamaan siellä toimintoja, esimerkiksi Get-metodin syötteillä. [3 s.93-94]

CSS
Casade Style Sheet eli CSS on yleisesti netissä käytetty tyylien muokkauskieli.

Cookie
*Katso **eväste.***

Dynaaminen Web ohjelmointi
Web 2.0 ilmiöön liittyvä termi jolla tarkoitetaan Web sivujen luomista jotka toimivat interaktiivisesti yhteydessä käyttäjien kanssa.

Eväste
Eväste eli Cookie on selaimeen tallennettava muuttuja-arvo-pari. Nettisivuilla olevat ohjelmat voivat sitten ajon aikana pyytää selaimelta evästeitä ja näiden tietojen avulla tunnistaa käyttäjiä, muistaa käyttäjän asetuksia jne.[6 s.271.].

Get-Metodi
Tietojen lähetystapa selaimelta palvelimelle joka perustuu tietojen kirjoittamiseen verkkosivun osoitteen perään muodossa ”?muuttuja=arvo”. Get -metodia ei suositella käytettäväksi palvelimen tietojen muuttamiseksi vaan ainoastaan erilaisten näkymien tms. välittämiseen. [6 s.223-224.].

.htaccess
Apachen kansiokohtainen konfigurointitiedosto.

HTML
HTML on lyhenne sanoista Hypertext Markup Language. Se on merkitsemiskieli jota käytetään nykyään sivujen taittamisen perustana. Alun perin HTML suunniteltiin sivujen sisällön ja ulkoasun erottelemiseksi ja siihen että eri koneilla eri sivut saataisiin näkymään eri tavoilla. [9 s.213.].

Input Encoding
Tällä tarkoitetaan Web-sivulta tulevan syötteen tarkistamista sen saapumisen yhteydessä, ennen syötteen tallentamista. [3 s.400.].

Injektiohyökkäys
*Katso **SQL-injektio**.*

JavaScript
Verkossa yleisesti käytetty scriptikieli, jota ajetaan pääosin käyttäjän selaimella. JavaScript on otettu ensimmäisen kerran käyttöön 1995, jolloin se tuli ensimmäisen kerran Netscape Navigator selaimiin. [6 s.254]

Khalastelu
Khalastelulla eli Phissingillä tarkoitetaan käyttäjätietojen huijaamista hyväuskoisilta käyttäjiltä erilaisten näköissivujen avulla. Yleensä käyttäjiä huijataan näille sivuille sähköpostilinkkien ja foorumilähetysten avulla. Khalastelua kutsutaan joissain yhteyksissä nimellä kalastelu, Petteri Järvinen käytti nimikellä Khalastelu kirjassaan ”Paranna tietoturvaasi” [1 s.273.].

Linux
Linux on Unix-käyttöjärjestelmästä kehitetty vapaan lähdekoodin käyttöjärjestelmä. Linux on nykyään hyvin yleinen käyttöjärjestelmä yksityisillä ja pk-sektorin palvelimilla. [9 s.303-304.].

MD5
MD% eli Message Digest 5 on 1991 kehitetty algoritmi joka laskee 128 bittisen tarkistussumman tekstistä. [9 s.328.]. MD5 tuottaa aina samalla syötteellä saman tarkistussumman, sitä voidaan käyttää niin tiedostojen tarkistussummana kuin salasanojen suojaamisalgoritmina. [6 s.265]

Merkkityyppi
Merkkityyppiä kutsutaan Suomessakin yleensä sen englanninkielisellä nimellä **Charset.** Erilaisilla merkkityypeillä tarkoitetaan yhteyttä eri binäärimerkintöjen ja ruudulla näkyvien merkkien välillä. Merkittävimpiä eroja tulee yleensä kielikohtaisten erityismerkkien kanssa, mutta merkkityyppien vaihtaminen saattaa joskus olla hakkereidenkin työkaluna.

Mod_rewrite
Apachen moduulina asennettu työkalu jolla voidaan muokata osoitekentän sisältöä suhteellisen vapaasti. Se mahdollistaa vaikkapa kentän lopun ohjaamisen GET metodille tai puuttuvan www:n lisäämisen osoitteen alkuun. [11 s. http://httpd.apache.org/docs/2.2/rewrite/]

MySQL
Relaatiotietokantajärjestelmä joka on saatavilla ilmaiseksi. Alun perin Ruotsalaisen MySQL AB:n valmistama järjestelmä. [6 s.37-38]

Olio-ohjelmointi
Olio-ohjelmoinnilla tarkoitetaan ohjelmointia erityisellä rakenteella jossa ohjelma jaetaan osasiin joita kutsutaan kielestä riippuen luokiksi tai olioiksi (class/object). Näitä eriytettyjä ohjelman osasia voidaan sitten ajaa erikseen, niitä voidaan hyödyntää toisissa sovellutuksissa, uusiokäyttää, periyttää ja yksittäistä palasta voidaan ajaa vaikka useita kertoja päällekkäin. [5 s.6-8.].

Open Source
Open Source eli avoin lähdekoodi tarkoitta ohjelmaa jota kuka tahansa saa hyödyntää, levittää ja edelleen kehittää. Lisenssejä näiden ohjelmien takana on useita.

Output Encoding
Output Encoding tarkoittaa Web-sivulta tulevan syötteen tarkistamista sen esittämisen yhteydessä. Puhtaimmillaan syöte ainoastaan tallennetaan kun se saapuu ja kaikki tarkistukset suoritetaan esitettäessä. [3 s.402.].

Palvelin
Lähi- tai tietoverkossa jatkuvasti oleva, yleensä tehokas, tietokone joka vastaa tiettyyn porttiin tuleviin palvelupyyntöihin. Esimerkiksi Web-palvelin vastaa porttiin 80 tai 8080 ja lähettää Web sivuja verkon ylitse käyttäjille jotka niitä pyytävät. [9 s.407.].

PDF
PDF on standardiksi asti noussut Adoben kehittämä dokumenttiformaatti. PDF on lyhenne sanoista Portable Document Format ja tämä nimi kertoo pitkälti mistä on kyse, eli dokumenttiformaatista joka on laitteistoriippumaton ja siirrettävä. [9 s.416.].

PDO
PDO eli PHP Data Object on ollut PHP:ssa mukana versiosta 5.1 lähtien. Se on tietoturvallinen työkalu erilaisten tietokantojen yksinkertaiseen hallintaan. [5 s.157.].

PHP
PHP eli "PHP: Hypertext Preprocessor" on hyvin laajalti käytetty webbisivujen scriptikieli. Kuten nimikin sanoo, on PHP nimenomaan hypertekstin esikäsittelijä. Se on suunniteltu lisäämään HTML sivuille toiminnallisuutta johon ei pelkkä merkitsemiskieli kykene. PHP ohjelmoinnissa komentoja voidaan kirjoittaa mihin tahansa kohtaan sivun HTML koodia. [6 s.12-14.].

Phissing
*Katso **khalastelu.***

Post-metodi
Tämä on informaation välitystapa selaimelta palvelimelle. Tieto lähetetään käyttäjältä piilossa, käytetään yleensä lomakkeen tietojen välittämiseen. [6 s.213.]. Post-metodin käyttöä suositellaan kun, sovellukseen lisätään tai muutetaan tietoja. [6 s.223.].

Serveri
*Katso **Palvelin.***

Sessio
Sessiot ovat nettisivuilla hyödynnetty tekniikka jolla mahdollistetaan kirjautumisen palveluihin. Sessio luodaan kun, käyttäjä kirjautuu palveluun ja tuhotaan kun käyttäjä kirjautuu ulos. Kirjautumisen aikana käyttäjä tunnistetaan yleensä evästeeksi tallennetulla sessio-avaimella. [4 s.113]

Sha1
Tiedon tiivistealgoritmi, joka muodostaa annetusta tiedosta 40 merkin mittaisen tarkistussumman jota ei voida palauttaa alkuperäiseksi tiedoksi [6 s.266.].

SQL
SQL eli Structured Query Language on yleisesti hyödynnetty tietokantojen hallintakieli. Sen avulla suoritetaan erilaisia operaatioita relaatiotietokantoihin. Komentojen rakenne lähtee ajatuksesta mitä, ei miten. [6 s.98.].

SQL-injektio
Tietokantaa vastaan suunnattu hyökkäys jossa hakuavaimen, lisättävän tiedot tai muut nettisivulta tulevan syötteen mukana tai sijasta lähetetään tietokannalle SQL-muotoisia hallintakomentoja. [4 s.73-74]

Type cast
Type cast:illa tarkoitetaan muuttujatyyppiin pakottamista. Tämä on tekniikka jota hyödynnetään usein PHP:ssa sen tehokkuuden ja tietoturvallisuuden vuoksi [4 s.28]

Verkkomato
Verkkomadot ovat virusmaisia tietoverkoissa leviäviä ohjelmia jotka tarttuvat koneesta toiseen. Toisin kuin virukset, madot eivät tartu ohjelmiin, eikä niiden tarttuminen vaadi isäntäohjelman ajamista. [1 s.88.].

Worm
*Katso **Verkkomato**.*

WWW
World-Wide Web, netin näkyvin osa jota jotkut käyttäjät ovat jopa erehtyneet luulemaan koko internetiksi. WWW muodostuu HTML merkitsemiskielellä ja laajennuksella toteutetuista sivuista. WWW:n tekniikat ovat W3-konsortin standardoimia. [9 s.619.].

XML
XML on yleiskäyttöinen merkitsemiskieli jossa ei ole mitään kiinteitä komentoja tai tageja, ainoastaan rakenne. Tällä merkitsemiskielellä on periaatteessa mahdollista kuvata mitä tahansa asioita, mutta se on erityisen hyödyllinen verkkohierarkian kuvaamiseen. [6 s.250.].

XHTML
XHTML on käytännössä XML:än säännöillä kirjoitettua HTML kieltä. Sen käyttäminen luo yhteisiä pelisääntöjä ja poistaa selainkohtaisia poikkeuksia HTML kielestä. Käytännön erot HTML 4.1 ja XHTML 1.0 välillä ovat olemattoman pieniä. [6 s.251.].

XSRF
*Katso **CSRF**.*

XSS
XSS on väännetty sanoista Cross Site Scriping, tähän ratkaisuun on päädytty, koska CSS lyhenne oli jo yleisesti muussa käytössä. XSS on oikeastaan kokoelma erilaisia väärinkäytöksiä ja hyökkäyksiä joissa palvelinpuolen haavoittuvuuksia hyödynnetään välittämään vaarallisia ohjelmakomentoja käyttäjien selaimiin ajettavaksi.[3 s.2-4]

XSS mato
XSS madoilla tarkoitetaan verkkomatoja jotka on toteutettu XSS työkaluilla ja tekniikoilla. [3 s.376-377.].

1 JOHDANTO

Erilaiset dynaamiset Web-palvelut ovat täyttäneet tietoverkot ja niillä on suunnattomia määriä käyttäjiä. Samalla, kun toimijat lisääntyvät ja tietomäärät kasvavat, on harvojen huvista tullut suurten massojen viihdettä. Kun käyttäjämäärät lisääntyvät ja tiedon määrä kasvaa, niin myös rahanarvoista ja muuten kriittistä informaatiota siirtyy verkkoon. Samalla kasvaa kiusaus varmasti monellakin vähän kokeilla jotain automaattista työkalua, tai kokeilla ihan vain arvata kaverin salasanaa. Jokainen tietomurtoa yrittävä ei välttämättä ole tekemässä mitään pahantahtoista vaan ihan vain jännityksen ja huvin vuoksi kokeilee. Huomattavasti synkemmän varjon luovat ammattimaiset väärinkäyttäjät, jotka metsästävät verkoista suorituskykyä ja informaatiota ja muuttavat sitä rahaksi. Ammattimaiset väärinkäytökset näkyvät lähes jokaisen päivittäisessä verkkoasioinnissa bottiverkkojen lähettämän roskapostin muodossa.

Nykyään kannustetaan asioimaan verkossa. Esimerkiksi pankkien palvelut ovat nykyään kaikki kätevästi saatavissa verkossa ja yhä useammat käyttäjät myös hyödyntävät näitä. Pankit ovat ääriesimerkki, koska koko maailmansivun ovat ne olleet esimerkkitapauksia erilaisten ryöstäjien kohteista ja niiden järjestelmissä on kiinnitetty erityistä huomiota tietoturvaan. Mutta useilla muilla sivuilla, joihin on kertynyt huomattavia määriä taloudellisesti merkityksellistä ja muutoin kriittistä informaatiota, näin ei välttämättä ole. Myöskään pelkkä tietoturvan huomioiminen ei yksistään riitä, vaan hyvin hajanaisen kentän eri osa-alueet tulee hallita. Sivuston tietoturva onkin kuin ilmapallo, yksikin reikä riittää koko rakennelman räjäyttämiseen.

1.1 Tutkimuskohteeni

Tämän opinnäytetyön tutkimuskohteena on erilaisten nettipalveluiden toteuttaminen tietoturvallisesti. Olen otsikkotasolla rajannut ohjelmointikielen PHP:hen ja tietokantapuolelta käsittelen pääosin MySQL tietokantoja, mutta PDO:ta koskevat osiot toimivat kaikilla yleisimmillä tietokannoilla [10
http://www.php.net/manual/en/pdo.drivers.php.].

Tietoturvapuolella on pääpaino kahdessa merkittävässä hyökkäystyypissä, Cross Site Scriptingissä(XSS) ja tietokantaa vastaan suunnatuissa injektiohyökkäyksissä (Database Injection).

Työn kirjoittamisen toteutin työskentelyn yhteydessä Riihimäellä, Teknologiateollisuuden KT-keskuksessa. Käsittelen työssäni kahta kesällä 2008 suoritettua projektia, EKES:in tilaaman Kehä V -promootiohankkeen verkkojulkaisujärjestelmän ja päivitystä Näytönpaikka Ry:n elektroniselle portfoliolle.

1.1.1 EKES – Kehä V -promootiohanke

EKES:in tilaama Kehä V promootiohanke jakautui kahteen osaan, eli dynaamiseen verkkojulkaisujärjestelmään ja USB-tikuilla levitettävään markkinointimateriaaliin. Verkkojulkaisujärjestelmä ja sen tietoturvallisuus liittyivät läheisesti työhöni. Kehä V -projekti oli varsinainen verkkojulkaisujärjestelmä, jossa yhdellä käyttäjätunnuksella on mahdollisuus muokata kaikkea sivujen sisältöä. Muokkaaminen tapahtuu verkkoympäristöstä eikä järjestelmän ylläpitäjä tarvitse teknisiä erityistaitoja. Koska ylläpitäjällä ei voida odottaa olevan teknistä tietämystä, niin päädyin ratkaisuun, jossa ylläpitäjällä on hyvin rajoitettu mahdollisuus HTML-muotoiluiden syöttöön ja kaiken ohjelmakoodin syöttö on ehkäisty. Tarkoituksena oli ehkäistä sosiaaliset hyökkäykset, esimerkiksi tilanne, jossa hakkeri lähettää koodinpätkän ja kehottaa laittamaan sen sivuille. Lisäksi tämä minimoi haittoja, jos sivuston hallintatunnukset joutuvat vääriin käsiin. Asiaton tekstisisältö on huomattavasti helpompi huomata kuin sivun lähdekoodiin upotettu käyttäjiä vakoileva ohjelma. Järjestelmään toteutettiin hallintatunnusten lisäksi toinen käyttäjätunnus. Tämän tunnuksen taakse on suojattu alasivuja, joissa on tarkoituksen tarjota lisäinformaatiota neuvotteluryhmälle ja muille Kehä V -projektin aktiivisille toimijoille. Tämä tunnus ei kuitenkaan sisällä minkäänlaista interaktiota.

Varsinaisen julkaisujärjestelmän rakensin mahdollisimman yleispäteväksi, jokainen sivu tulee tietokannasta ja sivujen päivittämiseksi riittää tietokantamerkinnän lisääminen. Sivuilla on valmis paikka jokaiselle elementille. Elementit lukitaan tietokannasta ja näin voidaan eritellä eri sivuille käyttötarkoituksia. Samaa julkaisujärjestelmää voisi minimaalisin muutoksin käyttää minkä tahansa muun sivuston moottorina. Tietoturvan kannalta tämän järjestelmän haasteellisin osa on tietokannan ja nettisivujen yhteenliittyminen.

1.1.2 Näytönpaikka Ry:n elektronisen portfolion päivitys

Olin toteuttamassa Näytönpaikan elektronista portfoliota kesällä 2007 yhteistyössä Matti Niemelän kanssa. Kesällä 2008 suoritin portfoliolle päivityksen, jonka rakenne on nähtävissä liitteestä 1. Varsinaisten päivitettyjen osien lisäksi kävin läpi yksityiskohtaisesti tietoturvan kannalta koko järjestelmän. Tämänkaltainen palvelu, jossa on luottamuksellista tietoa käyttäjistä ja erilaisia sosiaalipuolen ammattilaisia, on erityisen kriittinen, jos väärät tahot onnistuvat sen murtamaan. Kaiken lisäksi on palvelussa kerätty tieto rajoitettu siten, että käyttäjän on itse annettava muille lupa nähdä tiedot ihan lainsäädännöllisistä syistä. Käyttäjä saattaa jakaa hyvin henkilökohtaista tietoa, koska järjestelmän luonteesta johtuen sen voi ainoastaan nähdä vaitiolovelvollinen sosiaalityöntekijä. Tämänkaltainen tieto voi kiinnostaa myös väärinkäyttäjiä.

2 UHKAKUVAT

Verkossa tapahtuvat väärinkäytökset eivät ole perinteisesti ”oikeiden” rikollisten aikaansaannoksia. Suurin osa vain kokeilee omia rajojaan tai käyttää valmiita työkaluja. Väärinkäyttäjä kokee yleensä tekevänsä jotain väärää, juuri sen verran, että toiminta tuntuu jännittävältä, mutta oikeaksi rikokseksi tekoa ei useinkaan edes mielletä. Tätä voidaan verrata siihen, että keskimääräinen tietokoneenkäyttäjä tietää musiikin tai tietokoneohjelmien imuroinin netistä olevan laitonta. Tästä huolimatta näitä rikkeitä harrastelevat ihmiset, jotka eivät varastaisi esimerkiksi levyä levykaupasta. [2 s.32–33]

Esimerkiksi hyvin tuhoisa XSS-mato ”Samy Worm”, joka kaatoi MySpace palvelun, oli alun perin tarkoitettu vitsiksi ja sen aikaansaamat tuhot tulivat ohjelmoijalle täydellisenä yllätyksenä. [3 s.390]

Verkko on nykyään osa 800 miljoonan ihmisen jokapäiväistä elämää. Sähköpostit luetaan päivittäin, erilaiset kuvagalleriasivustot ja muut interaktiiviset kommunikaatiopalvelut ovat hyvin suosittuja. Koko ajan lisääntyvä verkkosisällön määrä kutsuu verkkoon lisää käyttäjiä ja käyttäjämäärien kasvu taas lisää sisällön tarvetta. Usein hyödyllinen tai mielenkiintoinen informaatio saattaa olla vaarallista väärissä käsissä. Lisääntynyt informaationmäärä ja huono tietoturva motivoivat entisestään väärinkäyttäjiä. Verkkomurroilla saattaa olla sotkettua vieraskirjaa merkittävämpiä seurauksia. Nykyään kyse on isoista rahoista, ja tästä johtuen väärinkäytöksistäkin on tulossa ammattimaisempia. Suuntaus on ollut aina vaan merkittävimpiin asioihin, joita kontrolloidaan verkkopalveluiden kautta, kauhukuvana vaikkapa virranjakeluverkko tai pato. Huolimaton tietoturva voi tulevaisuudessa olla elämän- ja kuoleman kysymys. [3 s.3-4.].

2.1 Uhkat verkossa

Cross Site Scripting ei ole uusi ilmiö, ensimmäiset XSS-hyökkäyksen tunnusmenrkit täyttävät väärinkäytökset ovat jo vuodelta 1996. Selainpuolen haavoittuvuuksia hyödyntävien palvelinpuolen hyökkäysten lokeroiminen yhteisen nimen Cross Site Scripting eli XSS alle on peräisin vuodelta 2000, mutta julkiseen tietoisuuteen se ei ole vieläkään noussut kunnolla. Lokakuussa 2005 tietoturva-alan ammattilaiset kiinnostuivat asiasta todella, kun ensimmäinen tehokas XSS-mato kaatoi MySpace- sivuston yhdessä yössä. Keskimäärin kahdeksan sivustoa kymmenestä on haavoittuvainen XSS-hyökkäyksille, joukossa useita palveluita, jotka sisältävät kriittistä informaatiota ja lisää tulee jatkuvasti. [3 s.3-4]

2.2 Haavoittuvuuksien hyödyntäminen

Vaikka netissä on paljon informaatiota, niin taloudellista merkityksellistä tietoa on vähän. Amerikkalaisilla sivustoilla suojataan erityisen tarkasti luottokorttien numerot ja sosiaaliturvatunnukset. Luottokortin numerolla kun voi suorittaa ostoja verkosta ja käyttäjien rekisteröityminen kaikkiin virallisiin instansseihin taas hoituu sosiaaliturvatunnuksilla. Ääriesimerkkinä näillä informaatioilla olisi mahdollista varastaa ihmisen elämä. [3 s.228.]. Web-palveluiden yleistymisen myötä yhä useampiin järjestelmiin lisätään Web-liittymiä. Jo nykyään olisi mahdollista ohjata virranjakelua, patojen toimintaa ja maksupalveluita verkon ylitse. Tällaisen järjestelmän joutuminen väärän tahon hallintaan voi johtaa todella katastrofaalisiin seurauksiin. [3 s.4]

2.3 SQL vulnerabilities

Tietokannat näyttelevät huomattavaa roolia erilaisissa verkkopalveluissa, yleensä tietokantoja hyödynnetään erilaisten tunnistuksien, hakutoimintojen ja muun dynaamisen sisällön kanssa. Kaikkiin näihin liittyy läheisesti käyttäjältä tuleva syöte. Tietokantahakuja ja tarkistuksia ajetaan niin, että käyttäjältä tulevaa syötettä sijoitetaan sopivaan osaan SQL-kyselyä ja sitten kysely ajetaan. Mikäli tietoturva ei ole kunnossa, niin voi käyttäjän syöte sisältää SQL-komennon osia tai kokonaisia komentoja ja tällä tavoin voi väärinkäyttäjä aikaansaada huomattavaa tuhoa tai muita ei toivottuja vaikutuksia. [4 s.73.].
Kaikki tietokantaan menevät syötteet tulee joko varmentaa ja muutoin estää mahdollisten SQL-komentojen ajaminen, esimerkiksi Prepared Statement –metodilla (kts. PDO).

2.4 Cross Site Scripting

Cross Site Scripting lyhennetään yleensä XSS ja sillä tarkoitetaan hyökkäystyyppiä, jossa käyttäjien selaimet ajavat paitsi haluamansa kohdesivun koodia, niin myös vihamielisen hyökkääjän koodia. Tämän aikaansaamiseksi on olemassa useita erilaisia tapoja. [3 s.3-4] Pääpaino tässä työssä on dynaamisten verkkosivujen haavoittuvuuksissa, joilla on mahdollista ajaa erilaisia haitallisia ohjelmakoodin osia. Myös muunlaisia hyökkäyksiä on olemassa. Sivustosta riippumattomia (ja siten hyvin hankalia torjua) ovat sosiaaliset hyökkäykset, joissa käyttäjä huijataan tulemaan sivulle toisen sivun kautta. Näin selaimessa pyörii jo sivulle tultaessa haitallista koodia. CSRF-tyypiset hyökkäykset taas esiintyvät käyttäjänä ja lähettävät sivuille jotain haitallista, yleensä peittääkseen hyökkääjän identiteetin tai hyödyntääkseen käyttäjän kirjautumista. Sivuston suunnittelussa on hyvä huomioida selainpuolen haavoittuvuudet. Näistä tekniikoista on käytetty lukuisia variaatioita, pahimmillaan hyvin tuhoisin seurauksin. Tuhoisin XSS-mato tähän asti on ollut nk. Samy-mato. Vaarattomaksi pilaksi tarkoitettu koodinpätkä infektoi vuorokaudessa yli miljoona käyttäjää ja johti MySpacen väliaikaiseen sulkemiseen. [3 s.386-390.].

2.5 Julkisten koneiden ongelma

Sivujen suunnittelussa on otettava huomioon se, että niitä saatetaan käyttää julkiselta koneelta. Mahdollisella väärinkäyttäjällä voi siis olla fyysinen pääsy koneelle. Selaimen välimuisti on ensimmäinen ongelma, modernit selaimet säilövät varsin tehokkaasti kaikki sivut, joilla käyttäjä vierailee, olenkin päätynyt käyttäjäläheiseen tapaan, jossa suositellaan uloskirjautumisen yhteydessä sulkemaan selain. Tämä on varmuudella tehokas tapa päästä eroon välimuistissa kummittelevista sivuista, joille seuraava käyttäjä pääsee selaamalla selaimen historia-listaa. [1 s.242.].

Koneelle jää helposti evästeitä, selain muistaa evästeet vanhenemisaikaan asti, eikä evästeitä ole kovalevyllä mitenkään suojattu. Lisäksi evästeet voidaan kaapata myös linjalta niiden lähetyksien yhteydessä. Jos käyttäjä ei kirjaudu ulos, niin selaimelle voi jäädä aktiivinen sessio palveluun eikä palvelu erota käyttäjän vaihtuneen. Näitä ongelmia korjaamaan on lukuisia tekniikoita. Erilaiset sessiot pitää aina laittaa vanhenemaan toiminnan loppumisen jälkeen, evästeiden vaihtaminen tekee evästeiden kaappaamisesta hyödytöntä. Mutta jos varokeinoja ei suorita, voi ammattitaidotonkin hakkeri kaapata toisten tilejä julkisilla koneilla. [4 s.69.].

Hienotkaan tekniikat eivät auta, jos julkiseen koneeseen tai muuhun tietoturvattomaan järjestelmään on päästy ennalta asentamaan ohjelmia, jotka tarkkailevat käyttäjän toimintaa, vaikkapa kaappaamalla kaiken, mitä näppäimistöltä kirjoitetaan. [1 s.242.].

3 TURVALLINEN OHJELMOINTI

Nykyiset nettisivut ovat täynnä erilaisia tekniikoita, ja näistä tekniikoista suurin osa perustuu erilaisten ohjelmakoodien ajamiseen käyttäjän selaimessa tai sen yhteyteen asennettujen ohjelmien tai liitännäisten kautta. Nykyaikaiset selaimet tukevat JavaScriptiä ja CSS varsinaisen HTML:n lisäksi. Lisäksi on enemmän sääntö kuin poikkeus, että jokaisesta selaimesta löytyy Flash-laajennus, mediaselain, joka tukee MP3:sta ja erilaisia videoformaatteja.

3.1 Syötteiden estäminen ja rajoittaminen

Nykyään kun puhutaan internetistä, niin tarkoitetaan usein WWW-verkkoa. Tämä HTML-muotoiseksi tiedonesityskanavaksi tarkoitettu ja sanomalehtimäiseksi suunniteltu alusta on kehittynyt koko ajan interaktiivisempaan suuntaan. Nykyään HTML-ympäristössä on paljon kaksisuuntaista liikennettä ja huomattava osa käyttöarvosta tulee juurikin tästä interaktiosta. Interaktiiviset palvelut hyödyntävät usein tietokantoja, on kyse sitten hakupalveluista tai erilaisista wiki- tms. tietovarastoista.

Web-puolen tietoturvaongelmat voidaan jakaa karkeasti kahteen kategoriaan. Ensimmäinen, perinteinen kategoria, on verkkoon tarkoituksella tehtyjä haitallisia sivuja, näitä ovat erilaiset khalastelu-sivut ja erilaisia haitallisia ohjelmakoodeja sisältävät sivut. Epäilyttävälle sivulle käyttäjät harvoin joutuvat ”vahingossa”. Linkkejä epäilyttäville sivuille lähetellään sähköpostitse ja haitallisille sivuille yritetään huijata ihmisiä lupailemalla halpoja hintoja, palkintoja tai pornografiaa. [1 s.51-52.]. Keskimäärin käyttäjä on kuitenkin varuillaan, vaikka kyseisille sivuille eksyisikin, eikä luottavaisin mielin luovuta tietojaan eikä varsinkaan rahojaan. Khalastelu sivustot ovat samalla tavoin omia sivujaan, mutta ne yritetään esittää osana käyttäjän tuntemaa sivustoa ja niillä yleensä yritetään kaapata käyttäjän käyttäjätunnus-salasanapareja. [1 s. 279-280.].

Toinen, uudempi kategoria perustuu hyvää tarkoittaville sivuille jääneisiin haavoittuvuuksiin. Erityisen ongelmallista tästä tekee se, että keskimäärin 8/10 sivustosta sisältää haavoittuvuuksia, näiden joukossa huomattavan käyttäjämäärän omaavia suosittuja sivuja ja virallisten tahojen omaamia sivuja, joille ihmiset antavat auliisti hyvinkin luottamuksellista tietoa. Näitä haavoittuvuuksia hyödyntäviä väärinkäytöksiä on äärettömän hankalaa huomata ajoissa, mutta ne saattavat johtaa vaikkapa käyttäjien tunnustautumistietojen joutumiseen vääriin käsiin tai mahdollistaa verkossa leviävän madon räjähdysmäisen leviämisen. Web-sivujen haavoittuvuuksien todellinen potentiaali on alkanut konkretisoitua vasta viimeaikoina. Tuhoisimpia lienevät MySpacen kaatanut verkkomato ja erilaiset karanneet salasanalistat. [3 s.4-5, 11.].

Luotettujen sivujen käyttäminen väärinkäytösten välikappaleena perustuu kahteen asiaan. Ensinnäkin selainpuolen haavoittuvuuksiin, nettisivujen käytettävyyttä ja ominaisuuksia varten on selaimiin laajentunut huomattava määrä erilaista toiminnallisuutta ja usein tietoturva laahaa pahasti perässä. Kaikkien selainpuolen haavoittuvuuksien poistaminen olisi periaatteessa mahdollista, mutta samalla suunnaton määrä hyödyllisiä ominaisuuksia lopettaisi toimintansa. Useat haavoittuvuudet kun ovat pohjimmiltaan väärinkäytettyjä ominaisuuksia. Toinen osa tätä vyyhteä on erilaisten verkkopalveluiden haavoittuvuuksien hyödyntäminen. Erilaiset interaktiiviset palvelut antavat käyttäjien lisätä sivuille tietoa. Väärinkäytökset perustuvat siihen, että odotetun tiedon sijasta ja sen lisäksi lähetetään ohjelmakomentoja. Mikäli verkkopalvelussa ei ole huolehdittu syötteiden tarkistamisesta erilaisten komentojen varalta, niin selain ei erottele varsinaisen sivun komentoja ja käyttäjien lisäämiä, mahdollisesti haitallisia komentoja. [3 s.396-397.].

3.2 PHP- ohjelmointi

PHP eli “PHP: Hypertext Preprocessor” on hyvin laajalti käytetty scriptikieli Web-palveluiden toteutuksessa. Oman kokemukseni perusteella käytännössä jokaisella kaupallisella Web-palvelimella pyörii jokin PHP-komentotulkin versio. PHP:n sivujen virallisen statistiikan mukaan huhtikuussa 2007 oli verkossa lähes 21 miljoonaa domainia PHP:n varassa [10 s. http://www.php.net/usage.php.].

Lisäksi PHP oli Tioben tilastojen mukaan maailman viidenneksi suosituin ohjelmointikieli elokuun 2008 tilaston mukaan
[13 s.http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.].

Kuten nimikin sanoo, on PHP nimenomaan hypertekstin esikäsittelijä. Se on suunniteltu lisäämään HTML-sivuille toiminnallisuutta johon ei pelkkä merkitsemiskieli kykene. Perinteisiä CGI-ohjelmia tehtiin siten, että sivulla esitettävä HTML luotiin CGI-ohjelman komennoilla. PHP-ohjelmoinnissa komentoja voidaan kirjoittaa mihin tahansa kohtaan sivun HTML-koodia. Suorituksen yhteydessä PHP-tiedoston käy läpi PHP:n komentotulkki, kaikki PHP:lle osoitetut kohdat (merkitty yleensä <?php ja ?> tagien väliin) suoritetaan ja lopputulos lähetetään eteenpäin, yleensä selaimelle. [6 s.12-14.].

PHP on suunniteltu vain skriptikieleksi. Esimerkiksi olio-ohjelmointi tuli PHP:hen varsin myöhään nykyisessä muodossaan, ja varsin suuren vastustuksen saattelemana. PHP:n alkuperäinen idea kun on mahdollistaa kevyitä muutoksia HTML-elementtien väliin, olio-ohjelmointi taas mielletään raskaiden sovellusten kehityksessä tarvittavaksi ympäristöksi, josta saadaan hyötyä, kun kehittäjäryhmä on suuri. [5 s.2-3.].

3.2.1 PHP ja ongelmallisten syötteiden estäminen

Yleisesti PHP-koodia ei voida syöttää suoraan sivuille, mutta kaikkia muita mahdollisia haavoittuvuuksia on mahdollista hyödyntää. On kyse sitten JavaScriptistä tai sivulle rakennettujen PHP-ominaisuuksien väärinkäyttämisestä. Suurin osa haavoittuvuuksista sivuilla johtuu nimenomaan huolimattomasta syötteiden tarkistamisesta. On ehdottoman tärkeää, että kaikki käyttäjiltä tuleva sisältö tarkistetaan. Kun erilaisia haavoittuvuuksia on alkanut ilmetä ja verkkosivulla olevat palveluita ja arvokasta informaatiota on kokoajan tullut lisää, niin kukaan ei enää voi vähätellä syötteiden tarkistamisen merkitystä. Kyse ei siis ole enää vain sekavista vieraskirjoista, vaan varastetuista pankkitunnuksista, henkilötiedoista ja kaatuneista miljardibisneksistä. [3 s. 4-6; 9 s.28.].
Mikäli syötteelle on rajattu jokin tietotyyppi, niin kaikki käyttäjän lähettämä voidaan pakottaa tiettyyn formaattiin. Numeraalisissa syötteissä tämä on erityisen tehokasta ja kätevää, numeraalinen syöte ei sisällä koodeja eikä tämän formaattiin pakottamisen (type cast) yhteydessä suoriteta yhtään turhaa tarkistusta. [4 s.28-29.].

3.2.1.1 Toteuttamistapoja

Useat tietoturva-alan ammattilaisetkin erehtyvät joskus luulemaan, että poistamalla hakasulut ja lainausmerkit (eli merkit <, >, ”, ’) kaikesta syötetystä sisällöstä saadaan sivuista tietoturvallinen. Valitettavasti tämä ei kuitenkaan ole koko totuus. Erilaisilla kiertoteillä on mahdollista ohittaa tavanomaiset merkkien eliminointitavat. Esimerkkinä näistä on haavoittuvuus, jonka Kurt Huwing oli löytänyt Standfordin yliopiston sivujen suojauksesta. Normaali korvausmetodi ei huomioi eri merkkityyppien vastaavia merkkejä, mutta selain huomioi. Tästä johtuen UTF-8 rakennettu suojaus päästi sisälle UTF-7 rakennetun koodin. [3 s.154-156]

Mikäli annettu syöte on tarkoitus tallentaa tietokantaan, niin tulee huolehtia sekä HTML ja javascript komennoista että tietokannalle vaarallisista syötteistä, eli ehkäistä tietokantaa vastaan suunnattua injektiohyökkäyksiä. Palaan tähän tietokantaa käsittelevässä osiossa tarkemmin.

3.2.2 Tietojen käsittely ja tallentaminen

Dynaaminen sisältö ei muodostu pelkästä tekstistä, vaan lukuisista erilaisista mediaelementeistä. Selaimet käsittelevät sellaisenaan vain kuvaa ja tekstiä, muut mediaelementit on sitten toteutettu erilaisilla liitännäisillä. Yleisiä liitännäisiä ovat erilaiset mediasoittimet (Quicktime, Media Player) sekä flash:in kaltaiset selainlaajennukset. [3 s 97-98]
Yleisimpiä dynaamisten sivujen ominaisuuksia on paitsi tekstin, niin myös kuvien dynaaminen syöttö, tämän ominaisuuden toteuttamiseen on olemassa kaksi erilaista metodia. Yleisesti hyväksytympi on kuvan lähettäminen palvelimelle. Toinen vaihtoehto on kuvan linkittäminen sivuille toiselta palvelimelta. Kuvan linkittäminen on ongelmallista, kun kuva haetaan toiselta palvelimelta, niin se rasittaa se toisen palvelimen kaistaa. Li-

säksi linkitetty kuva ei ole meidän hallinnassamme ja se saattaa poistua tai vaihtua ilman, että saamme mitään ennakkovaroitusta.
Tietoturvan kannalta sekä linkitetyn että lähetetyn kuvan kohdalla on omat ongelmansa.
Lähetettyä kuvaa pidetään yleensä hyvin turvallisena, mutta Internet Explorer -selaimessa on mahdollista virittää kuvatiedosto, joka suorittaa JavaScript-koodia. Tämän takia kuvan sisältö on aina tarkistettava kunnolla. [3 s.121-122]
Linkitetyn kuvan sisältöä on hankalampi valvoa ja pahimmillaan voi linkki kuvaan ohjautua uudelleen suoraan haitalliseen ohjelmakoodiin tai suorittaa CSRF-tyyppisiä hyökkäyksiä.[3 s.96]

3.2.2.1 Lomakkeiden käsittelystä

Suojaamaton lomake on vaarallinen, mikäli lomakkeen lähettämistä ei ole suojattu. Lomakkeen tietoja muistuttava tietopaketti tulla miltä tahansa koneelta, ja palvelin tulkitsee sen lomakkeen lähettämäksi. Tältä voi lähinnä suojautua hyödyntämällä koodattua aikaleimaa lomakkeessa, joka varmennetaan tietokantaan ja jota verrataan syötteen kanssa [3 s.93-97.]. Session käyttäminen lähettäjän tunnistamiseen ratkaisee luonnollisesti ongelman.

3.2.2.2 Metodit, joita ei tulisi käyttää

PHP:n tietoturvallisuutta parantamaan on kehitetty sisäänrakennettu ominaisuus nimeltään Magic Quotes. Ideana on, että kaikki GET, POST ja COOKIE -syötteet käsitellään automaattisesti, lisäämällä kenoviiva (\) ennen vaarallisia merkkejä (tässä tapauksessa ', ", \ ja NUL eli \0). Valitettavasti Magic Quotes saa aikaan vähintään yhtä paljon uusia ongelmia kuin se korjaa vanhoja, se hankaloittaa tehokkaampien tietoturvaratkaisuiden toteuttamista, eikä se edes suojaa kaikilta hyökkäystavoilta. Sen käyttöä ei voi suositella. Kaikki magic quote -toiminnot vanhenevat lopullisesti PHP:n versiossa 6.0.0 ja ne tullaan poistamaan kokonaan. [10 s. http://www.php.net/manual/en/security.magicquotes.php; 9 s.43]

Toinen yhtä merkittävä ongelma magic quotesin kanssa on se, että palvelinpuolen päivitykset saattavat muuttaa sen tilaa. Jos sovellus on rakennettu hyödyntämällä magic quotesia, niin voi sovelluksen tietoturvaratkaisu kadota kuin taikaiskusta.

Alunperin on PHP:ssa kaikkien syötteiden sisäänvientitapana ollut register globals -mekanismi. Perusidea tässä on yksinkertaisen nerokas, kaikki muuttujat, jotka tulevat selaimelta, ovat suoraan käytössä PHP ohjelmassa. Valitettavasti tämä ominaisuus on hyvin ongelmallinen tietoturvan kannalta. Ensimmäinen ongelma on konflikti eri metodien prioriteettien kanssa. PHP:n asetustiedostosta php.ini:stä on säädettävissä prioriteetti järjestys GET, POST ja COOKIE-syötteille. Mikäli sama muuttuja ilmenee enemmän kuin yhdessä metodissa, niin prioriteeteissa korkein jää voimaan.

Tämä toimisi periaatteessa hienosti, jos rakennettu järjestelmä on tarpeeksi yksinkertainen eikä eri asioita varten ole eri requesteja. [4 s.22.].

Tämän toteuttamistavan ongelmallisuutta voidaan havainnollistaa esimerkillä, otetaan tilanne, jossa kirjautuneen käyttäjän sessiota hallitaan evästeiden avulla, joten evästeiden pitää olla korkeimmassa prioriteettiluokassa, mutta kaikki syötteet toteutetaan POST-metodilla. Tästä seuraa, että sopiva eväste korvaa vastaavan lomakkeen arvon. Erityisen ongelmallista tästä saattaa muodostua, jos väärinkäyttäjä pääsee infektoimaan lukuisten käyttäjien selainten evästeosiota haluamallaan syötteellä ja näin käyttämään järjestelmää ja kaikkien käyttäjien selaimia omiin tarkoituksiinsa.

Toinen ongelma syntyy paikallisista muuttujista, joita hyödynnetään PHP:n ajon aikana, mikä tahansa muuttuja on mahdollista alustaa uudella arvolla yksinkertaisesti GET-metodin avulla, tämä koskee myös taulukoita. Tämä ongelma on ohitettavissa sillä, että jokainen muuttuja alustetaan huolellisesti aina, kun ohjelmakoodia ajetaan. Yksikin unohtunut apumuuttuja voi aikaansaada erittäin vaikeasti havaittavan tietoturva-aukon, joten tämän metodin hyödyntämistä ei suositella. [4 s.24-25]

Näitä ongelmia vastaan kehitettiin ensin tekniikka, HTTP_*_VARS. Tämä oli vain hyvin työläs ja hankala tapa suorittaa yksinkertaista asiaa. [4 s.23.].

Tämä kaikki ongelmallisuus on ratkaistu jo PHP:n versiossa 4.1, mutta valitettavasti vanhat tavat jäävät helposti elämään. PHP:n versiosta 4.2 on register globals oletuksena kytketty pois käytöstä, eikä sen käyttöä suositella kenellekään. Ratkaisevaa ominaisuutta kutsutaan nimellä ”Superglobals”. Siinä on ongelmallisempia osia, kuten $_REQUEST, jolla haetaan yhdistelmä GET, POST ja COOKIE -syötteistä. Kuten muutkin yhdistelmämetodit, tämäkin mahdollistaa tarkoituksettoman syötteiden ylikirjoittamisen käyttäjän selaimen toimesta. [4 s.25-27]

Superglobals-tekniikalla jokainen syötetyyppi on mahdollista hakea erikseen ja kaikkiin syötteisiin voi viitata muuttujan nimellä. Muuttujat joita ei pyydetä erikseen jäävät huomiotta, eivätkä siten voi aiheuttaa ongelmia. Huomattava ominaisuus superglobalsseissa on se, että ne ovat globaaleja muuttujia. Ne ovat olemassa kaikkialla sovelluksessa, myös erillisissä olioissa.
[10 s.http://www.php.net/manual/en/language.variables.superglobals.php; 9 s.25-27]

Sivun ohjauksessa käyttäjältä tulevat syötteet tulevat yleensä GET tai POST-metodin kautta. [6 s.223.].

GET-metodilla välitetty syöte on luettavissa $_GET superglobaalin kautta. Kaikki GET-syötteet ovat käyttäjälle näkyvissä, ne välitetään osoitepalkin kautta. Etuina tästä on linkkien kopioitavuus ja se, että niitä on helppo käyttää vaikkapa sivuston ohjauskomentojen tai hakujen välittämiseen. Haittana on, että selaimet tallentavat osoitepalkin informaatiota ja siten

myös GET-metodilla välitettyä dataa. Osoitepalkissa oleva osoite kaikkine GET-syötteineen kasvaa äkkiä valtavaksi, jolloin se ei ole selkeän näköinen, eikä sitä ole helppo muistaa. Väärinkäyttäjät voivat hyödyntää GET:illä annettuja ohjauksia omiin tarkoituksiinsa, vaikkapa lähettelemällä osoitteita foorumeille, joiden perässä olevat GET-muuttujat on räätälöity heidän omiin tarkoitusperiinsä, esimerkiksi CSRF-hyökkäykseen. [6 s.223; 4 s.93-94.].

POST-metodilla välitetyt syötteet ovat pääosin luettavissa $_POST superglobaalin avulla, poikkeuksena tästä ovat välitetyt tiedostot. Vaikka ne välitetään POST-metodin kautta, niin niiden lukeminen tapahtuu $_FILES superglobaalilla. Tämäntyyppisen syötteen käyttäminen erilaisten lomakkeiden välittämiseen on suositeltu tapa. Kyseiset syötteet eivät näy osoitepalkissa eikä niitä voi kopioida, näitä syötteitä ei voi liittää linkkiin eikä siten väärinkäyttää samalla tavoin kuin GET-syötteitä. POST:illa välitetyissä syötteissä on omat ongelmansa, kuten välimuisti. Kun selaimen eteen- ja taaksepäin-toimintoja käytetään, niin nämä tiedot eivät säily automaattisesti. Käyttäjät, joilla on tapana hyödyntää näitä toimintoja, kohtaavat helposti ongelmallisia tilanteita, modernit selaimet kysyvätkin aina haluaako käyttäjä, että samat tiedot lähetetään uudestaan. Toinen ongelma seuraa siitä, ettei POST:in informaatiota voi lisätä linkkiin. Suurin osa verkossa välitetyistä linkeistä kun ei ole väärinkäytöksiä, vaikkapa tilanne, jossa ihmiset lähettelevät hauskoja linkkejä ystäville nähtäväksi. [6 s.223.].

Selain välittää myös käyttäjästä riippumattomia syötteitä, erilaisia tietoja selaimesta ja ympäristöstä sekä tallennettuja evästeitä. Näihin tietoihin pääsee käsiksi $_COOKIE ja $_SERVER -superglobaalien avulla. On hyvä huomata, ettei mitään selaimesta tulevaa syötettä voida automaattisesti pitää turvallisena. [4 s.66.].

3.2.3 Syötteiden tarkistaminen

PHPssa on hyvin tehokas sisäänrakennettu htmlentities –komento, joka korvaa kaikki erityismerkit vastaavilla html-koodatuilla merkeillä (muoto &xxx;). Merkit siis näkyvät sivuilla oikein mutta niillä ei ole mahdollista suorittaa komentoja.
[10 s. http://www.php.net/manual/en/function.htmlentities.php; 9 s.54-56].]. Lisäksi on olemassa htmlspecialchars, joka korvaa vain koodin syöttämiseen tarvittavat merkit (heittomerkit, lainausmerkit, hakasulut sekä &-merkit) em. html-koodauksella. [10 s.
http://www.php.net/manual/en/function.htmlspecialchars.php.].

PHP:n sisäänrakennetut suojausfunktiot ovat tehokkaita, mutta joissain tilanteissa omien suojafunktioiden rakentaminen voi olla hyödyllisempää. Tähän tarkoitukseen voi käyttää työkaluna joko PHP:n string muutujan käsittelijöitä, eli str_replace funktiota, jolla voi korvata merkkejä ja merkkijonoja string tyyppisistä merkkijono muuttujista.
[10 s.http://www.php.net/manual/en/function.str-replace.php.]. Monimutkaisempia operaatiota voi suorittaa regular expression –funktioilla joita

PHP:ssa on kaksi eri kantaa, ereg ja preg. Ensimmäinen on POSIX-pohjaisia ja jälkimmäiset Perl-pohjaisia. POSIX-pohjaisia ereg funktioita ei suositella tietoturvatarkistuksiin, koska ne eivät tarkista binääritasolla kaikkia merkkejä. Lopputuloksena on, että sopiva merkkityyppi saattaa ohittaa tarkistusalgoritmin. Perl-pohjainen preg on suositellumpi erilaisten suojausten työkaluna. [10 s.http://www.php.net/manual/en/book.pcre.php s. http://www.php.net/manual/en/book.regex.php.].

Toinen hyvin mielenkiintoinen kysymys on syötteiden käsittelemisen ajankohta. Tähän on olemassa kaksi lähestymistapaa, joko syötteen saapuessa palvelimelle(Input Encoding) tai syötteen esittämisen yhteydessä(Output Encoding). Molemmissa on omat etunsa ja haittansa. [3 s.396.].

Input Encodingissa eli sisään tulevan syötteen tarkistamisessa on paljon etuja. Tietokantaa hyödynnettäessä on huolehdittava tietokantaan menevistä syötteistä ennen tallennusta. Molemmat tarkistukset voi siis tehdä kerralla. PDO, jota käsittelen myöhemmin, luo tähän oman poikkeustensa. Input Encoding on suorituskykyistä, kun käyttäjä lähettää syötteen. Se tarkistetaan vain lähetyksen yhteydessä. Vaikka syötteen tarkistaminen on nopea operaatio, niin saattaa tästä tulla huomattavaa hyötyä, jos lukijoita on huomattavia määriä. Syötteiden tullessa sisään on niiden tarkistaminen kootusti helpompaa kuin ulosmenneessä. Jokainen tuleva syöte tulee vain muistaa käyttää tarkistusalgoritmin lävitse, tämän jälkeen tarkistuksessa esiintyvien virheiden korjaaminen on tehokasta tähän tarkistusalgoritmiin. Tässä näkyy myös Input Encoding -tarkistamisen suurin heikkous. Kun syöte on kerran tallennettu, niin oletetaan, että se on turvallinen. Tietoturvakorjaus ei korjaa aikaisemmin lähetettyjä syötteitä ja on mahdollista, että kantaan unohtuu vaarallistakin sisältöä. Vielä katastrofaalisempaa on, jos hakkeri pääsee jollain tavalla käsiksi kantaan. Hänellä on silloin vapaat kädet ja hänen on erittäin helppo kätkeä jälkensä. [3 s.400-401.].

Output Encodingilla tarkoitetaan kaiken dynaamisen informaation käsittelemistä esityksen yhteydessä. Tärkeätä on, että käsittely on suoritettava aina, kun tietoa esitetään. Unohtunut syöte on aina potentiaalinen tietoturvariski, mutta onneksi korjaukset tietoturvaan poistavat uhkia myös takautuvasti. Tästä kaikesta haittana on suorituskyvyn menetys, mutta samalla se parantaa päivitettävyyttä ja tietoturvaa päivitysten ajalta. [3 s.402]

Oma lukunsa on vielä mahdollisuus lähettää linkkejä. Tämän ominaisuuden liittämistä kannattaa harkita tarkkaan, koska linkkien kautta on mahdollista aikaansaada CSRF-tyyppisiä hyökkäyksiä. CSRF muistuttaa XSS-hyökkäystä moneltakin osaltaan. Puhtaimmillaan se voi ilmetä vaikkapa kuvalinkin kohdalla. Kuvalinkki, joka näyttää kuvan sivuilla, ohjaakin dynaamiseen ohjelmatiedostoon, joka hakee käyttäjän selaimesta osoiterivilta muuttujat ja lähettää sitten kuvan. Pahimmillaan mitään virhettä ei näy, mutta tiedot ovat siirtyneet vääriin käsiin. Toinen esimerkki on ohjata käyttäjän selain ajamaan jonkin kolmannella sivulla jotain mielenkiintoista. Mikäli kuva on sivulla, jolla käyttäjä on todennäköisesti kirjautunut sisälle ja käyttäjällä on sessio kyseisellä sivulla aktiivisena, niin CSRF viritetty linkki voi GET-syötteellä muuttaa session asetuksia. Lähetetty linkki

pitäisi aina tarkistaa mahdollisten GET-syötteiden varalta, jos linkki menee julkisille sivuilla. Hyvin ongelmallista tästä tekee se, että erilaiset kuvapalvelimet saattavat käyttää GET:iä myös normaalisti navigointiin. [3 s.93-97.].

3.2.4 Käyttäjästä riippumattomat syötteet

Interaktiivisilla Websivuilla hyödynnetään yleisesti sessioita. Erilaisilla sessionluontitavoilla on tarkoituksena erotella sivuilla olevat toimijat. Sessiot perustuvat yleensä avaimen välittämiseen. Käyttäjälle annetaan avaimeksi id-luku. Aina, kun uusi sivu ladataan, niin tarkistetaan samalla että avain on yhä voimassa. Jos järjestelmä on toteutettu tietoturvattomasti, niin pelkästään session avaimen kaappaaminen riittää identiteetin varastamiseen. [4 s.113-114.].

Sessioiden ongelmat perustuvat pitkälti siihen, että session avain joutuu vääriin käsiin. Tätä riskiä varten on olemassa lukuisia hyödyllisiä tekniikoita, jotka suojaavat järjestelmää. Ensimmäinen on session vanhenemisaika. Tämä jakautuu kahteen osaan, ensinnäkin evästeelle voi määrittää vanhenemisajan, mutta tämähän ei ratkaise mitään, jos evästeen informaatio on jo väärissä käsissä. Siksi pitääkin avaimen vanhenemisesta huolehtia myös palvelinpuolella. [4 s.119-120.]. Kehä V-palvelun latauksen yhteydessä verrataan aikaa edelliseen latausaikaan, jos aikaväli on liian pitkä, niin tuhotaan sessio eikä latausta jatketa.

Session avaimen vanheneminen kaventaa aikaikkunaa jona aikana väärinkäyttäjän tarvitsee sessio saada haltuunsa. Tämä ei kuitenkaan estä session kaappaamista, käyttämistä tai jopa pitämistä elossa myöhempää käyttöä varten automaattisella järjestelmällä. Työkalut näitä väärinkäytöksiä vastaan ovat session regenerointi eli uudistaminen ja session avaimen jatkuva rotaatio.

Regeneroinnilla tarkoitetaan sitä, että käyttäjän session avain vaihdetaan kun käyttäjä kirjautuu sisään, tai kun käyttäjätilaa vaihdetaan. Näin ollen ei jälkikäteen ole mahdollista koneelta kaapata session avainta evästeistä. Vanhentuneita session avaimia ei hyväksytä sisään. [4 s.118]

Vaikka sessio olisikin hyvin lyhyen aikaa voimassa ja session avain vanhenisi tilan muutoksen yhteydessä, niin jää hakkerille yhä aikaikkuna, jolloin sessio on varastettavissa. Tätä ikkunaa voidaan kaventaa vielä huomattavasti luomalla käyttäjälle uusi avain jokaisella latauskerralla ja hävittämällä vanha. Lopputuloksena on palvelu, jossa jokainen sivunlataus alkaa vanhan evästeen tarkistamisella ja uuden lähettämisellä. Näin aikaansaadaan mahdollisimman lyhyt aikaikkuna väärinkäyttäjälle. On kuitenkin huomioitava että PHP:n sisäänrakennettu sessionluokka ennen PHP 5.1 ei sisällä ominaisuutta tähän, vaan ”session_regenerate_id” funktio luo toki uuden, mutta säästää vanhan. Version 5.1 jälkeen on mahdollista ajaa tieto-turvallinen ”session_regenerate_id(true)”, joka poistaa edellisen session avaimen evästeistä ja toimivien listasta. [4 s.127.].

Erittäin tärkeä osa sessioiden turvallisuutta ei perustu hienoihin teknisiin ratkaisuihin, vaan käyttäjien koulutukseen. Uloskirjautumisen yhteydessä ajetut siivous- ja suojausrutiinit menettävät tehokkaasti merkityksensä, jos sivuilta ei kirjauduta ulos. Salaiset sivut eivät pysy kovinkaan salaisina, jos ne jäävät selaimen välimuistiin seuraavan koneenkäyttäjän luettavaksi. Samalla tavoin kuin tehokkainkaan sessionkaappauksen esto ei suojaa, jos hakkeri arvaa käyttäjän tunnuksen ja salasanan. [4 s.131.].

3.2.4.1 PHP:n sisäänrakennettu session luokka

PHP:n sisäänrakennetut sessiot ovat helppokäyttöisiä ja käteviä. Session tunnistamistapoina toimivat säätöjen mukaan evästeet sekä GET metodilla välitetyt muuttujat. Vaikka GET-välitteisissä metodeissa on ongelmansa, niin tulee tilanteita, jolloin ei voida vaatia, että evästeet ovat päällä. Välitysmetodien rajoittaminen on mahdollista session luokan asettelussa.
[10 s.http://www.php.net/manual/en/session.idpassing.php]

PHP:n session luokka perustuu tunnistetiedot sisältävään avaimeen, joka voidaan välittää evästeenä tai GET-metodin välityksellä. [10 s.http://www.php.net/manual/en/session.security.php] Se ei huomioi käyttäjän ip-osoitetta tai muita tunnistustietoja. Ip-osoite on muutenkin ongelmallinen, useat käyttäjät voivat näkyä samalla osoitteella (proxyt, verkkojen nat:it) ja saman käyttäjän ip saattaa vaihtua liittymästä johtuen.
Jos tunnistetiedot välitetään evästeillä, voidaan hyödyntää normaaleita evästeiden asetuksia, mm. vanhenemisaika, näkyvyys ja muut varastointiasetukset. Pelkän evästeen vanhenemisasetukset eivät kuitenkaan yksin riitä. Johtuen selainpuolen haavoittuvuuksista pitää varautua siihen, että evästeet saattavat joutua vääriin käsiin. [3 s.238.]. Evästeiden kaappaaminen ei ole aivan yhtä triviaalioperaatio kuin GET-metodilla välitettävän näkyvän URL kaappaaminen, mutta ei voida olettaa niiden olevan turvallisia.

Varsinaisen session vanheneminen on aina varmistettava palvelinpuolelta, koska vanhenevakin eväste saatetaan kopioida. Tämä tapahtuu esim. tallentamalla session muistiavaruuteen aikaleima ja vertaamalla sitä kelloon. Mikäli aikojen erotus on vähemmän kuin haluttu vanhenemisaika, päivitetään muistiavaruuteen nykyinen aika, muutoin sessio tuhotaan ja käyttäjä kirjataan ulos. Näin hankaloitetaan session kaappaamista ja suojaudutaan tilanteelta, jolloin käyttäjä on unohtanut vaikkapa julkisen koneen kirjautuneena sivustolle. [4 s.120]

3.2.4.2 Evästeet

On kyse sitten PHP:n sisäänrakennetusta sessionluokasta tai itse rakennetusta session tunnistusjärjestelmästä, on käyttäjä tunnistettava jollain tavoin.

Yleensä sessioiden tunnistustiedot välitetään evästeiden avulla, toinen vaihtoehto on sessioinformaation välittäminen hyödyntäen GET-metodia.

GET:illä tarkoitetaan osoitteen perään kirjoitettua muuttujaan ja sen arvoa, esim. http://www.jokusivu.fi/?muuttuja=arvo. GET-metodin käyttäminen sessioiden välittämiseen mahdollistaa ”sosiaalisia hyökkäyksiä”. Esim. jos varomaton käyttäjä saadaan lähettämään linkki sivuilta, saa hyökkääjä samalla session tunnistetiedot ja käyttäjän kirjautumisen haltuunsa. Selain saadaan myös varsin yksinkertaisella kyselyllä paljastamaan osoiterivin sisällön, jos käyttäjä saadaan ohjattua väärälle sivulle.
[10 s.http://www.php.net/manual/en/session.security.php; 9 s.115–116.].

Evästeiden käyttäminen ei suojaa kaikilta hyökkäyksiltä, mutta ne eivät sentään liiku osoitteen mukana käyttäjältä toiselle. Evästeiden kaappaaminen vaatii joko pääsyä käyttäjän koneelle tai tietoturva-aukon hyödyntämistä järjestelmässä, jonka sessio halutaan kaapata. Selain kun luovuttaa evästeet vain niille sivuille, joille ne on ilmoitettu kuuluvaksi. Hakkerin oma sivu ei voi evästeitä kaapata, jos ei hakkeri voi hyödyntää tavalla tai toisella kohdesivua. [4 s.69.].

Tämä ei kuitenkaan ole aivan koko totuus. Evästeet välitetään usein verkon ylitse suojaamattomina, eikä jokaiselle sivulle ole mahdollisuutta hankkia HTTPS-suojausta, varsinkin kun tehokas suojaus on maksullinen. Suojaamatonta liikennettä voidaan kuunnella koneilla, joiden kautta se kulkee, ja tämän mukana kulkevat myös session avaimet. Tietoturvalliseltakin järjestelmältä voivat laillisen käyttäjätilin omaavat käyttäjät kalastella sessioiden avaimia, koska ne tallennetaan tiedostojärjestelmään eikä niitä ole automaattisesti suojattu mitenkään. Lukemista ei voida oletusasetuksilla olevassa järjestelmässä estää PHP:lta muutenkaan, koska suorittavaa käyttäjää ei erotella. [4 s.114-115.].

3.2.5 JavaScriptin hyödyntäminen

Vuodesta 1995 käytössä ollut JavaScript on kuten PHP:kin tulkattava kieli, jonka komennot upotetaan nettisivun HTML-koodin sekaan. Merkittävimmät erot näiden kahden kielen hyödyntämisessä tulevat siinä että PHP suoritetaan palvelimella sivun lähettämisen aikana, kun JavaScript taas lähetetään ja ajetaan selaimessa. Tästä ominaisuudesta seuraavat JavaScriptin suurimmat edut ja haitat. [6 s.254.]. Modernissa Web-ohjelmoinnissa selainpuolen suoritusta hyödynnetään AJAX:iksi kutsutussa tekniikassa. AJAX perustuu siihen, että JavaScriptillä voidaan tutkia ja käsitellä HTML-dokumentin rakennetta. Yhdistettynä siihen, että selainpuolella ajettu koodi voi reagoida käyttäjän toimiin, saadaan aikaiseksi reaaliaikaisia ominaisuuksia ja käyttäjän syötteiden ja tietokannan välisiä yhteyksiä. [6 s.260, 381.].

3.2.5.1 Näkyvyysongelma

Javascript ajetaan selaimessa ja se on nähtävissä sivun lähdekoodissa. Koska JavaScript on kaikille nähtävissä, on väärinkäyttäjien helppo etsiä tietoturva-aukko suoraan lähdekoodista. JavaScript on rosoisen taipaleensa aikana saanut muutenkin maineen ”tietoturvariskinä” ja tästä johtuen se-

laimissa on mahdollisuus kytkeä JavaScript pois käytöstä. Mitään ominaisuutta ei siis ole suositeltavaa toteuttaa pelkästään JavaScriptin varassa. JavaScriptin puuttuminen ei saa tehdä sivusta käyttökelvotonta tai osalla käyttäjistä saattaa asia muodostua kynnyskysymykseksi.

3.2.5.2 Moninkertainen tarkistelu

Vaikka JavaScript ei ole kelvollinen suojametodi erilaisille syötteille, jotka tulevat sivuilta, niin se voi joskus olla erittäin suureksi avuksi myös tämänkaltaisessa toiminnassa. Esimerkiksi tiedoston lähettäminen palvelimelle Web-käyttöliittymän kautta, jossa tiedoston lisäksi on pakotettu jotain meta-tietoja, vaikkapa tiedoston esitysnimi. Jos tämä toteutetaan ainoastaan PHP:n varassa, niin syntyy seuraavanlainen tilanne. Selain lähettää tiedoston ja siihen liittyvät tiedot POST-metodilla, tähän saattaa kulua huomattava aika riippuen tiedoston koosta. Kun kaikki on siirretty, niin käsitellään tiedostoon liittyvät tiedot. Jos vaadituissa tiedoissa on puutteita, niin tiedostoa ei käsitellä, vaan käyttäjää pyydetään täydentämään tietoja ja tiedosto on lähetettävä uudelleen. Ratkaisu selainpuolen PHP-ohjelmassa on, joko toteuttaa erillinen tietojen täydennysjärjestelmä, jossa tiedosto säilytetään väliaikaissäilössä, kunnes tiedot on täydennetty tai täyttää puuttuvat tiedot automaattisesti parhailla mahdollisilla arvoilla. Näistä tavoista saattaa seurata ongelmia, joista pitää huolehtia erikseen, esimerkiksi jos käyttäjä päättääkin jättää tietojen täydentämisen kesken ja tiedosto jää väliaikaissäilytykseen. Yksinkertainen varmistus tähän on toteuttaa kenttien pakottaminen JavaScriptin avulla ennen tiedoston lähettämistä. Tämä ei estä väärinkäytöksiä ja syötteet on tarkistettava vielä PHP:lla mutta se tekee järjestelmästä käyttäjäystävällisemmän.

3.3 MySQL tietokannat

MySQL on yleisesti verkko-ohjelmoinnissa käytetty tietokantajärjestelmä. Toisin kuin lukuisat kilpailijansa on MySQL ilmaiseksi saatavilla. Lisäksi MySQL omaa erinomaisen suorituskyvyn ja saumattoman yhteensopivuuden PHP:n kanssa. [6 s.37-39] MySQL tietokanta sisältää mahdollisuuden rajata hyvin tarkasti eri käyttäjätunnusten oikeuksia. Tarkasti määritellyillä oikeuksilla olisi mahdollista rajata esimerkiksi Websivun eri käyttäjätasojen oikeudet tietokannankin puolelta ja saada näin vielä yksi suojaus mahdollisia injektio-hyökkäyksiä ja muita tietokantaan kohdistuvia väärinkäytöksiä kohtaan. Pahimmassa tapauksessa, jos väärinkäyttäjä saa käsiinsä tietokannan salasanan ja onnistuu luomaan yhteyden tietokantaan, olisivat rajoitukset vielä suojaamassa osaa tiedoista ja rajoittamassa tuhoja. [6 s.120–122]

Käyttäjätilit ovat yleensä suojattu paitsi salasanalla, niin myös määritellyllä sijainnilla. Vaikka käyttäjätunnus ja salasanapari joutuisivat vääriin käsiin, ei tietokantaan voi muodostaa yhteyttä muualta. [6 s.123]

On oman kokemuspohjani perusteella hyvin tavanomaista, että palveluntarjoaja eli järjestelmän ylläpitäjä luo jokaiselle asiakkaalleen tietokantaan yksittäisen tunnuksen, jolla on kaikki hallintaoikeudet sivustoa varten luotuun tietokantaan /-kantoihin.

Tämä asettaa PHP sovellutukselle omat vaatimuksensa. Mikäli PHP sovellutuksen tarkistukset eivät ole kunnossa ja väärinkäyttäjä pääsee kirjoittamaan omaa koodiaan tietokantaan *(injektio-hyökkäys)*, niin suoritetaan koodi hallintatunnuksilla usein tuhoisin seurauksin. Pahin mahdollinen vaihtoehto on, että tavalla tai toisella tämmöinen salasana joutuu vääriin käsiin, mikäli palvelinpuoli mahdollistaa ulkopuoliset yhteydet, niin voi sillä aikaansaada huomattavaa tuhoa ja päästä käsiksi kaikkeen tietokannan informaatioon. Onneksi yleensä näin ei ole, tavanomaisempaa on, että tietylle sivustolle tarkoitettu tietokanta on rajoitettu toimimaan vain kyseisen sivuston palvelimelta.

On hyvin yleistä, että palvelimilla on valmiina PhpMyAdmin-niminen erinomainen työkalu. Mikäli sitä ei ole asennettu valmiiksi, onnistuu sen asentaminen ja käyttöönottaminen samoilla oikeuksilla, joita tarvitaan sivujen lisäämiseen palvelimella, koska PhpMyAdmin on toteutettu kokonaan PHP:lla. [6 s.101.]. PhpMyAdmin on helppokäyttöinen ja tehokas MySQL-tietokantojen hallintatyökalu. PhpMyAdmin on PHP-ohjelma joka pyörii palvelinkoneella ja mahdollistaa käyttäjälle pääsyn tietokantojensa tietoihin, sekä hallintaympäristön kautta, että SQL kyselyitä kirjoittamalla. [6 s.98.]. On erittäin tärkeää huomata, että PhpMyAdminin tehokkuus ei ole pelkästään hyvä asia, mikäli palvelimella on PhpMyAdmin, niin tietokannan käyttäjätunnus-salasanapari on ainut suojaus väärinkäytöksiä vastaan.

3.3.1 PHP:n Tietokantakäsittelijät

Tietokannat ovat olennainen osa nykyisiä verkkosovelluksia. Dynaamisuus tarvitsee tapoja hallita suuria määriä tietoa ajonaikaisesti. On kyse sitten wikistä, kirjautumista vaativasta palvelusta tai verkkokaupasta, on tietokanta luonnollinen valinta. PHP:n vahvuuksia on ehdottomasti sen loistava yhteensopivuus erilaisten tietokantojen kanssa. [6 s.37-38.].
Tietokantojen käyttöönotto ja hyödyntäminen on PHP:ssa vaivatonta ja helppoa. MySQL-tietokanta, jota on käytetty molemmissa käsitellyissä projekteissa, hyödyntää asiakas-palvelin-arkkitehtuuria. Tästä johtuen se voidaan halutaessa asettaa pyörimään, vaikka eri koneelle kuin sivusto, tai ajaa sitä samalla koneella, mutta erillisenä prosessina. PHP:n puolelta tietokantaan pitää ensin ottaa yhteys ja sitten lähettää SQL-kielisiä kyselyitä. [6 s.39-40.].
Ennen PHP:n versiota 5.1 luotiin yhteys MySQL tietokantaan ”mysql_” alkuisista komennoista rakentuvalla kokonaisuudella, joka on yhä aivan toimiva varsin tehokas tapa käsitellä tietokantoja. Näytönpaikan Portfoliossa on hyödynnetty näitä ominaisuuksia, ohjelmakoodin selkeyden ja turvallisuuden vuoksi nämä komennot suoritetaan kuitenkin erillisessä oliossa. Tämä tietokantakäsittelijäolio sisältää kaikki tietokantaa koskevat muuttujat, kaikki tietokantaan vaikuttavat funktiot ja se hallinnoi yhteyttä

tietokantaan sivun lataamisen ajan ja katkaisee yhteyden, kun sivu on ladattu.
Ekesin Kehä 5 palvelussa on hyödynnetty uudempaa PDO-luokkaa.

3.3.1.1 PDO luokka

PDO tulee sanoista ”PHP Data Objects” ja se on ollut PHP:ssä versiossa 5.1 lähtien. Käytännössä PDO on sisään rakennettu tietokantojen hallintaluokka. Toisin kuin edeltäjänsä ei PDO ole sidottu mihinkään yksittäiseen tietokantaan, vaan se tulee kaikkia yleisimpiä tietokantoja. Hyödynsin sitä itse MySQL tietokannan kanssa, mutta yhden rivin muutoksella toimisi kaikki ohjelmakoodini yhtäläisesti Oracle DB:n, Microsoft SQL Serverin, PostgrepSQL, ODBC:n tai vaikka SQLiten kanssa. [5 s.155-156.].

3.3.1.2 MySQL-komentokanta

PHP:ssa on versiosta 4 asti ollut sisäänrakennettuna tuki MySQL tietokannalle joka toimii ”mysql_” alkuisten funktioiden avulla. MySQL-komentokanta on ollut PHP:ssa vakiona mukana versiosta 4 lähtien. Kyseessä on vanha komentokanta, eikä sitä mielestäni ole suunniteltu tietoturvallisuuden näkökulmasta kovinkaan hyvin. Koska tässä järjestelmässä on omat rajoituksensa ja heikkoutensa, on tarkistuksiin kiinnitettävä erityistä huomiota. MySQL-komennot käyttävät oletuksena aina viimeistä kantayhteyttä ja useamman kannan käyttöön valmistautuessa on syytä erottaa kantaan liittyvät kahvamuuttujat, kyseiset muuttujat on myös syytä suojata. Modernimpi vaihtoehto on hyödyntää tämän komentokannan kanssa olio-ohjelmoinnin työkaluja. Rakentamalla toimivan olion komentokannan ympärille saadaan huomattava osa komentojen rajoituksista korjattua. Merkittävin riski MySQL-komentokannan kanssa on kuitenkin sen haavoittuvuus tietokantaa vastaan suuntautuville injektiohyökkäyksille. MySQL komentokannan tietokantakomennot eivät automaattisesti suojaa injektiolta, vaan siitä tulee huolehtia manuaalisesti komentokannassa olevien suojauskomentojen tai muuttujien tyyppikonversioiden avulla. [10 s.http://www.php.net/manual/en/book.mysql.php]

3.3.2 Vaarallisten syötteiden torjuminen

Tietokantoja vastaan suunnattujen hyökkäysten merkitys kasvaa koko ajan, kun tietokannassa olevan tiedon arvo kasvaa. Toisinkuin vaikkapa XSS eivät tietokantakeskeiset hyökkäykset suuntaa suoraan käyttäjiä tai heidän selaimiaan vastaan, vaan suoraan palvelinta. Yleisin tapa toimia tietokantoja vastaan on yrittää ajaa kyselyn alkuperäisen kyselyn yhteydessä omia kyselyitä tai muokata normaalin kyselyn osia. Mikäli syötteitä ei ole tarkistettu kunnolla, voi väärinkäyttäjä tehdä jotakuinkin mitä haluaa. Vaikkapa tilanteessa, jossa tarkistetaan kirjautumista, voi väärinkäyttäjä manipuloida tietokannasta tulevaa vastausta. Pahantahtoinen hyökkääjä voi pahimmillaan tuhota kokonaisia tauluja tai jopa koko tietokannan. [4 s.73- 74, 77-78.].

3.3.2.1 PDO:n vahvuudet

PDO:lla on muitakin etuja kuin edellä mainittu yhteistoimintatuki kaikkiin yleisiin tietokantoihin. PDO hyödyntää tekniikkaan, jota kutsutaan yleensä Prepared Statements:iksi. Idea tässä tekniikassa on, että varsinainen kysely ja siihen liittyvät muuttujien arvot erotellaan ja valmistellaan erikseen, ennen kyselyn suorittamista. Tästä saadaan useita huomattavia etuja. Tehtävien suorituskyky tehostuu, mikäli on tarkoitus ajaa sama kysely useaita kertoja käyttäen eri arvoja. PDO:lla riittää, että varsinainen kysely valmistella ja ohjata tietokantaan vain kerran, tämän jälkeen voidaan lähettää useita erilaisia dataosoita jotka sitten sijoitetaan samaan kyselyyn.
Tietoturvan kannalta tämä erikseen valmistelu on erinomainen vaihtoehto, koska kaikki komennot käsitellään erikseen datan kanssa. Seurauksena on, ettei datan seassa olevaa ohjelmakoodia suoriteta. Tämä on erittäin tehokas tapa suojautua injektio-hyökkäyksiltä. [5 s.162-163, 9 s.75-77]

Hyvin toimivassa järjestelmässä unohtuvat helposti erilaiset virheilmoitukset ja vastaavat sovelluskehittäjille suunnatut työkalut toimintavalmiuteen. Etuna tässä kaikessa on, että mahdollisen virhetilanteen ilmenemisen yhteydessä kehittäjä näkee suoraan, missä vika on. Ongelmalliseksi tämän kaiken tekee se, että virheilmoitukset paljastavat usein hyvinkin paljon tietoja ja huolimattomasti sijoitettuna ne eivät valikoi, kenelle näitä tietoja paljastavat. Ääriesimerkkinä PDO:n virheilmoitukset, jotka tietokantalinkin puuttumisen yhteydessä kirjoittavat virheilmoitukseen mm. tietokannan salasanan. Erilaisten virhetilanteiden aikaansaaminen onkin yksi hakkereiden käyttämistä tavoista erilaisten haavoittuvuuksien löytämiseksi. [4 s.79-80.].

3.3.2.2 MySQL komentokannan suojaukset

MySQL komentokanta sisältää kaksi suojausfunktiota. Nämä funktiot ovat “mysql_escape_string” ja “mysql_real_escape_string”. Käytännössä kyse on lähes identtisestä funktiosta, ”mysql_real_escape_string”-functio tuli PHP 4.3.0:ssa ja samalla se korvasi edeltäneen ”mysql_escape_string” funktion, mutta vanhemmissa versioissa ja yhteensopivuuskysymysten vuoksi molemmat funktiot ovat yhä toimivia. Erona näissä funktioissa on, että ”mysql_real_escape_string” huomioi merkkisetin eikä sitä voi ohittaa syöttämällä tietokantakomentoja eri merkkisetillä.
[10 s.http://www.php.net/manual/en/function.mysql-escape-string.php, s.http://www.php.net/manual/en/function.mysql-real-escape-string.php]

3.3.2.3 Vaihtoehtoja

Tietokantahaavoittuvuuksien eliminoiminen on mahdollista myös manuaalisesti, hyödyntäen PHP:n tarjoamia merkkien korvausmetodeita. Manuaaliselle tarkistelulle ei PDO:n käytön yhteydessä ole syytä, koska PDO:ssa käytetään prepared statement logiikkaa, mutta mysql_ komentokannan yhteydessä se voi poikkeustapauksessa olla hyödyllinen.

Esimerkiksi tilanne, jossa käyttäjän tulee voida ajaa omia komentoja suoraan tietokantaan, eli esimerkiksi oma versio phpMyAdminista vaatii että sivuilta tulee voida syöttää SQL komentoja suoraan kantaan. Manuaalisella tarkistelulla olisi mahdollista estää oikeasti vaarallisia syötteitä ja mahdollistaa vain osa komennoista. Yleisesti ottaen on kuitenkin parempi, jos ei yritä olla liian ovela, vaan hyödyntää valmiiksi testattuja suojametodeita. [10 s.http://www.php.net/manual/en/function.mysql-real-escape-string.php; s.215.].

Mikään ei estä käyttämästä myöskään magic quotes gpc:tä estämään tietokantaan kohdistuvia hyökkäyksiä, mutta silloin on ensinnäkin huomioitava se, että palvelinpuolen päivitys poistaa tämän ominaisuuden käytöstä tai muuttaa sitä, ja että kaikki syötteet joutuvat samaan käsittelyyn riippumatta niiden käyttötarkoituksesta. [4 s.74.].

Muuttujatyyppiin pakottaminen (type cast) ja erilaiset salasanatiivisteet ovat hyvä vaihtoehto erilaisille tarkistusfunktioille. Ne ovat tehokkaita, koska eivät sisällä turhia tarkisteluita eikä niiden läpi ole mahdollista suorittaa hyökkäyksiä tietokantaa vastaan. [4 s.78; 7 s.264-266.].

3.3.3 Ei kenenkään informaatio

Tavalliselle verkonkäyttäjälle ei välttämättä ole aina aivan selvää, miten paljon informaatiota hänestä itse asiassa jää verkkoon. Sivusto voi tallentaa kellonaikojen lisäksi, käyttäjän selaimen antamat tekniset tiedot ja yksityiskohtaista tietoa käyttäjän toimista.
Käyttäjää koskevan informaation keräämistä on rajoitettu monien maiden lainsäädännössä.

Vuonna 1999 säädetyssä henkilötietolaissa säädetään reunaehtoja erilaisten tietojen keräämisestä rekistereihin. Laissa on määritelty miten henkilötietoja on käsiteltävä, miten niitä saa kerätä ja mitä niillä saa tehdä. Erityisesti nettisivuilla tapahtuvaan henkilötietojen keräämiseen on liityttävä aina tieto siitä, että tietoja kerätään ja julkistettava ihmisille kerätyt tiedot. Laki koskee kuitenkin ennen kaikkea henkilötietoja, anonyymeihin selaustietoihin sitä ei yleensä sovelleta. [2 s.411-414.].

Amerikassakin kriittisimpien informaatioiden keräämistä on rajoitettu, ihmisten luottotietojen rekisteröintiä harjoittaa vain kolme yritystä. Ongelmalliseksi muodostuu informaation laittaminen verkkoon nähtäväksi. Kuinka estää, etteivät asiattomat tahot pääse näkemään kriittistä informaatiota? Yleinen tapa tunnistaa yksittäisiä ihmisiä on sosiaaliturvatunnus, mutta se on useiden yritysten tiedossa juurikin tästä syystä. Tunnistamiseen tarvitaan siis lisäinformaatiota, henkilötietoja, käyttäjätunnuksia ja salasanoja. Valitettavasti suuryritykset, jotka tietävät hallinnoivansa hyvin kriittistä informaatiota, saattavat olla haavoittuvaisia erilaisille hyökkäyksille. Miten on sitten kaikkien niiden pienten yritysten, yhdistysten ja yksityisten sivustojen laita, jonne ihmiset antavat samoja palasia? Kaikki henkilötiedot ovat arvokkaita väärinkäyttäjille, joiden tarkoituksena on kaapata verkossa olevia identiteettejä. [3 s.228-231.].

Varomattomat käyttäjät saattavat käyttää samaa käyttäjätunnus-salasanaparia useilla eri sivuilla. Pahimmassa tapauksessa yksi palvelu, jolla tietoturva ei ole kunnossa, vaarantaa useita muita. Tästä syystä netissä vastuulliset nettipalvelut eivät tallenna salasanoja selväkielisinä mihinkään. Sen sijaan salasanoista lasketaan salasanatiivisteitä, erilaisia algoritmeja tähän tarkoitukseen on useita, yleisimpiä lienevät md5 ja sha1. [6 s.264-266.].
Suomessa esiintynyt laajin tapaus ilmeni 07/2007, kun Viestintäviraston Tietoturvayksikkö CERT-FI ilmoitti, että nettiin oli julkaistu 80 000 käyttäjätunnuksen tiedot. Suureksi onneksi suurin osa näistä tunnuksista oli tallennettu md5 / sha1 – salasanatiivisteinä.
[14 s. http://www.cert.fi/varoitukset/2007/varoitus-2007-7.html.].
Selväkielisistä käyttäjätunnus-salasanapareista voi väärinkäyttäjä luoda automaatiotyökalun, joka kokeilee näitä tunnuksia kymmenille sivustoille ja merkitsee ylös toimineet tunnukset.

3.3.3.1 Salasanojen suojaamisen tapoja

Yleisimpiä tapoja suojata tietokannassa olevia salasanoja ovat erilaiset tiivistealgoritmit. Käytetyimpiä algoritmeja ovat md5 ja sha1. Salasanatiivisteiden perusidea on, että salasanasta lasketaan tarkistussumma, jota ei saa palautettua takaisin alkuperäiseksi salasanaksi. Tiivisteet ovat aina samanmittaisia, riippumatta lähteen pituudesta. Esimerkiksi md5 summa on aina 32 merkin mittainen merkkijono ja se voidaan yhtä hyvin laskea lyhyestä salasanasta salasanatiivisteeksi tai valtavasta tiedostosta tarkistussummaksi. Lopputuloksesta ei ole mahdollista sanoa, mistä summa on laskettu. Tämä tekniikka ei luonnollisesti suojaa tarpeeksi heikkoja salasanoja paljastumasta, mutta vahvojen salasanojen kohdalla se on vähintäänkin tehokas hidaste hakkerille, joka on päässyt näin pitkälle. [6 s.264-266.].
Tallennetun salasanan suojausta voidaan tehostaa tallentamalla tietokantaan tiiviste, joka ei muodostu pelkästä salasanasta, vaan salasanasta ja koodiosasta, joka lisätään käyttäjän salasanan yhteyteen. Tällä tavalla voidaan hankaloittaa huomattavasti heikkojen salasanojen purkamista. [6 s.279-280.].

3.4 Apachen ja palvelimen ominaisuudet

Apache on yksi merkittävimpiä palvelinympäristöjä netissä, perinteisesti Apachea ajetaan Linux-palvelinympäristössä. Apache ja sen kakkosversio (Apache 2) ovat sinällään vain verkkopalvelinohjelmistoja. Dynaamiseen Web-ohjelmointiympäristöön tarvitaan muitakin työkaluja. Hyvin yleinen palvelinalusta, jota itsekin tässä käsittelen, kulkee lempinimellä LAMP. Nimen kirjaimet tulevat sanojen Linux, Apache, MySQL ja PHP alkukirjaimista. [7 s.294-300.].

3.4.1 Apachen laajennuksia

Apachelle on olemassa lukuisia laajennuksia, joita kutsutaan moduuleiksi. Ideana näissä moduuleissa on, ettei kaikkea toiminnallisuutta tarvitse rakentaa palvelimen ytimeen, vaan ominaisuuksia saadaan lisättyä helposti. Näin palvelimen ylläpitäjä kykenee hienosäätämään järjestelmänsä itselleen sopivaksi. Nämä moduulit lisäävät palvelimeen todella tehokkaita ja hyödyllisiä lisäominaisuuksia. [7 s.297.].

Tietoturvan kannalta ehdottomasti mainitsemisen arvoinen on mod_ssl. Tämä on vaihtoehtoinen tapa muokata Apache palvelin suojattua SSL yhteyttä varten. Toinen tapa asentaa SSL-suojaus palvelimelle on asentaa Apache alusta asti SSL tilassa, moduuli käytännössä vain helpottaa palvelimen ylläpitoa. [7 s.299.].

Toinen erittäin hyödyllinen moduuli on nimeltään mod_rewrite. Apachen sivuilla sitä kuvataan sanoilla ”the Swiss Army Knife of URL manipulation!” (Sveitsin armeijan taittoveitsi URL manipulaatioon). Hyvin kuvaavaa tästä tekee se perusajatus, että kyseessä on hyvin yksinkertainen kapistus, mutta osaavissa käsissä sillä on lukematon määrä hyödyllisiä käyttötapoja. Perusideana mod_rewritessä on URL osoitteiden muokkaaminen latauksen yhteydessä. Vaikkapa osoitteen loppuosan muuttaminen toiseen muotoon, esimerkiksi käsiteltäväksi GET metodilla ja siten hyödynnettäväksi dynaamisen koodin sisäisenä informaationa.
[11 s. http://httpd.apache.org/docs/2.2/rewrite/]
Käytännössä olen kaikilla käsitellyillä sivuilla hyödyntänyt mod_rewriteä siten, että GET-muuttuja request_uri tallennetaan kaikki URL:n alainen informaatio. Esimerkiksi kehä 5 sivusto, joka on täysin dynaaminen. Apache ohjastaa osoitteeseen http://www.keha5.fi asti ja kaikki sen jälkeinen käsitellään mod_rewritellä. Käytännössä tämä tarkoittaa, että käyttäjän palkissa näkyvästä osoitteesta

http://www.keha5.fi/Logistiikka_ja_liikenne/Satamat

tulee mod_rewriten käsittelyn jäljiltä palvelimelle osoite

http://www.keha5.fi/?request_uri=Logistiikka_ja_liikenne/Satamat

Tämän jälkeen pilkotaan request_uri irtonaisiin osiin käyttäen erottimena kenoviivaa. Näiden palasten avulla sitten haetaan tietokannasta halutun alasivun sisältö.

Tietoturvan kannalta tästä käsittelystä saadaan muutamia erittäin huomattavia hyötyjä. Ensinnäkin ohjelmatiedostot ovat kaikki kansiossa, joka ei ole näkyvissä Webin puolelta. Tästä kansioon viitataan index.php tiedostossa ja siellä tarkistellaan käsitelty URL. Vahvuuden tässä järjestelmässä verrattuna perinteiseen kansiorakenteeseen ovat huomattavia. Ensinnäkään hyökkääjä ei voi vain alkaa arvailla ohjelmatiedostojen nimiä kansiossa, koska osoitteella ei ole mitään tekemistä todellisten kansioiden kanssa. Pahimmillaan hyökkääjä voi arvaamalla päästä käsiksi lähdekoodeihin ja

mikäli hyökkääjä saa käsiinsä lähdekoodin, helpottuu tietoturva-aukkojen etsintä huomattavasti.
Salasanojen takana olevat sivut ovat kaikki suojatussa kansiossa, joten päästäkseen käsiksi mihinkään suojattuun informaatioon on käyttäjän selaimen ensin osoitettava, että sillä on avoin sessio sopivilla oikeuksilla.
Kolmas merkittävä etu on dynaamisuus, yksittäinen suojattu kansio sisältää perusohjelmiston, johon kaikki syötteet ohjataan. Virtuaaliset kansiot, eli vaikkapa edellisen esimerkin Logistiikka_ja_liikenne ovat vain dynaamisesti lisättyä sisältöä tietokannassa. Näin ei hakemistorakenne muutu ollenkaan, vaikka alasivuja lisättäisiin.

Periaatteessa samankaltaiseen syöttöön pääsisi ilman mod_rewriteä luomalla sivun ohjaukset suoraan GET-metodilla, eli viittaamalla suoraan osoitteeseen joka sisältää GET-muuttujia. Tämä tekisi kuitenkin sivun osoitteesta huonommin luettavan käyttäjälle ja paljastaisi enemmän tietoja sivustosta mahdollisille väärinkäyttäjille. Samalla tietojen suojaaminen palvelimella vaikeutuisi huomattavasti.

3.4.2 Kansioiden näkyvyysasetukset

Kuten jo aikaisemmin mainitsin, niin on aina hyvin riskialtista, jos sivuston lähdekoodi joutuu vääriin käsiin. Tietoturva-aukkojen etsintä on huomattavasti helpompaa, jos voi tutustua suojauksiin. Apachessa on kokonaisvaltaisten asetusten lisäksi mahdollista säätää kansiokohtaisia asetuksia. Tämä tapahtuu lisäämällä Web-sivun kansioon ”.htaccess” nimisen tiedoston. Tämä tiedosto on yleensä käytössä kaikissa Web-palvelimissa. Sen oikeudet ovat säädettävissä Apaches konfigurointi tiedostoista. Paitsi Apachen sivukohtaisia asetuksia on tässä tiedostossa mahdollista määritellä joitain PHP:n asetuksia ja säätää mod_rewriten asetuksia.
[11 s. http://httpd.apache.org/docs/2.2/rewrite/rewrite_flags.html]
Itse olen hyödyntänyt PHP säätömahdollisuuksia lähinnä kasvattamalla suoritukselle varattua timeout aikaa ja tiedonsiirron rajoja. Oletuksena kun php.ini:ssä nämä ovat hyvin rajoitettuja. Tiedonsiirtoraja katkaisee tiedoston siirron tietyn koon jälkeen, ja estää näin valtavien tiedostojen lähettämistä ja vähentää liikennettä. Varsinkin videotiedostojen kanssa ovat nämä rajat kuitenkin auttamattomasti liian alhaisia. Timeout-aika katkaisee koodin suorittamisen, kun sivun lataaminen viivästyy tietyn ajan ylitse. Ilmiö tulee käytännössä vastaan samaisten isojen tiedostojen kanssa tai jos ylläpidollisilla sivuilla ajetaan scriptejä, jotka käsittelevät suuria tietomääriä. (vaikkapa tietokannan päivityksiä tai kuvien muokkauksia.)
Kansion selausasetukset verkosta ovat säädettävissä ”.htaccessissa" ja yhdistettynä mod_rewriteen, voidaan kaikki ko. kansioon tulevat pyynnöt ohjata haluttuun ohjelmatiedostoon. Näin voidaan rajoittaa kaikki ulkoapäin tuleva liikenne oikeille urille. En ole nähnyt tarpeelliseksi suojata grafiikkaa ja CSS-tiedostoja sisältäviä kansioita, koska on tehokkaampaa hakea ne suoraan Apachen läpi kuin viitata niihin dynaamisella koodilla.
Suojatun alueen grafiikka ja tiedostot (vaikkapa valokuvat ja PDF-tiedostot) olen säilönyt verkosta näkymättömiin kansioihin. Niiden lukeminen tapahtuu siten, että PHP ohjelmakoodi tarkistettaa ensin käyttäjän

luvan kyseiseen tietoon, ja lähettää sen tarkistuksen jälkeen käyttäjän selaimelle.

3.4.3 Tiedostojen oikeudet palvelimella

PHP:n perinteisiä ongelmia on se, että PHP ajetaan itse asiassa ”kolmantena käyttäjänä”. Tämä käyttäjätili on Ubuntun vakioasetuksilla nimetty www-dataksi. Nämä vakioasetukset ovat hyvin ongelmalliset tietoturvan kannalta. Jotta PHP kääntäjä voi lukea tiedostoja, on tiedostoon oltava lukuoikeudet kolmansilla käyttäjillä. Tämä koskee sekä ohjelmakoodia että suojatussa kansiossa olevia sisältötiedostoja. Pahimmillaan, jos PHP:n on tarkoitus kirjoittaa tiedostoon, on sen kirjoitusoikeudetkin jaettava kolmannelle käyttäjälle. Kolmannella käyttäjällä tarkoitan siis palvelimen käyttötiliä, joka ei ole tiedoston omistaja tai tiedoston omistajan ryhmässä.[4 s.136.].

Tästä seuraa useita mahdollisia väärinkäytöksi, joille perusasetuksilla oleva Web-hotelli on haavoittuvainen. Jotta PHP-tulkki voi lukea ohjelmakoodin, on sillä oltava oikeudet. Vielä kun käyttäjällä ei yleensä ole admin tunnuksia, ei käyttäjälle jää muuta vaihtoehtoa kuin jättää ohjelmakoodiin kaikki oikeudet. Tästä seuraa, että ohjelmakoodi on jokaisen palvelimen käyttäjän nähtävissä. Toinen vaihtoehto olisi rajoittaa lukeminen PHP-tulkille, mutta tästä ei ole paljoa enempää hyötyä, koska jokainen käyttäjä hyödyntää samaa PHP-tulkkia, niin on triviaali urakka rakentaa PHP-ohjelma, joka käy lukemassa haluamasi tiedostot ja esittää ne. [4 s.136.].

Palveluntarjoajat ovat onneksi heränneet tähän ongelmaan ja PHP:hen on olemassa useita erilaisia tietoturvapäivityksiä. Esimerkiksi SJR:ssä (palvelin, jolla Näytönpaikan Portfolio pyörii) on käytössä suPHP-niminen tietoturvallisuuspaketti. SuPHP:n idea on, että tulkki toimiikin käyttäjän nimellä ja ryhmällä. Lopputuloksena on, ettei ohjelmakoodia tai suojattuja tiedosta sisältäviin kansioihin tarvitse päästää muita käyttäjiä. Tämä on tehokas tapa suojata kaikki ohjelmakoodit väärinkäyttäjiltä. Erityisen huomattava etu tulee erilaisten asetustiedostojen kohdalla. Koska esimerkiksi tietokannan käyttäjätunnuksen ja salasanan on oltava PHP:n luettavissa jossain tiedostossa, yleensä ohjelmakoodin seassa. [12 s. http://www.suphp.org]

SuPHP on rakennettu niin, että se antaa virheilmoituksen, jos tiedostojen oikeuksia ei ole rajoitettu riittävästi. Vähimmillään on tiedostojen oikeuksien oltava 755, eli omistajalla kaikki oikeudet ja muilla oikeudet lukea ja ajaa. Sekä Näytönpaikan Portfoliossa, että Ekesin Kehä 5 promootiopalvelussa olen rajoittanut oikeudet kaikkiin ohjelmatiedostoihin ja kansioihin muotoon 700, eli omistajalla on kaikki oikeudet ja muilla ei mitään. Kukaan ei siis pääse edes näkemään näitä tiedostoja. [7 s.104–108.].

Koska varsinainen index.php tiedosto ajetaan julkisesta kansiosta, niin mitään tärkeitä määrittelyitä ei tule laittaa sinne. Apachen ”.htaccess” tiedostossa määritellään, että mod_rewrite välittää kaikki kyseistä kansiota seuraavat syötteet GET:inä tähän tiedostoon. Sitten index.php asetetaan polun

suojattuun hakemistoon, joka sisältää ohjelmakoodit ja kutsuu sitten varsinaisen sivun ohjelmakoodia.

Ohjelman suunnittelussa tulee aina asennoitua siihen, että käyttäjältä tuleva GET-syöte voi sisältää mitä tahansa ja päätyä kenen tahansa nähtäväksi. GET:iä pidetään niin suojattomana, että W3C:n suosituksen RFC 2616 mukaan ei GET:iä tulisi ollenkaan käyttää. [4 s.160.].
On siis selvää, ettei mitään kriittistä informaatiota tule välittää GET:inä, eikä esimerkiksi tiedoston nimiä. Molemmissa toteutetuissa projekteissa on käytetty seuraavanlaista rakennetta. Index.php lataa suojatussa kansiossa olevan main.php tiedoston, joka sisältää switch-case rakenteen GET-syötteen käsittelemiseksi. Mikäli GET ei ole annetulla valkealla listalla, eli esiinny yhtenä ennakkoon valmistelluista sivuista, niin ohjelma lataa virhe-ilmoituksen eikä jatka eteenpäin. [4 s.96.].

3.4.4 SSL suojaus

Kantava idea SSL yhteydessä on, että käyttäjän ja palvelimen välinen liikenne suojataan SSL Cryptaus-algoritmilla. Lisäksi tähän suojaukseen liittyy se, että SSL- sivulla on luotettavan tahon myöntämä sertifikaatti. Näin asiakas voi olla suhteellisen varma, että asioi oikean sivuston kanssa. SSL suojatun sivuston tunnusmerkit ovat https ja selaimessa näkyvä lukon kuva. [7 s.299] .

4 OHJELMAVERSIOIDEN MERKITYS TIETOTURVAAN

4.1 Monijakoinen päivitysongelma

Nykyään päivittäminen on yksi tietoturvan kulmakivistä. Päivittämisestä on tullut arkista rutiinia ja jokaiselta käyttäjältä odotetaan automaattisesti päivittämistä. Helpottaakseen käyttäjän urakkaa tarjoavat ohjelmat päivityksiä automaattisesti tai jopa päivittävät itseään kysymättä. Yleensä maksulliset ohjelmat sisältävät ilmaisia päivityksiä, joko eliniäksi tai määräajaksi. Varsinkin virustorjuntaohjelmissa on tapana myydä määräaikaisia päivityspaketteja. On enemmän sääntö kuin poikkeus, että ohjelma on päivitettävissä. Tämä ei siis ole riippuvainen ohjelman tyypistä. Päivityksen korjaavat ohjelmien virheitä ja tilkitsevät ohjelmien aukkoja. [1. s.15-23]

4.1.1 Päivitysten merkitys

Päivitysten merkitys on huomattava, on kyse sitten käyttöjärjestelmästä, selaimesta tai palvelinohjelmasta. Windows-koneen ollessa kyseessä on käyttöjärjestelmän päivittäminen ehdottoman elintärkeää. [1 s.38-47] Ohjelmien päivityksen korjaava yleensä ohjelmassa olevia virheitä ja korjaavan haavoittuvuuksia erilaisille väärinkäytöksille. Web-hotellin asiakkaina olemme kokonaan palvelimen ylläpidon päivitysten varassa. Tästä johtuen on asiakas yleensä vanhempien versioiden varassa.

4.1.2 Tiedon julkaisusta seuraavat ongelmat

On hyvin yleistä että ohjelmien päivitysten julkaisemisen yhteydessä kerrotaan, mitä on korjattu ja mitä tietoturva-aukkoja on tukittu. Esimerkiksi php.net sisältää jokaisen version kohdalla tiedot korjatuista XSS haavoittuvuuksista. Etuna julkistuksesta on, että tietää kuinka kriittinen päivitys on ja taitava ohjelmoija voi suojata oman koodinsa kyseisiltä haavoittuvuuksilta. Haittana on, että kun tietoturva-aukko tulee julkiseen tietoon, niin mahdollinen väärinkäyttäjäkin saa tietoonsa kyseisen haavoittuvuuden. Tämä johtaa siihen, että päivittämättömien palvelimien asema huononee päivityksen julkistuksen yhteydessä.

Erityisen ongelmallisen tästä tekee se, että merkittävimmät palveluntarjoajat tarjoavat hyvinkin vanhoja versioita ohjelmista. Nopea katsastus Elisan, Soneran ja Saunalahden sivuille paljasti käytössä olevan yhä PHP 4-versioita. [1 s.26, 10 s. http://www.php.net/releases/]

Nollan päivän hyökkäyksiksi (zero day attacks) kutsutaan hyökkäyksiä, jotka suoritetaan samana päivänä kuin tietoturvapäivitys julkaistaan. Ajatuksena on hyödyntää aukkoa ennen kuin se ehditään tukkia. [7 s.230-231]

4.1.3 Vanhojen versioiden käytön edut

Vanhojen versioiden käytössä on myös omat etunsa. Koska tietoturva-aukot ovat tiedossa, voidaan niihin varautua ohjelmallisesti. Erilaisia tietoturvapäivityksiä on saataville esim.PHP eri versioille (esimerkiksi HardenedPHP http://www.hardened-php.net/hardening_patch.14.html), jotka tilkitsevät vanhojen ohjelmien aukkoja. Mikään ei kuitenkaan estä sitä, etteikö vanhastakin järjestelmästä voisi löytyä uusia aukkoja. Aina ei ole mahdollista suorittaa radikaaleja päivityksiä palvelimelle.
Esim. PHP:n kanssa ongelmaksi muodostuu alaspäin yhtyeensopimaton koodi. Ohjelma, joka toimii täydellisesti PHP 4-versioissa, ei välttämättä toimikaan PHP:n 5-version kanssa.
[11 s. http://www.php.net/manual/en/migration5.incompatible.php]
Päivityksissä ongelmaksi saattaa joskus muodostua uusien versioiden epävakaus ja mahdolliset asentelusta johtuvat käyttökatkokset. [1 s.21]

4.2 PHP:n versiohistoria tietoturvan näkökulmasta

Päädyin käsittelemään PHP:n versiohistoriaa tämän työn toteutuksen yhteydessä tarkemmin, koska se on toistaiseksi ollut se yleisin kysymys, joka pitää selvittää palvelinvalinnan yhteydessä. Apachen versio ei yleensä rajoita toimintaa. Samoja SQL-kyselyitä hyödyntää eri tietokantojen kanssa ja PDO mahdollistaa tämän jopa ilman raskaita muutostöitä. PHP:n versio on kuitenkin aina rajoittava tekijä. Jos palvelimella on käytössä PHP:n 4:n versio, niin muuttaa tämä koko olio-ohjelmoinnin luonteen, eikä PDO:ta voi yleensä hyödyntää jos PHP:n versio ei ole vähintään 5.1.

Jokaisessa PHP:n versiossa on tullut huomattavia päivityksiä, ja useilla korjatuilla virheillä saattaa olla huomattaviakin vaikutuksia tietoturvaan. Jokainen kriittinen virhe on potentiaalinen tietoturvariski, koska kaatuva ohjelma paljastaa usein tietoja hakkereille. Pahimmillaan osittain kaatunut järjestelmä saattaa jäädä hyvin haavoittuvaiseksi. Keskityn kuitenkin käsittelemään vain mielestäni merkittäviä tietoturvaan liittyviä korjauksia ja parannuksia versiohistoriasta.

Versio 4
Tarkempaa tietoa PHP versio 4-päivityksistä ja kehityksestä on saatavissa sivuilta. http://www.php.net/ChangeLog-4.php

4.03
mysql_escape_string tuli PHP:hen, tämä oli yleisesti käytetty suojaustapa mysql_ tietokantakäsittelijän kanssa. Vanheni versiossa 4.3. Tämä funktio korvaa SQL kysellyssä käytetyt merkit vaarattomilla ja estää näin tietokannan injektiohyökkäykset. [10 s. http://www.php.net/ChangeLog-4.php]

4.1
Uudet $_GET, $_POST, $_COOKIE, $_SERVER ja $_ENV lisättiin. Nämä korvasivat vanhan $HTTP_*_VARS arrayn. Uusien muuttujien etuina on lyhyempi ja selvempi käsittely, sekä näkyvyys koko sovellutuksessa. Muutenkin päivitys antoi hyviä työkaluja tietoturvallisen PHP:n kir-

joittamiseen, mutta osaamattomissa käsissä tehokkaat työkalut aikaansaavat riskejä.

Lisättiin $_REQUEST array, joka sisältää GET, POST ja COOKIE syötteet. Tämä ominaisuus on hyvin kyseenalainen tietoturvan kannalta. [10 s. http://www.php.net/ChangeLog-4.php]

4.2
Register Globals oletuksena pois käytöstä. Register Globals poistunee lopullisesti versiossa 6.0. Tämä on kaikin puolin hieno juttu PHP:n tietoturvan kannalta. [10 s. http://www.php.net/ChangeLog-4.php]

4.2.2
Päivitys, jolla korjattiin yksittäinen erittäin vaarallinen tietoturva-aukko. Aikaisemmissa versioissa oli POST-metodilla tulevien lähetteiden tarkistus puutteellinen. Tämä haavoittuvuus mahdollistaa paikallisen ohjelmakoodin lähettämisen ja ajamisen palvelimelle, joka vastaanottaa POST-metodin syötteitä. Mikäli tätä päivitystä ei voida asentaa, niin on suositeltavaa rajoittaa koko POST-metodi pois käytöstä Apachen asetusten kautta. [10 s. http://fi.php.net/releases/4_2_2.php]

4.3
PHP:ssä otettiin käyttöön uusi mysql_real_escape_string ja samalla vanheni aikaisempi mysql_escape_string. Erona on, että real_escape tarkistaa ensin tietokannan merkkityypin ja huomioi sen korvauksessa, muutoin merkkien korvaus on identtinen edeltäjään verrattuna.
Tämän lisäksi vanhenivat mysql_drop_db ja mysql_create_db ja otettiin käyttöön huomattava määrä uusia mysql_ komentoja. Käytännössä mysql_ metodi sai nykyisen muotonsa ja sen tietoturvaa koskeva tutkielmani koskee vain versio 4.3 jälkeisiä ominaisuuksia.
Magic_quotes tuli php.ini:iin säädettäväksi. Tämän tietoturvallisuusominaisuuden hyödyistä ja haitoista on yllä.
SHA1 eli US Secure Hash Algoritm 1 (40 merkkinen hash summa, kuten MD5) tuli lisättiin PHP:hen. [10 s. http://www.php.net/ChangeLog-4.php]

4.3.1
Korjattiin vakava CGI haavoittuvuus, joka mahdollisti käyttäjien lukea tiedostoja palvelimelta, tämä koski kaikkia tiedostoja jotka ovat näkyvissä PHP-koodia ajavalle käyttäjälle. Tämän lisäksi käyttäjän oli mahdollista ajaa omaa PHP-koodiaan tämän haavoittuvuuden kautta. [10 s. http://fi.php.net/releases/4_3_1.php]

4.3.5
Tässä versiossa korjattiin vakava virhe PHP:ssa, mikä mahdollisti, että käyttäjät pääsivät käsiksi toisen käyttäjien virtuaalihostien konfiguraatiotietoihin. Tämä mahdollisti vielä enemmän tiedonkalastelua palvelimen muiden käyttäjien asetuksista ja esti salasanojen suojaamiset php.ini tiedostoon. [4 s.83]

4.4.2
Korjattiin jotain tietoturvapuutteita, merkittävimpinä headereiden rajoittaminen, jolla ehkäistään header injektiota sekä XSS-haavoittuvuus virheilmoituksista. [10 s. http://fi.php.net/releases/4_4_3.php]

4.4.3
Korjattiin XSS-haavoittuvuutta phpinfo() työkalusta.
[10 s. http://fi2.php.net/releases/4_4_2.php]

4.4.7
Paikattiin lisää XSS-haavoittuvuuksia phpinfo() työkalusta.
[10 s.http://www.php.net/releases/4_4_7.php]

Versio 5
PHP:n 5-versiossa tuli hyvin merkittäviä päivityksiä, varsinkin tietoturvan kannalta.

Olio-ohjelmointi muuttui täysin PHP:n 5-versiossa, siihen asti oli se ollut lähinnä tapa järjestellä ohjelmakoodia oliomaisesti ilman olioiden ominaisuuksia. Versiossa 5 olioiden tietojen suojaaminen mahdollistui, kun vanhemmissa versiossa kaikkien olioiden tiedot olivat näkyvissä ohjelmakoodiin ja toisiin olioihin. PHP:n 5 versiossa tuli myös huomattava määrä valmiita luokkia ja tätä laajennettiin versiossa 5.1. Tämän päivityksen jäljiltä PHP:ssa on yli sata valmista luokkaa hyödynnettäväksi. [5 s.12.]. Kuten edellä olen käsitellytkin, perustuvat useat hyödyntämäni tietoturvaratkaisut nimenomaan olio-ohjelmoinnin työkaluihin.

Komentotulkin ydin, Zeng Engine päivitettiin seuraavaan version. Tästä seurasi huomattavia parannuksia tehokkuuteen ja toimivuuteen.
[10 s. http://www.php.net/manual/en/migration5.php]

Hyvin mielenkiintoinen lisäys koskien tietokantoja, oli SQLiten lisääminen vakiona osaksi PHP:n moottoria[10 s.http://www.php.net/ChangeLog-5.php; 8 s.141]. SQLite on tietokanta, joka perustuu yksittäiseen tiedostoon, johon koko kanta tallennetaan, täten kyseinen kanta on helposti siirrettävissä. Esimerkiksi hyökkääjien toimintaa seuraamaan rakennettu honeypot järjestelmä kannattaa rakentaa SQLiten varaan, koska siirrettävyys luo lisää mahdollisuuksia analysointia varten. [4 s.168.]. SQLite tietokantaa ei muutenkaan tulisi väheksyä, koska se on aivan täysipainoinen tietokanta ja se on saatavilla myös silloin, kun raskaammat tietokannat kuten MySQL puuttuvat.[5 s.139]

5.0.5
Lisättiin ”mysql_set_charset” komento, jolla voidaan säätää tietokantaan menevän syötteen merkkityypille vakioasetuksia.
[10 s. http://www.php.net/manual/en/function.mysql-set-charset.php]

5.1.0
PDO, PHP Data Object tuli pysyväksi osaksi PHP:ta. Tämä mahdollisti hyvin erilaisen lähestymistavan tietokantoihin. Lisäksi huomattavia haa-

voittuvuuksia korjattiin kielestä, mm. XSS-haavoittuvuus phpinfo() komennosta ja yli 400 muuta virhettä.
[10 s. http://www.php.net/releases/5_1_0.php]

Session luokkaan lisättiin hyvin hyödyllisiä ominaisuuksia. Session_regenerate_id sai lisämäärityksen, jolla PHP poistaa automaattisesti vanhan session ja mahdollistaa session id:n kierrättämisen. Ilman tätä kun ei id:n kierrättämisestä paljoa etua ollut, kun vanha sessio id jäi odottamaan väärinkäytöksiä. [4 s.128]

5.1.2
Tämä päivitys sisälsi huomattavan tärkeitä päivityksiä tietoturvaan. Estettiin ettei header() funktio välitä päällekkäisiä headereistä ja poistettiin XSS-haavoittuvuus virheilmoituksista. Lisäksi korjattiin mm. PDO:ssa olleita ikäviä virheitä. [10 s. http://www.php.net/releases/5_1_2.php.].

5.1.3
Tämä päivitys lisäsi huomattavia korjauksia koskien XSS-haavoittuvuuksia sekä buffer overload -hyökkäyksiä. Lisäksi pakotettiin copy() funktio safe_mode:en.
[10 s. http://www.php.net/releases/5_1_3.php.].

5.1.5/5.1.6
Tässä päivityksessä korjattiin safe_mode, jota toimi väärin useista eri komennoista. Lisäksi paikattiin lisää overload virheitä, korjattiin GD laajennuksessa ollut overload ongelma virheellisten GIF-kuvien kanssa. [10 s.http://www.php.net/releases/5_1_5.php.].

5.1.6 on käytännössä korjattu 5.1.5 jonka julkaisussa oli virhe.
[10 s. http://www.php.net/releases/5_1_6.php.].

5.2.0
Tämä päivitys, vaikka korjasi pääosin varsinaisia virheitä, ratkaisi myös joitain yksityiskohtia tietoturvasta. Lähinnä koskien safe_modea ja joidenkin funktioiden raja-arvojen ylityksistä seuraavia tietoturvaongelmia.
[10 s.http://www.php.net/releases/5_2_0.php.].

5.2.1
Tässä päivityksessä oli keskitytty pääosin erilaisiin tietoturva-ongelmiin. Tässä päivityksessä PHP:n muistinkäyttö rajattiin oletuksena ja suojattiin useita funktioita eri-laisilta ylikuormittamiseen liittyviltä tietoturvaongelmilta (overload, un-derload).
[10 s.http://www.php.net/releases/5_2_1.php]

5.2.2
Tässä päivityksessä huomionarvoinen oli superglobals arvojen ylitsekirjoittamisen estäminen, sekä CSRF:n estäminen ftp_ komentojen yhteydessä. Graafisia funktioita sisältävään GD-luokkaan tehtiin tietoturvapäivitys, joka esti väärän kuvakoon hyödyntämistä. Edellisissä versioissa oli mah-

dollista aktivoida register_globals hyödyntämällä mb_parse_str():ssä olevaa bugia. [10 s.http://www.php.net/releases/5_2_2.php]

5.2.3
Graafisen GD luokan virheitä korjattiin lisää, PNG-muotoisen kuvan kohdalla oli mahdollista aikaansaada loputon silmukka. Tällä oli mahdollista kuormittaa palvelinta ja mahdollisesti kaataa se. Samassa päivityksessä ratkaistiin haavoittuvuuksia sähköpostin lähetysominaisuuksien kanssa.
[10 s.http://www.php.net/releases/5_2_3.php]

5.2.4
Useiden GD:n alaisten funktioiden kohdalta ehkäistiin numeraalisten arvojen raja-arvojen ylityksistä johtuvia virhetiloja.
[10 s.http://www.php.net/releases/5_2_4.php]

5.2.5
PHP:n suojafuntiot htmlentities ja htmlspecialchars korjattiin toimimaan oikein eri merkkiseteillä. Aikaisemmissa versioissa nämä hyvinkin käytetyt suojafunktiot oli mahdollista kiertää esimerkiksi UTF-7 merkkityyppiä hyödyntämällä.
[10 s.http://www.php.net/releases/5_2_5.php]

5.2.6
Työn toteuttamishetkellä viimeisin PHP:N päivitys. Merkittävimpiä muutoksia olivat mielestäni PCRE:n versio päivitys (eli PERL-tyyppisten regular expresioineden).
[10 s.http://www.php.net/releases/5_2_6.php]

Versio 6.0.0
PHP-kielestä on tulossa seuraavaa versio. Varmaa tietoa seuraavan version sisällöstä ei tämän työn tekohetkellä ole julkaistu, mutta spekulaatioita tulevan version sisällöstä on olemassa. Merkittävin julkaistu uudistus on siirtyminen UTF-8 merkistöön oletuksena. Tässä seuraavassa versiossa ollaan keskittymässä enemmän tietoturvallisuuteen ja useat vanhentuneet, mutta silti yleisessä käytössä olevat ominaisuudet tullaan poistamaan. Lähdössä on ainakin register_globals asetukset, magic_quotes ja huonosti toimiva safe_mode ominaisuuspaketti. [6 s.396–399]

5 PROJEKTIKOHTAINEN LÄHESTYMINEN

Tämä osio käsittelee varsinaista työosuuttani, eli kahta hyvin erilaista sivustoa. Käsittelemäni projektit on toteuttanut Teknologiateollisuuden KT-keskus, jonka palkkaamana toimin Riihimäen HAMK:in toimipisteessä kesällä 2007 ja 2008. Kesällä 2007 olin toteuttamassa Näytönpaikan Portfolion luomista ja vuonna 2008 sille toteutettiin huomattava päivitys. Pääasiallinen työni oli Webohjelmoinnin parissa, eli toteutin HTML, CSS, PHP ja MySQL-osuudet näistä projekteista. Opinnäytettäni koskien olen kiinnittänyt erityistä huomiota näiden sivustojen tietoturvaan. Aikomukseni ei ole läpikäydä projekteja kokonaisuudessaan tässä osiossa. Yleisen kuvauksen ja rakenteen jälkeen keskityn tietoturvaan vaikuttaviin ominaisuuksiin, havaintoihin ja ratkaisuihin, joita olen hyödyntänyt.

5.1 EKES - Kehä V Promootiohanke

Ekesille toteutettu Kehä V -projekti toteutettiin kesällä 2008 ja se sisälsi kaksi osaa, dynaamisesti päivitetyn verkkosivuston sekä USB-tikuilla jaettavan markkinointimateriaalin.

5.1.1 Toiminnallisuuden määrittely

Sivuston www.keha5.fi on tarkoitus olla virtuaalinen Kehä V -promootiohanke, ja sen on tarkoitus tukea varsinaista markkinointia. Tästä johtuen sivuston ulkoasu toteutettiin yhteistyössä markkinoinnista vastaavien tahojen kanssa. Varsinainen toiminnallisuus oli rajattu jo projektin suunnitteluvaiheessa kolmelle eri käyttäjäryhmälle, vierailijalle, kirjautuneelle käyttäjälle sekä ylläpitäjälle. Sivuston kaikki dynaaminen toiminta rajattiin pelkästään ylläpitäjän käyttöön. Kirjautuneen käyttäjän näkymä on kuten vierailijankin näkymä, mutta siellä on nähtävissä vain kirjautuneille suunnattuja osioita. Kirjautuneella käyttäjällä tai vierailijalla ei ole minkäänlaista interaktiivista sisältöä.

Ylläpitäjän näkymässä on kaikki sivuston sisältö muokattavissa lomakepohjaisella käyttöliittymällä. Tämä mahdollistaa sivuston ylläpitämisen ilman teknistä osaamista; sivuston ylläpitäjä voi keskittyä vain sisällön tuottoon. Sivusto jakautuu kolmeen palstaan ja yläpalkkiin, jossa olevat graafiset elementit valitaan alasivun mukaan.

5.1.2 Tekniset ominaisuudet

Ekes tilasi palvelintilan Kehä V projektin sivulle www.keha5.fi kotimaiselta webhotellilta www.avaruus.net. Versiot olivat toteuttamisvaiheessa Apache 2.0.59, MySQL 5.0.20a-log ja PHP 5.2.3. MySQL ja PHP:n versiot olivat nähtävissä php:n kautta, Apachen-version pyysin sähköpostitse palvelimen ylläpidolta. MySQL-tiedonsyöttöä varten oli palvelimella käy-

tössä phpMyAdmin 2.6.1-rc2. Sivuston rakenteen toteutin yhden sivun periaatteella [6 s..]. Tällä tarkoitetaan sitä, että sivu rakennetaan vain kertaalleen ja varsinainen koodi hakee kaiken sisällön tietovarastoista ennen sivun varsinaista esittämistä. Lopputuloksena aikaansaadaan hyvin mukautuva sivustorakenne. Kehä V:ssä päädyin vieläpä poikkeuksellisemmin hyödyntämään vain yhtä templaattia. Tähän templaattiin rakensin sijoittamispaikat kaikelle mahdolliselle informaatiolle, jota sivulla saattaa esiintyä. Näinpä varsinaisen ohjelmakoodin kannalta jokainen sivu on täysin samanlainen.

5.1.3 Tietoturvaan vaikuttavat ratkaisut

Kehä V -tietokannan hallinta toteutettiin PDO:n avulla. PDO suojaa automaattisesti tietokantaa vastaan suunnatuilta injektio-hyökkäyksiä, koska PDO:ssa kaikki tietokantakyselyt on suoritettu Prepared Statement -logiikalla.

Kaikki dynaaminen sisältö tarkistetaan vaarallisen sisällön varalta ruudulle tulostuksen yhteydessä (Output Encoding). Toinen vaihtoehtoinen lähestymistapa olisi ollut tarkistusten sijoittaminen syötteiden lähetyksen yhteyteen. Etuna tässä lähestymistavassa olisi ollut parempi suorituskyky. Vain yksi käyttäjä voi syöttää sivuille dataa, mutta useat käyttäjät voivat sitä lukea. Nyt jokaisen käyttäjän kohdalla suoritetaan tarkistukset erikseen. Tähän ratkaisuun päädyin kahdesta syystä. Ensinnäkin, sivujen toiminnallisuudesta huolehtiva julkaisujärjestelmän tietoturvapäivitykset vaikuttavat takautuvasti, ja toiseksi, sallittujen syötteiden lista vaikuttaa jo syötettyihin sivuihin. Lisäksi etuna jälkikäteisessä tarkistelussa on se, että erilaisten syötteiden käsittely voidaan toteuttaa toisistaan poikkeavien ehtojen mukaan. [3 s.400-402]

Syötteiden käsittely on toteutettu hyödyntäen PHP:n vakiomuotoista htmlentities-komentoa. Erityismuotoiluita varten on mahdollistettu hyvin rajattu HTML käyttö. Tämä toimii siten, että ennen htmlentitiesin ajamista käydään syöte lävitse ja korvataan sallittujen HTML-tagien hakasulut aaltosuluilla. Esimerkiksi <h1> korvataan {h1}, htmlentitiesin jälkeen käydään sama lista uudestaan ja palautetaan sallitut tagit, eli {h1} korvataan <h1>:llä. Jokainen korjaus on eritelty, molempiin suuntiin ja vain listan mukaiset korvaukset suoritetaan. Tätä tekniikkaa kutsutaan White List Validatinion:iksi [4 s.36.].

Yleinen rakenne on toteutettu siten, että julkisessa kansiossa on gfx hakemisto, joka sisältää JavaScript ja css-tiedostot sekä kuvitusgrafiikat. Varsinaista kuvasisältöä varten on kansio kuvat ja pdf-sisältöä on kansio pdf. Kuvat kansioon on tallennettu kaikki kuvituskuvat, kuvien nimeäminen tapahtuu automaattisesti ja eri kuvatyypit on eritelty nimeämiskäytännön avulla. Kuvan nimi alkaa tunnisteosalla, esim. ”gal” on gallerian tunnistusosa. Kuvat eritellään tunnisteosan jälkeen seuraavalla 5 numeron sarjalla. Tämä sarja on suoraan kuvaa vastaavan tietokantamerkinnän id-kenttä, ja siten yksilöllinen jokaisella kuvalla.

Yleisessä hakemistossa sijaitsee index.php, johon ”.htaccess” tiedostossa ohjataan mod_rewriten avulla urlin loppu GET-metodilla. Index.php sisältää ainoastaan ohjauksen oikeaan kansioon ja lataa sieltä main.php tiedoston. Main.php:ssä tarkistetaan, onko GET:inä tuleva syöte jokin ennakkoon sallituista tai löytyykö se tietokannassa määriteltyjen sivujen tunnistetiedoista. Mikäli kumpikaan ehto ei täyty tulostetaan virhesivu. Suojatut tiedostot on tallennettu suojattuun kansioon, ohjelmakansion sisällä ja ne on tallennettu ilman mitään kytköksiä sivuilla oleviin tiedostonimiin, linkitys on ainoastaan tietokannassa. Suojatuilla sivuilla olevien tiedostojen lukeminen tapahtuu siten, että ensin haetaan kannasta tiedoston tiedot ja käyttöoikeudet, tarkistetaan käyttäjän kirjautuminen ja käyttöoikeus, lopuksi luetaan tiedoston sisältö suojatusta kansiosta ja lähetetään se käyttäjän selaimelle.

Palvelinpuolella on huomioitava että PHP-tulkki pyörii tilin käyttäjän oikeuksilla, eikä kolmantena henkilönä (www-data tms.). Tämä mahdollistaa sen, että kaikkien palveluun liittyvien tiedostojen käyttöoikeuden on rajattu käyttöjärjestelmän puolelta. Tämä koskee niin lähetettyjä tiedostoja kuin ohjelmakoodiakin.

5.2 Näytönpaikka Ry:n elektronisen portfolion päivitys

Näytönpaikka Ry:n portfolio on tarkoitettu yhteydenpitovälineeksi sosiaalialan ammattilaisten ja elämästä syrjäytyneiden asiakkaiden välille. Palvelu saattaa sisältää hyvinkin luottamuksellista tietoa käyttäjistään. Lähtien siitä, ketkä ovat palvelun käyttäjiä. Koska palvelun sisältämä informaatio saattaa olla arkaluontoista, on suojautuminen väärinkäytöksiltä erityisen tärkeää. Tämän lisäksi oli huomioitava, että osa asiakkaista saattaa omata rajoituksia, jotka ovat alun perin vaikuttaneet heidän tilaansa, ja tämä asetti käyttöliittymälle ja käyttäjän vastuulle omat rajoituksensa.

5.2.1 Toiminnallisuuden määrittely

Lopullisessa palvelussa käyttäjätilejä on kolmea eri luokkaa. Ensimmäisenä on Admin, tällä tilillä ei ole näkyvyyttä kenenkään tietoihin, eikä viestiyhteyksiä. Adminilla on oikeudet luoda ja poistaa käyttäjätilejä sekä vaihtaa käyttäjätilien tyyppiä. Henkilökuntatilit eivät sisällä omaa profiilia, mutta niillä on näkyvyys niiden käyttäjien jakamiin tietoihin jotka ovat valinneet heidät omiksi yhdyshenkilöikseen. Henkilökuntatililtä voi lähettää viestejä niille käyttäjille, jotka ovat valinneet kyseisen henkilökunnan jäsenen. Varsinainen käyttäjätili sisältää profiilin, johon on mahdollista luoda verkostokarttoja, ansioluettelo, tallettaa tukiverkon yhteystietoja, pitää verkkopäiväkirjaa ja oppimispäiväkirjaa, vastata elämäntilannetta kartoittaviin kyselyihin ja tallentaa tiedostojen turvakopioita. Käyttäjätilin omistaja voi valita yhteyshenkilönsä, joille tiedot jaetaan henkilökuntatilien listasta ja mitä tietoja hän haluaa jakaa, hän voi myös lähettää viestejä omille yhteyshenkilöilleen. Palvelun runko rakennettiin kesällä 2007 ja päivitys toteutettiin kesällä 2008. Päivityksessä toteutetuista muutoksista on lista liitteestä 1.

5.2.2 Tekniset ominaisuudet

Näytönpaikan portfolio on rakennettu Matti Niemelän kanssa yhteistyössä ja luokkarakenne tässä projektissa on hänen käsialaansa. Template moottorin perusideana on, että muistiavaruus suojataan käyttäjältä. Samalla PHP ja HTML-koodit saadaan eriytettyä mahdollisimman hyvin. Tietokantakäsittelijänä on PHP:n MySQL-luokkaa hyödyntävä käsittelijäluokka. Palvelimellä pyörii suPHP, jonka ansiosta kaikki lähdekoodi on säädetty vain sivun ylläpitäjän ja luonnollisesti pääkäyttäjän nähtäväksi. [12] Näytönpaikan sivut ja portfolio (http://www.naytonpaikka.fi ja http://portfolio.naytonpaikka.fi) sijaitsevat SJR:n palvelimella (http://homepage-sjr.fi/). Ohjelmaversiot olivat toteuttamishetkellä Apache 2.2.3 (Debian), PHP 5.2.6 ja MySQL 5.0.32. Tietojen syöttö tapahtui phpMyAdmin 2.8.1 kautta.

5.2.3 Tietoturvaan vaikuttavat ratkaisut

Näytönpaikan Portfolion tietokantayhteys on toteutettu PHP:n MySQL_ komentojen avulla. Tietokantaan menevät tekstimuotoiset syötteet on käsitelty mysql_real_escape_string komennolla. Numeraaliset syötteet on type castattu numeraalisiksi muuttujiksi, eli esim. kokonaislukusyöte tallennetaan (int) alkuliitteellä, jolloin ainoastaan numerot tallentuvat kantaan, mahdolliset muut tiedot ohitetaan.

Kaikki dynaaminen sisältö sivulla käsitellään htmlentities-komennolla syötteen esittämisen yhteydessä. Laajensin käsittelyn koskemaan myös lomakkeita. Koska testauksessa havaitsimme, että viestijärjestelmässä vastaanotettu viesti on loogista avata suoraan painamalla reply painiketta. Tämä mahdollistaisi suoraan toista käyttäjää vastaan suunnatun xss-hyökkäyksen aloittamalla textarean sulkemisella.

Syötteet käsitellään Näytönpaikan Portfoliossa aina syötteen esittämisen yhteydessä, mikä oli luonnollinen lähestymistapa, koska näytönpaikassa keskimääräinen käyttäjä kirjautuu sisälle ja tuottaa itse sisältöä, joka esitetään vain hänelle itselleen ja hyvin rajatusti toiselle käyttäjälle. Varsinaista julkista dynaamista sisältöä on, mutta sitä on hyvin rajoitetusti. Ainut julkinen dynaaminen sivu on järjestelmän etusivu eikä kyseisellä sivulla ole rajoitettu HTML-syötteiden käyttöä. Etusivua voi muokata vain Admin-oikeuksilla, samoilla tunnuksilla on myös rajoittamattomat oikeudet muokata kaikkien käyttäjätilien tietoja.

Käyttäjä voi määritellä mitä informaatiota hän haluaa jakaa ja valita työntekijätilit, joille kyseiset informaatiot jaetaan. Missään tilanteessa eivät käyttäjät saa tietoa toisistaan, henkilökunnan edustajille näkyvät tiedot on käsitelty joko täysin erilaiseen muotoon tai vähintään estetty muotoilut ja ohjelmakoodin suorittaminen.

Kevät-kesällä 2008, SJR asensi palvelimilleen suPHP- laajennuksen. Tämä mahdollistaa käyttäjätilin suojaamisen palvelimella siten, etteivät toiset käyttäjät pääse tietoja lukemaan. Kaikki tiedostot ja ohjelmakansiot on nyt

suojattu muilta käyttäjiltä. Index.php:hen ohjataan mod_rewritella kansiorakenne GET metodilla. Index.php lataa suojatussa kansiossa olevan main.php:n, jossa switch-case rakenteella käydään läpi sallittujen syötteiden lista.

6 TESTAUKSEN TOTEUTTAMINEN

6.1 Lähtökohtien tarkentaminen

Oikean työkalun käyttäminen oikeassa paikassa on hyvä lähtökohta kaikessa. Tämä pätee myös verkkosivujen tietoturvallisuuden tarkistamisessa. Alkutilanteen kartoittaminen on ensimmäinen vaihe turvallisuuden kartoittamisessa. Erilaiset erityispiirteet erilaisissa palveluissa kannattaa huomioida testauksessakin. Erilaiset sivustojen ja palveluiden toteutuksessa hyödynnetyt tekniikat ja kokonaisuudessaan erilaiset ominaisuudet asettavat kehyksen mahdollisille haavoittuvuuksille.

Valitsin ensimmäiseksi lähtökohdaksi käyttäjän, keitä he ovat? Miten käyttäjät on rajattu? Erilaisten käyttäjäryhmien kohdalla tämä pohdinta on tehtävä erikseen. Jos tekstinsyöttökenttä on rajoitettu ainoastaan ylläpitäjän näkymään niin joskus voi olla edullistakin mahdollistaa html muotoinen syöte ja enemmän. Tämänkaltaisessa tilanteessa harkittu tietoturva-aukko mahdollistaa sivulle toiminnallisuutta joka muuten olisi hyvin hankalaa toteuttaa. Tämä tilanne esiintyy näytönpaikan ylläpitonäkymässä.

Hallittu tietoturva-aukko on rajattava huolellisesti. Jokaisessa portaassa on tarkistettava että käyttäjällä on oikeus tähän syötteeseen. Ensimmäinen tarkastus on sivun lataamisen yhteydessä ja seuraava ennen lomakkeen syötteen käsittelemistä. Näin varmistutaan siitä, ettei väärinkäyttäjä voi lähettää jollain toisella tavalla (esim omalla lomakkeellaan) dataa, jonka palvelumme käsittelisi.

Tähän liittyy myös erilaisten käyttäjäryhmien tietotason arviointi. Esim. html syötteen mahdollistaminen ei ole kovinkaan perusteltua jos siitä ei synny käyttäjille mitään lisäarvoa. Html:ää muistuttavia merkintäkieliä on käytetty aikaisemmin foorumeilla, mutta niistä saatava hyöty on mielestäni välillä hieman kyseenalainen.

Tekniset ominaisuudet ovat toinen huomioitava lähtökohta, lähinnä dynaamisen sisällön laatu. Jos palvelu ei toimi ollenkaan tiedostojen kanssa, on turvallisinta estää koko tiedostojen lähetys, näin suojaudutaan joistain palvelinpuolen haavoittuvuuksilta. Kuvien kohdalla on turvallisempaa aina käsitellä kuvat jollain muokkausmoottorilla, koska jossain poikkeustilanteissa on mahdollista upottaa ohjelmakomentoja kuvatiedoston sijaan sivuille. Tästä on lisätietoja tämän työn kohdassa 3.2.2.

Kahdessa erityistapauksessa joita tässä työssä on käsitelty, on hyvin erilaiset tekniset ja toteutukselliset ominaisuudet.

6.1.1 Näytönpaikka Ry

Näytönpaikka on käyttäjävetoinen palvelu. Sillä on suuri määrä peruskäyttäjiä joilla kaikilla on mahdollisuus lisätä sivuille sisältöä. Tästä johtuen tarkistettavien kohtien määrä on suuri, koska kaikki käyttäjien lisäämä sisältö on potentiaalinen riski. Eräissä tapauksissa sisältö on ainoastaan käyttäjän itsensä nähtävissä ja riski on suhteellisen pieni. Mahdollinen skenaario on, että potentiaalinen väärinkäyttäjä voi ensin turvallisesti testata hyökkäyksiään sivujen jakamattomalla osalla, ja kun ne toimivat sitten jakaa ne työntekijöille. Jakamaton osio on myös erinomainen kokeilukenttä tietokantaa vastaan suuntautuville hyökkäyksille, koska tämä sisältö ei tule koskaan kenenkään tietoon.

Toinen käyttäjäryhmä Näytönpaikan sivuilla ovat työntekijät, minulle annettujen tietojen mukaan työntekijöiden teknisen osaamisen taso saattaa olla hyvin alhainen. Tämä ei kuitenkaan estä mahdollista yksittäistä väärinkäyttötapausta, eikä se että työntekijät on rajattu ihmisryhmä. Heidän oikeuksiaan koskevassa määrittelyssä olikin vielä korostettu, etteivät he saa missään tilanteessa päästä näkemään käyttäjien tai toisten työntekijöiden tietoja ilman että ne on erikseen jaettu heille.

Ylläpitäjälle on annettu huomattava määrä valtaa, mutta ylläpitäjältä on estetty pääsy käyttäjien tietoihin. Tietosuojan lisäksi tällä on tarkoitus minimoida vahinkoja, jos ylläpitäjän tunnukset joutuvat väärinkäyttäjän käsiin, niin väärinkäyttäjä voi tuhota käyttäjiä ja lisätä uusia muttei pääse huomaamatta luottamuksellisiin tietoihin käsiksi. Tilanne on kuitenkin kriittinen, koska ylläpitäjä voi vaihtaa käyttäjän salasanan ja sen jälkeen kirjautua ko käyttäjänä sisään, tämän jälkeen huomaa käyttäjä ettei hänen salasanansa toimi. Toinen mahdollinen väärinkäytönpaikka on etusivulla oleva html muotoista syötettä hyväksyvä kenttä, tähän on mahdollista asettaa erilaisia xss hyökkäyksiä jotka jäädessään huomaamatta keräilevät esimerkiksi käyttäjien tietoja. Joka tapauksessa ylläpitäjän näkymän sisällä turvallisuuden lisääminen on vain viimeisen turvaverkon rakentamista, koska ylläpitäjän oikeuksilla epäpäteväkin väärinkäyttäjä saa aikaan huomattavaa tuhoa.

6.1.2 Ekes KehäV

Kehä V sisällöntuotannosta vastaa ylläpitäjä. Mikäli joku ulkopuolinen pääsee syöttämään tietoja näihin näkymiin, niin pahin vahinko on jo tapahtunut. Väärinkäyttäjällä on kaikki valta muokata sivujen sisältöä haluamansa kaltaiseksi ja tehdä sillä pitkälti mitä haluaa. Jos tähän tilanteeseen on jouduttu, niin lisäriskinä on enää tämän sovelluksen hyödyntäminen hyökkäyksiin jotka kohdistuvat toisille sivuille tai suoraan sivuston käyttäjiin, ei niinkään tälle sivustolle. Sivujen julkisella puolella on yksi ainut kohta joka hyväksyy syötteitä käyttäjältä, tämä on kirjautuminen sivustolle. Tietenkään tämä ei siis ole ainut mahdollinen haavoittuvuus sivulla, mutta oletettavasti tämä on ensimmäinen johon mahdollinen hyökkäys suuntautuisi.

6.2 Suunnittelu

Testauksen suunnittelu on tärkein vaihe onnistuneeseen lopputulokseen pääsemiseksi. Huolimattomasti suunniteltu ja toteutettu testaus saattaa olla pahempi kuin tekemätön testaus, koska tämä jättää valheellisen turvallisuudentunteen, että sivu on nyt tietoturvallinen, eikä välttämättä kiinnitä huomiota mahdollisiin murronjälkiin.

Suunnittelu on syytä aloittaa sivuston kartoittamisella ja jokaisen alasivun erillisellä tarkistelulla. On syytä merkitä, onko sivuilla tietojen syöttökenttiä, mitä syötteitä koodi kuuntelee(myös html koodin seassa olevat hidden kentät) ja onko alasivulle pääseminen rajattu. Kun kaikki haavoittuvuuskohdat on kirjattu, niin kannattaa vielä tarkistaa koodi mahdollisten globaalien parametrien varalta. Näitä ovat siis kaikki GET tai POST metodisyötteitä jotka aikaansaavat efektejä riippumatta alasivustosta. Tämän ei pitäisi olla ongelma, jos koodi on hyvin organisoitua ja suunnitteluvaiheessa on muistettu tietoturva.

Mikäli testausta tehdään jälkikäteen tai jopa eri henkilön kuin sivuston toteuttaja toimesta, niin mielestäni suunnittelun pitäisi yhä alkaa koodin tarkistelulla. Ensin on selvitettävä joka paikassa, mitä sivusto kuuntelee ja että sen toteuttamisessa on käytetty tietoturvallisia tekniikoita. Joitain haavoittuvaisuuksia voi olla hyvin hankala huomata testaamalla tai ne saattavat aueta vasta myöhemmin(kts. esim. Magic Quotes kohdasta 3.2.2.2 Metodit, joita ei tulisi käyttää)

6.3 Hyökkäysvektoreiden valitseminen

Tässä työssä käsitellyt hyökkäysvektorit kohdistuvat nimenomaan sivuston dynaamisen osion ja sen haavoittuvuuksien kartoittamiseen. Eli palvelinpuolen haavoittuvuuksiin en ota mitään kantaa. Tämä johtuu ensinnäkin siitä, että molemmat sivustot sijaitsevat vuokratilassa erillisien yritysten palvelimilla. Hyökkäysvektorin joita käsittelen tässä, ovat suoraan sivustoon kohdistuvia hyökkäyksiä.

Pohjimmiltaan käsiteltyjen hyökkäysten päätyypit ovat URL hacking, SQL injektioita ja erilaiset HTML koodin läpiviemistavat. Kahdessa muussa hyökkäystyypeissä testattavat lähestymistavat ovat varsin samankaltaisia. Näissä hyökkäys voi tulla joko GET tai POST protokollaa hyväksikäyttäen, se voidaan aikaansaada joko sivuilla olevalla lomakkeella tai lähettää sivuston ulkopuolelta. Joskus molemmat lähestymistavat on syytä ottaa huomioon. Kolmas haavoittuvuus lähtee siitä valheellisesta ajatuksesta, että jos linkkiä ei ole, niin sivulle ei pääse. Vaikka tätä ei ajattelisikaan tietoisesti, niin riippuen arkkitehtuurissa on valitettavan yksinkertaista aikaansaada haavoittuvaisuus jossa esimerkiksi väärien käyttäjäryhmien edustajat pääsevät käsiksi rajoitettuihin sivuihin, tai käyttäjät pääsevät näkemään toistensa tietoja vaihtamalla vain id numeronsa osoitekentässä.

Testin vektorit on vielä suotava valita siten, ettei ensimmäinen testaus korruptoi koko tietokantaa tai muuten lamaannuta koko sivua. Vaikka tällä päästäisiinkin realistiseen testitulokseen niin, ei tämä ole sen enempää tarpeellista kuin tehokastakaan. Samat haavoittuvuudet kun voidaan havaita vähemmälläkin vaivalla.

Erilaiset tietokantajärjestelmät, ohjelmointikielet ja näiden versiot on syytä huomioida testausta suunnitellessa. Mielestäni hyvä lähestymiskohta on, että kuvitteellisella väärinkäyttäjällä on käytössään kaikki tieto ohjelmasta. Sen varaan jättäytyminen, että palvelussa käytettyjen muuttujien- tai taulujen nimet eivät saa joutua vääriin käsiinhän, luo itse asiassa palveluun uuden haavoittuvuuden.

MySQL injektio lauseeksi valitsin seuraavanlaisen lausekkeen

1';INSERT INTO np_users SET login='a';--
1';INSERT INTO ekes_user SET login='a';--

Tämän lausekkeen idea on seuraavanlainen. Se alkaa numerolla yksi joka välitetään nykyiseen tietokantakyselyyn, sen jälkeen seuraa heittomerkki joka katkaisee tekstikentän ja puolipiste joka lopettaa kyselyn. Tämän jälkeen aloitamme uuden komennon tietokantaan joka on lisäämislauseke INSERT INTO. Oletamme automaattisesti, että hakkeri on arvannut tai tietää tietokantamme taulun nimen, sitten ihan perus mysql:lää jatkaen SET ja login='a'; jolla lisäämme käyttäjän annettuun tauluun ja lopetamme komennon, tässä testissä salasanan antaminen on tarpeetonta, käyttäjää kun ei ole tarpeen pystyä nyt käyttämään, todellisessa tilanteessa tämä olisi luonnollisesti toisin. Lopetamme lausekkeen kahdella miinusviivalla, tämä on mysql kommettimerkki. Tällä tavoin kommentoimme alkuperäisen kyselyn loppurivin. Näin saamme pois mahdolliset where yms. lauselmat jotka olivat alkuperäisessä kyselyssä.

Pahimmassa tapauksessa tämä saattaa korruptoida koko alkuperäisen taulun uuden käyttäjän lisäämisen lisäksi. Tämä voi tapahtua jos kyseinen lauseke menee täysin suojaamattomana UPDATE rakenteeseen. Tällöin kaikkien rivien kyseinen arvo muuttuu numeroksi yksi, koska where ehto on kommentoitu pois. Tämä on siis syytä pitää mielessä.

XSS haavoittuvuuksien valitsin ensin seuraavan lausekkeen variantteineen:

<script>alert("XSS")</script>.

Erilaiset variantit ovat sama lauseke jossa lainausmerkit on korvattu heittomerkeillä sekä lisättä kenoviivoja \ joilla on tarkoitus poistaa ohjelmakoodin lisäämiä viivoja. Etuna tässä rivissä on, että se on ehdottoman visuaalinen. Mikäli tämä rivi menee läpi, niin se ei jää huomaamatta. Tämän huomaa sen varmasti myös niiltä sivulta jonne sen läpiviennin saattaisi muuten unohtaa.

Toinen testauksessa käyttämäni rivi on

";!--"<XSS>=&{()}

Tästä rivistä näkee suoraan erilaisten merkkien käyttätymistä sekä hakasuluissa olevien komentojen tilanteen, vaikkakaan se ei ole yhtä visuaalinen kuin alert rivike.

6.4 Läpikäynti

Konkreettisimmillaan hyviä tapoja testata tietokannan haavoittuvaisuuksia on ajaa tietokantaa vastaan injektiohyökkäys jokaisen lomakkeen jokaiseen kenttään. Samalla logiikalla kannattaa testata myös HTML puolen haavoittuvuuksia. URL:ään jääneitä aukkoja on mielestäni tehokkainta testata kokonaisuudessaan käsityönä. Ensin käydään yksi taso läpi ja kirjataan kaikki urlit ylös, sitten vaihdetaan seuraavalle tasolle. Kun testausta varten on kerätty tarpeeksi kattava lista linkkejä, niin käydään nämä linkit lävitse selaimella erilaisissa tiloissa.

XSS hyökkäysten testauksessa kannattaa ottaa suunnitteluvaiheessa toteutettu lista kaikista alasivuista ja käydä sivukerrallaan kaikki sivut läpi valituilla hyökkäysriveillä. Tietokantahaavoittuvuuksien ja xss haavoittuvuuksien testaamista samaan aikaan kannattaa harkita. Toisaalta tämä säästää sivuston kaksinkertaiselta läpikäynniltä, mutta tietokantahaavoittuvuuksien testaamiseen liittyy sivulle injektoinnin jälkeen vielä yksi työvaihe. Jokaisen injektoinnin jälkeen on syytä tarkistaa, ettei mitään ole päässyt lävitse ja jäänyt kantaan.

Mikäli haavoittuvuuksia löytyy, niin saattaa olla parempi, ettei niitä heti syöksytä korjaamaan ainakaan kesken testausta. Mielestäni on tehokkaampaa ensin testata ja kirjata kaikki ylös. Sivun kokonaisvaltaisen testaamisen suorittamisen jälkeen on testaajalla kokonaiskuva kaikista haavoittuvuuksista. Tämän jälkeen tiedät, mitkä ovat yksittäisiä ongelmia ja mitkä taas koskevat koko sivustoa. Samalla estetään se, ettei pääse käymään niin, että testauksen ja korjauksen välissä jokin kohta jää huomioimatta.

6.5 Jälkihoito

Kun kaikki on testattu ja mahdolliset aukot on korjattu, niin on syytä käydä sivusto uudestaan läpi ja poistaa kaikki testauksesta syntyneet rivit. Ennen kaikkea tämä koskee tietokantaa, varsinkin mahdollisen injektion aikaansaamat rivit saattavat jo itsessään olla vaarallisia.

Esimerkkinä voidaan teoretisoida, että jos edellä mainittuja alert-scriptejä olisi mennyt järjestelmään ja niiden syöttäminen olisi korjausvaiheessa estetty syötteen tarkistelulla ennen kantaan tallennusta. Jo kantaan menneet alertit olisivat jo tarkistelun tuolla puolen, vaikkakin vaarattomina niin yhä toimivina. Nämä saattaisivat aiheuttaa paljonkin sekaannusta tullessaan

esille. Pahimmassa tapauksessa, jos testausta varten on luotu testikäyttäjä ja se on testauksen jälkeen poistettu suoraan teitokannasta, niin samalla on vapautunut käyttäjän id. Kun tämä kiertoon vapautunut id tulee uudelle käyttäjälle, niin viestit jotka testaaja on unohtanut siivota voivat periytyä käyttäjälle jolla on kyseinen id.
Tietenkään tämä ei voi tapahtua, jos tietokannan taulujen tietojen välillä on linkityksiä (esim. on delete:cascade foreing key)[16 s. http://dev.mysql.com/doc/ refman/5.0/en/innodb-foreign-key-constraints.html].

7 MUITA HUOMIONARVOISIA ASIOITA

7.1 Open Source tietoturvan kannalta

Linux, Apache, MySQL ja PHP ovat kaikki Open Source pohjalla kehitettäviä ja julkaistuja järjestemiä [6 s.11.].
Open Sourcen eli avoimen lähdekoodin vaikutuksista tietoturvaan on ollut keskustelua. Amerikkalainen lähestymistapa, johon Microsoftin kaltaiset yhtiöt helposti vetoavat lähtee ajatuksesta, että avoimen lähdekoodin järjestelmistä voivat väärinkäyttäjät helposti löytää haavoittuvaisuuksia ja jopa rakentaa niitä itse. Toisaalta antaa avoin lähdekoodi käyttäjälle mahdollisuuden itse todentaa ohjelmiston toimintaa. Erilaiset linux ja unix-järjestelmät on suunniteltu palvelin käyttöön, joten tietoturva ominaisuuksia on mietitty alusta lähtien, ja avoimen lähdekoodin järjestelmissä erilaisia tietoturvallisuus laajennuksia on yleisesti saatavilla, ilman huomattavia lisäkustannuksia. [7 s.29; 1 s.247-250.].

7.2 Inhimillinen tekijä

Riippumatta siitä, kuinka loppuun asti sivun tietoturva on hiottu, on aina olemassa haavoittuvaisuuksia ja aukkoja, joita ei voi paikata. Nämä haavoittuvuudet johtuvat kompromisseista, joita joudutaan tekemään tavallisen tietokoneenkäyttäjän ammattitaidon ja käyttötottumusten vuoksi, tehokkaan toimivuuden, hyvän käytettävyyden sekä houkuttelevan ulkoasun ja ympäristön vuoksi. Käyttäjään ei voi luottaa, on parasta varautua pahimpaan. [4 s.35.]. Osansa ongelmasta muodostuu käyttäjistä, jotka eivät huolehdi tietoturvasta omalta osaltaan, eli heillä on sama salasana useissa eri palveluissa, tietoturvaton salasana tai saastunut kone. Toinen osa ovat oikeasti vihamieliset käyttäjät, jotka yrittävät aktiivisesti hyödyntää haavoittuvuuksia. Perinteiseen tietoturvaan perehtyneetkään eivät aina suhtaudu tarpeeksi vakavasti tai tiedä tarpeeksi Web-puolen haavoittuvuuksista, jotta osaisivat olla varuillaan, Web puolen haavoittuvuudet ovat kuitenkin suhteellisen uusi kenttä. [3 s.2-3.].
Vaikka suurin osa verkossa olevista vaaroista olisikin vältettävissä ihan pelkällä maalaisjärjellä, niin ihmisiä menee huijareiden ansaan päivittäin. [1 s.49-51.].

7.2.1 Khalastelu eli Phising tyyppiset hyökkäykset

Khalastelulla eli phissingillä tarkoitetaan tietojen urkkimista käyttäjiltä erilaisten näköissivustojen avulla. Perusideana on että käyttäjä huijataan khalastelusivulle, jossa käyttäjä luulee olevansa turvallisessa ympäristössä ja luovuttaa kriittistä informaatiota, vaikkapa yrittää kirjautua sisään palveluun. Khalastelu perustuu puhtaasti käyttäjien huijaamiseen, sitä vastaan on hyvin vaikeaa yrittää taistella teknisin keinoin. Vaikka khalastelu ei ole

sivustolla oleva haavoittuvuus, eikä siten varsinaisesti tietoturva-aukko PHP-koodissa on se silti mainitsemisen arvoinen, koska se saattaa kyseenalaistaa sivuston tietoturvallisuuden. [1 s.273-274.].

7.2.2 Salasanojen ongelma

Helposti muistettava salasana ei yleensä ole tietoturvallinen. Valitettavan usein käyttäjien salasanat eivät ole tietoturvallisia ja tämän kanssa on vain elettävä. Varsinkin laskutehon kasvu asettaa paineita salasanoille aina vaan enemmän, alamme jo olla tilanteessa, jossa oikeasti tietoturvallinen salasana on monimutkaisuudeltaan ja pituudeltaan kohtuuton käyttäjän muistettavaksi. Tätä on yritetty ratkaista älykorteilla ja salasanalistoilla, mutta nämä ratkaisut luovat samalla uusia ongelmia tarvittavien lisälaitteiden ja vääriin käsiin joutuneiden listojen muodossa.
[15 s. http://vrk.fineid.fi/.].

Salasanalistaominaisuus on liitetty myös moderneihin selaimiin. On huomattavasti helpompaa käyttää eri salasanojen eri palveluihin, jos vaikka selain muistaa salasanat puolestasi, ja tämän ominaisuuden hyödyntämistä suositellaan myös alan kirjallisuudessa [8 s.171-172.].

Valitettavasti tälläkin ominaisuudella on kääntöpuolensa. Huolimatta selainten kehittäjien ponnisteluista väärinkäytösten ehkäisemiseksi, niin sopiva XSS, joka esiintyy sivun koodina, saattaa huijata selainta luovuttamaan käyttäjätunnuksen ja salasanan. [3 s. 220-223.].

8 JOHTOPÄÄTÖKSIÄ

Verkkopalveluiden toteuttaminen on kehittynyt vähitellen, kaikki on yritetty tehdä mahdollisimman yksinkertaiseksi ja helpoksi toteuttaa. Tietoturvanäkökanta on valitettavasti laahannut perässä.

Tämä sysää vastuun täysin kehittäjän niskaan, PHP:lla on erittäin helppoa toteuttaa tietoturvattomia sovellutuksia. Oikotietä onneen ei ole. Valmiit työkalut ovat toistaiseksi osoittautuneet ongelmallisiksi sillä ne luovat yleensä omat ongelmansa, myöskään valmiit alustat, joiden varaan voi rakentaa sivunsa, eivät ole aina varmoja. Tuottavuuden nimessä samaa ohjelmakoodia yritetään hyödyntää aina vaan uudestaan jopa kokonaisten sivu ja foorumipohjien muodossa. Huolimattomuus väärässä paikassa saattaa tehdä suunnattoman määrän verkkopalveluita haavoittuvaisiksi samalle hyökkäykselle. Tämänkaltainen massa-haavoittuvuus voi pahimmillaan olla haavoittuvainen automaatiotyökaluille joka hyökkää kerralla satoihin tuhansiin verkkosivuihin ja aikaansaa uskomattomia tuhoja ympäriinsä. Viimeistään ongelmia tulee, jos tietoturva-asioita tuntematon päättää ”ihan vähän” korjailla tai lisäillä ominaisuuksia omaa sivuaan varten. Jos tämänkaltainen toiminnallinen innovaatio leviää laajempaan käyttöön, niin lopputulos voi olla hyvinkin surullista.

Nämä tietoturvaan liittyvät haavoittuvuudet eivät ole sidoksissa PHP:hen, vaan ne ovat koskevat käytännössä kaikkia verkkosivujen toteuttamistekniikoita. Niin pitkään kun selaimet tukevat html:ää ja javascriptiä ovat näitä koskevat kohdat ajankohtaisia. Erilaiset tietokantavetoiset palvelut ovat mielestäni vielä lapsenkengissä, aina kun tulee uusi palvelu niin se venyttää mahdollisuuksia aina vaan laajemmaksi, vaikka kyseessä on vanhan tekniikan tehokkaampaa soveltamista. Tietokantapalveluiden uudet hyödyntämismahdollisuudet lisäävät toiminnallisuutta mutta samalla syntyy uusia mahdollisuuksia väärinkäyttää tai vaurioittaa verkkopalveluita.

Kilpailumainen kehittymisnäkökohta verkkosivujen kehittämisessä lisää paineita tietoturvaan, jos suuret yksiköt tahtovat markkinoiden teknisesti kehittyneimmän palvelun. Verkkopalveluita saatetaan toteuttaa hyvinkin kokeellisilla tekniikoilla ja suurella kiireellä. Tämänkaltainen kehittäminen ei ole mahdollista jolle ohjelmoijalla ole vapautta ja valtaa käyttää järjestelmää tavoilla jota sen kehittäjät eivät ole osanneet edes ajatella. Tämä ohjelmoijalle annettu vapaus sysää vastuun ohjelmoijan harteille. Meillä on monenlaisia palvelinpuolen rajoituksia ja suojauksia jotka estävät pahimpia virheitä, mutta ilman rampauttavaa rajoittamista jää pääosa palvelinpuolen tietoturvasta täysin sovelluksen toteuttamiseen käytetyn ohjelmakoodin varassa. Lopputuloksen kannalta tämän tekee ongelmalliseksi vielä se, että yksikin aukko riittää. Tästä johtuen ajattelutapa ”Pääosin se on kunnossa” saattaa muodostua hyvin vaaralliseksi. Aukkoja kun saattaa syntyä käytännössä mihin tahansa sivuilla hyödynnettyyn elementtiin.

Tietoturva onkin syytä pitää mielessä kaikkien eri osien kohdalla, ja selvittää eri asiat, joista tulee huolehtia. Erilaiset työkalut ja palvelut kun aikaansaavat erilaisia haasteita. Quicktime video tai Flash-multimediapalvelu saattaa asettaa hyvin erilaisia haasteita kuin AJAX-chat palvelu, mutta molemmat saattavat olla haavoittuvaisia samanlaisille XSS-hyökkäyksille.

Erilaisia hyökkäyksiä on lukuisia ja niiden tavoitteet saattavat olla hyvinkin erilaisia. Mielestäni suoraviivaisin väärinkäytös on yritys tuhria sivuja jotenkin, kirjoitella sinne omia tekstejään ja vaihdella kuvia. Tämänkaltainen väärinkäytös saattaa aikaansaada taloudellisia tuhoja mutta yleensä tuhot ovat hyvin rajoitettuja ja lyhytkantoisia. Tämänkaltainen väärinkäytös huomataan varmasti yleensä hyvin pian ja vahingot rajautuvat heti kun palvelu saadaan ajettua alas. Tosin palvelun sulkeminenkaan ei aina ole ilmaista, koska suljettu palvelu ei ole tuottavassa käytössä. Lukuisat väärinkäyttäjät yrittävät toimia havaitsemattomasti, kerätä salasanoja palveluun, nämä kerätyt salasanat saattavat sitten olla rikollisen toiminnan välikappaleena tai ne saatetaan vaikkapa vuotaa internetiin. Tarkoituksena näissä lienee vain väärinkäyttäjän oman erinomaisuuden osoittamisyritys, itsetunnon kasvattamista ja arvostuksen hakua. Verkkosivuilla on nykyään paljon rahanarvoista informaatiota, mutta kun kaikki tapahtuu kotona ja omalla koneella, niin saattaa rikollisen ja laillisen toiminnan raja hämärtyä nopeasti. Esimerkiksi sisäverkko työpaikalla saattaa houkutella kokeilemaan tylsänä työpäivänä, ja lopputuloksena koneella leikkiminen voi johtaa vakaviin seurauksiin jos toteuttaminen on hoidettu epäasiallisesti. Sitten on olemassa vielä varsinainen rikollinen aines jonka ensisijainen tarkoitus on taloudellinen tai muu hyötyä. Teollisuusvakoilu, vaikka onkin romantisoitu nykymediassa, saattaa olla taloudellisesti hyvin merkittävää.

Henkilökohtainen mielipiteeni on, ettei verkossa ole niin merkityksetöntä verkkosivua, ettei sitä kannattaisi suojata (siihen en ota kantaa ovatko kaikki verkossa olevat sivut verkkoon laittamisen arvoisia, ainoastaan tietoturvaan). Merkityksettömyys voi osaltaan tehdä sivusta houkuttelevamman väärinkäyttäjälle ja toisaalta merkityksettömälle sivulle saattaa syntyä jossain vaiheessa merkityksellistäkin käyttöä. Kannattaa myös varautua siihen, että merkityksettömälle sivulle tehty koodi saattaa jossain elinkaarensa vaiheessa päätyä uudelleenkäytetyksi jossain muualla.

Iso osa tässä työssä esitellyistä haavoittuvuuksista koskevat erilaisia syötteitä. Syötteitä vastaan suojautuminen voidaan toteuttaa kootusti, tehdään yksittäinen aliohjelma joka poistaa tavalla tai toisella syötteestä vaaralliset merkit. Tämän jälkeen on vain ensisijaisen tärkeää hyödyntää tätä aliohjelmaa aina. Etuna on vielä, että mahdolliset muutokset ja korjaukset on helppoa toteuttaa kootusti. Jossain tilanteissa saattaa olla edullista palauttaa merkit takaisin alkuperäiseen muotoonsa ennen esittämistä. Esimerkkinä tästä on heittomerkkien kahdentaminen SQL injektiota vastaan. Kun kyseinen syöte palautetaan lomakkeeseen muokkaamista varten, eikä kaksinkertaisia merkkejä ole poistettu, niin jokaisella tallennuksella syntyy lisää merkkejä.

Oman mausteensa tietoturvaan aikaansaa käyttäjien kirjo. Lähes joka palvelussa on syytä olettaa, että käyttäjiä löytyy aina tietotekniikan ammattilaisista ensikertalaisiin verkonkäyttäjiin. Tällaisella oletuksella rakennettu sivusto suojaa kaiken käyttäjien kokeiluilta, mutta ei odota käyttäjien osallistuvat varsinaisiin tietoturvallisuusratkaisuihin. Täydellisenkään tietoturvan ei tarvitse häiritä käyttäjän käyttökokemusta, mutta koska käyttäjään ei koskaan uskalla luottaa, niin tietty varovaisuus on aina paikallaan.

LÄHTEET

Painetut kirjat

[1] Järvinen – Paranna tietoturvaasi, WS Bookwell 2006

[2] Järvinen – Tietoturva & yksityisyys; WS Bookwell 2002

[3] Grossman, Hansen, Petkov, Rager, Fogie . XSS Attacks, Syngress 2007

[4] Alshanetsky - phplarchitect's Guide to Security, Marco Tabini & Associates, Inc. 2005

[5] Lavin – Object-oriented PHP : concepts, techniques, and code, No Starch Press, Inc. 2006

[6] Heinisuo, Rauta - PHP ja MySQL, Talentum 2007

[7] Kuutti, Rantala- Linux, WS Bookwell 2007

[8] Boström - Kotimikron tietoturva; Talentum 2003

[9] Järvinen - Tietotekniikan termit; WSOY 1996

Sähköiset lähteet

[10] PHP: Hypertext Preprocessor – http://www.php.net

[11] Documentation: Apache HTTP Server - http://httpd.apache.org/docs

[12] suPHP - http://www.suphp.org/

[13] TIOBE Software: Tiobe Index
– http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[14] CERT-FI - http://www.cert.fi

[15] Väestörekisterikeskus - http://vrk.fineid.fi/

[16] MySQL Developer Zone - http://dev.mysql.com/

LIITE 1

OTE NÄYTÖNPAIKAN PORTFOLION PROJEKTISUUNNITELMASTA

Toteutettavat muutokset

1. Työntekijätunnusten päivitys
   - 1.1 Tunnusten muokkaus siten, että käyttäjä pystyy valitsemaan **yksittäiset** työntekijät, jotka pääsevät tarkastelemaan valittuja tietoja.
   - 1.2 Tämä muutos tehdään vain, jos se on mahdollista toteuttaa hallitusti käytettävissä olevilla resursseilla, tekniikoilla ja tuotantoaikataulun puitteissa.
   - 1.3 **Työntekijätunnusten rakentaminen uudelleen on päivitysprojektin pääpainopiste ja muita päivitystöitä voidaan jättää tarvittaessa toteuttamatta.** Mahdollisista toteutukseen liittyvistä muutoksista sovitaan asiakkaan kanssa erikseen.
2. Etusivun muutokset seuraavasti:
   - 2.1 Etusivun leipätekstin muokkaus asiakkaan toimittaman tekstin pohjalta.
   - 2.2 Sähköpostiosoitteiden muuttaminen leipätekstiksi tai kuvaksi
   - 2.3 Rekisteriselosteeseen osoittavan linkin lisääminen etusivulle. Tilaaja toimittaa lisättävän rekisteriselostetekstin.
3. Päävalikon muokkaukset (valikko sivun vasemmassa laidassa)
   - 3.1 CV siirretään osaksi Oma osio-kohtaa.
   - 3.2 CV:lle luodaan oma kuvake.
   - 3.3 CV-osio nimetään uudelleen Ansioluetteloksi.
   - 3.4 Päävalikkoon lisätään Viestit-osio CV:een linkin tilalle.
4. Viestit
   - 4.1 Työtekijätunnuksilla kirjautunut työntekijä voi lähettää viestejä käyttäjille, jotka ovat valinneet hänet työntekijäkseen.
   - 4.2 Peruskäyttäjänä kirjautunut voi lähettää viestejä valitsemilleen työntekijöille.
     - Tähän ratkaisuun päädyttiin, jotta käyttäjä/työntekijä voi aina vastata saamiinsa viesteihin. Eli jos käyttäjä voi lähettää viestiä työntekijälle, tulee työntekijän aina voida vastata siihen.
5. Lomakkeiden päivitys
   - 5.1 Elämäntilanne-kyselyjen järjestys, kysymysten sanamuodot ja uudet vastausvaihtoehdot toteutetaan tilaajan toimittamien ohjeiden mukaan.
   - 5.2 Tämän muutoksen myötä käyttäjien aiemmin tekemien kyselyjen vastaukset poistetaan WWW-palvelusta.
6. Aikajana

6.1 Aikajanan ohjeteksti muokataan uudelleen tilaajan toimittaman tekstin mukaan.
6.2 Aikajanaan lisätään sekä menneitä että tulevia vuosia.

7. Verkostokartta-sovellus
   7.1 Flash-sovellus päivitetään tilaajan ja toimittajan yhdessä määrittelemien tarpeiden mukaan
8. Tukiverkko osio
   8.1 Nimetään uudelleen: Yhteystietopankki
   8.2 Yhteystietopankkiin päivitetään yhteisöjen ja toimijoiden nimet tilaajan toimittamien tekstien pohjalta.
   8.3 Myös toimijoiden järjestys listalla muokataan
   8.4 Lisätään uusi kenttä Koulut ja oppilaitokset
9. Päiväkirja
   9.1 Päiväkirjaa muokataan siten että käyttäjä voi tehdä päiväkirjamerkinnän myös menneen päivämäärän kohdalle.
10. Oppimispäiväkirja
   10.1 Oppimispäiväkirjan kentät nimetään tilaajan toimittamien tekstien mukaan
   10.2 Mikäli käyttäjä on aiemmin tehnyt merkintöjä oppimispäiväkirjaan, oletuksena lomake tarjoaa viimeksi täytettyjä tietoja
   10.3 Osiota muokataan siten että käyttäjä voi lisätä oppimispäiväkirjamerkinnän menneen päivämäärän kohdalle
11. Omat tiedostot
   11.1 Osioon lisätään sanallinen tiedostojen lisäämisohje tilaajan toimittaman tekstin pohjalta
12. CV
   12.1 CV:n otsikoksi muutetaan Ansioluettelo
   12.2 Ansioluetteloon lisätään kenttiä
   12.3 Ansioluettelon olemassa oleviin kenttiin mahdollistetaan merkinnät ilman päivä- ja kuukausi-informaatiota
13. Yhteystiedot
   13.1 Yhteystiedot-osion teksti muokataan tilaajan toimittaman tekstin pohjalta
14. Linkit
   14.1 Linkit osion sisältöä voidaan muokata tarvittaessa