VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Juha Nevala

VERKKOSOVELLUSTEN TIETOTURVA

Tietojenkäsittely
2017

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Juha Nevala
Opinnäytetyön nimi	Verkkosovellusten tietoturva
Vuosi	2017
Kieli	suomi
Sivumäärä	47
Ohjaaja	Antti Mäkitalo

Opinnäytetyön tarkoituksena oli kerätä kirja- sekä verkkolähteitä apuna käyttäen tietoa yleisimmistä haavoittuvuuksia PHP-pohjaisissa verkkosivuissa, ja kehittää tiedon pohjalta tietoturvalliset verkkosivut. Työllä ei ole toimeksiantajaa, joten työn kohteeksi valikoitui oma harrastusmielessä kehitetty sivusto.

Opinnäytetyötä varten luotu sivusto on ohjelmoitu käyttäen PHP-ohjelmointikieltä ja sen sisältö on tallennettu MySQL-tietokantaan. Ulkoasun toteuttamisessa käytössä olivat HTML, CSS sekä JavaScript.

Keskeisiä tutkimusalueita olivat yleisimmät haavoittuvuudet verkkosivuissa, kuten esimerkiksi SQL-injektio sekä XSS-hyökkäykset. Työssä esitellään ratkaisuja näiden välttämiseksi.

Työtä tehdessä tuli selväksi, että PHP-kielellä on aloittelijankin mahdollista toteuttaa tietoturvallisia verkkosovelluksia, ja että useimpien haavoittuvuuksien välttäminen on loppujen lopuksi varsin yksinkertaista.

Avainsanat verkkosivut, php, tietoturva, mysql

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Juha Nevala
Title	Web Application Security
Year	2017
Language	Finnish
Pages	47
Name of Supervisor	Antti Mäkitalo

The objective of this thesis was to use literature and web sources to gather information on the most common vulnerabilities in PHP-based websites, and use this knowledge to develop a secure website.

The website developed for the project was programmed using PHP, and it used a MySQL database solution. The layout was created using HTML, CSS and JavaScript.

The thesis aimed to present some simple to implement solutions for avoiding some of the most common web application security issues and attack vectors.

Keywords websites, php, data security, mysql

SISÄLLYSLUETTELO

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Työn tavoitteena oli toteuttaa verkkosovellus ja perehtyä sitä myötä verkkosovellusten tietoturvaan. Työ keskittyy pääosin PHP-sovelluksiin, joissa käytetään MySQL-tietokantaa.

Projektityönä tein verkkosovelluksen, jonka sisältö koostuu lähinnä automaattisesti kerätystä tiedosta, mutta myös käyttäjillä on mahdollisuus lisätä sisältöä. Tavoitteena oli tehdä sovelluksesta tietoturvallinen, ja samalla oppia enemmän tekniikoista, joilla sovelluksen tietoturvaa voi parantaa.

1.1 Työn tausta

Työtä varten tehty verkkosovellus on WWW-sivusto, jota Steam-jakelualustan käyttäjät voisivat käyttää apuna esim. sellaisten pelien löytämiseen, jotka eivät enää ole virallisesti ostettavissa. Sivusto on jo ollut yli satahenkisen Steam-pelien keräilijöistä koostuvan yhteisön käytössä, ja palaute on ollut positiivista. Työllä ei ole toimeksiantajaa.

1.2 Tavoitteet ja tutkimuskysymykset

Opinnäytetyöni ja tutkimukseni tavoitteena oli oppia suunnittelemaan ja toteuttamaan tietoturvallinen verkkosivusto ja sitä ohjaava järjestelmä. Valmiita sisällönhallintajärjestelmiä työssä ei käytetty, koska halusin oppia tekemällä jotain omaa, enkä vain muokata valmista tuotetta. Jos kuitenkin tekisin projektin uudelleen, pyrkisin käyttämään jotain ohjelmistokehystä koodin parantamiseksi, esimerkiksi Laravelia.

Opinnäytetyön päätavoite oli kartoittaa projektityön tekemisen ohella yleisimpiä tietoturvariskejä, ja käytännöllisiä ratkaisuja niiden välttämiseksi. Tarkasteluun valikoituneet tietoturvariskit ovat sellaisia, joita esiintyy eriävissä määrin käytännössä millä tahansa verkkosivulla toteutukseen käytetystä ohjelmointikielestä riippumatta.

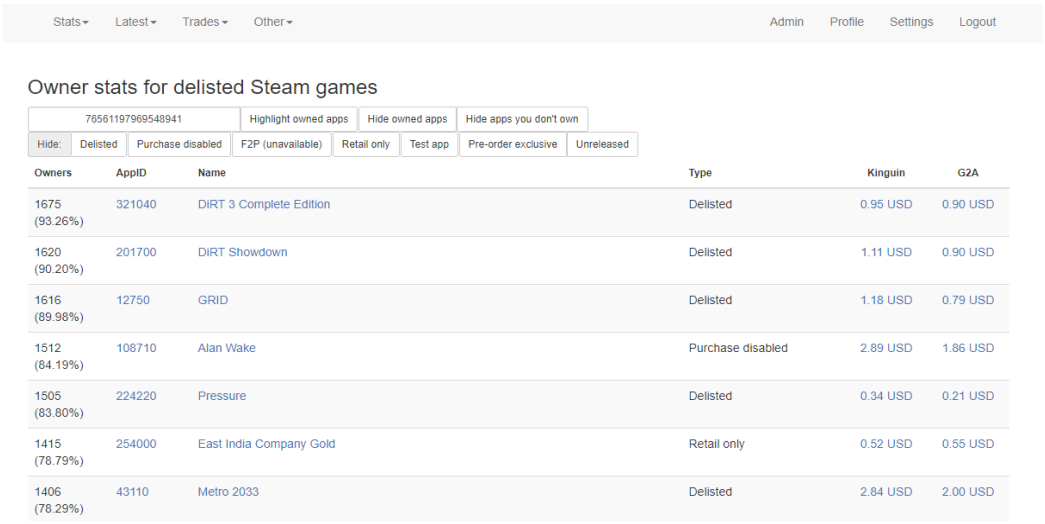
2 VAATIMUKSET

Tässä luvussa käsitellään sivuston ulkoasulle ja ominaisuuksille asetettuja vaatimuksia.

2.1 Ulkoasu

Sivuston ulkoasu on toteutettu pääosin käyttäen Bootstrap-elementtejä. Responsiivisuus on kytketty pois päältä, koska Google Analyticsin mukaan vain alle prosentti sivuston käyttäjistä käytti sitä mobiililaitteella, ja responsiivisuus aiheutti turhia ongelmia työpöytäversion ulkoasulle.

Tavoitteena ei ollut saada aikaiseksi mitään näyttävää, joten sivusto koostuukin tällä hetkellä pääosin tietoa sisältävistä tauluista, joista tarvittava tieto löytyy nopeasti ilman turhaa selailua. Sivuston värimaailma koostuu oman mieltymyksen mukaisesti lähinnä tummista väreistä, jotka eivät rasita silmiä. Tässä työssä olevat kuvat on otettu käyttäen musteen säästämiseksi tavallista valkoista Bootstrap-teemaa.



Kuva 1. Sivuston tämänhetkinen etusivu.

Kuvassa 1 näkyy miltä sivusto näyttää, kun sille saavutaan. Varsinaisella etusivuksi suunnitellulla sivulla käyttäjälle voisi kertoa esimerkiksi enemmän sivuston

käyttötarkoituksesta ja muusta, mutta koska se on vielä työn alla, käyttäjä ohjataan suoraan tietoa sisältävään taulunäkymään.

2.2 Taulunäkymän ominaisuudet

Käytettävyyden kannalta oli tärkeää, että tauluissa olevan tiedon voisi helposti järjestää haluamaansa järjestykseen. Tämän toteuttamiseksi käytin Tablesorter jQuery-pluginia. Plugin mahdollistaa minkä tahansa taulun nopean uudelleenjärjestämisen selaimessa ylätunnistetta klikkaamalla. (Bach, n.d.)

Taulun yläpuolella sijaitsevilla painikkeilla käyttäjä voi mm. piilottaa tai korostaa omistamansa pelit tai näyttää vain tiettyihin kategorioihin kuuluvat pelit.

Jokaisen pelin kategoria-ID on sijoitettu tauluun käyttäen data-attribuutteja (data-attributes). Data-attribuuttien avulla HTML-elementteihin voidaan sijoittaa tietoa joka ei näy suoraan käyttäjälle, mutta on skriptien käytettävissä ilman ylimääräisiä http-pyyntöjä. (W3schools, n.d.) Kategorioiden piilottaminen on toteutettu yksinkertaisella jQuery-skriptillä, joka käy läpi data-attribuuttien arvot ja piilottaa (tai näyttää) ne rivit, joissa kategorian ID vastaa haluttua.

Käyttäjän omistamien pelien korostus toimii siten, että palvelimelle lähetetään jQueryä käyttäen AJAX-pyyntö ja palvelin vastaa siihen JSON-muotoisella pelilistalla. Lista käydään skriptissä läpi, ja pelin ID:tä vastaaviin riveihin lisätään CSS-luokka, joka värittää rivit vihreäksi.

2.3 Sisällön syöttäminen

Sisällön helppo ja nopea syöttäminen oli olennainen vaatimus järjestelmälle, jotta tiedon ylläpito ei olisi liian aikaavievää. Sivustolle oli lisättävä vain admin-oikeuksilla käytössä olevat toiminnot, joilla tiedot voi halutessaan päivittää manuaalisesti, tai poistaa niihin pääsyn kokonaan.

Jokaisesta tietokannassa olevasta pelistä tallennetaan siihen liittyvät tiedot, jotka rajasin seuraaviin; julkaisuvuosi, pelin kehittäjät, pelin julkaisijat, kansikuva sekä lajityyppi. Alunperin suunnittelin antavani käyttäjille vapaat kädet, eli käyttäjä voisi

itse syöttää lisättävään peliin liittyvät tiedot. Tulin kuitenkin nopeasti siihen tulokseen, että yhden käyttäjän antamiin tietoihin ei voi välttämättä luottaa, ja tiedon korjaamiseen menisi ylläpitäjältä kohtuuttomasti aikaa. Parempi vaihtoehto olisikin peliä syötettäessä hakea tiedot jostain muualta. Tähän tarkoitukseen käytän mm. videopelien arvostelusivusto Giant Bombin wikiä, josta tiedon pystyy helposti hakemaan käyttämällä sivuston julkista ohjelmointirajapintaa (engl. Application Programming Interface, API). Lisäksi työssä käytetään tiedon hankkimiseksi Steamin julkista web- ohjelmointirajapintaa. Hintatietojen hankkimiseksi käytin verkkosivunharavointia (engl. web scraping), koska varsinaista ohjelmointirajapintaa ei ollut saatavilla.

2.4 Ajastetut toiminnot

Ajastetuille toiminnoille asetetut vaatimukset olivat yksinkertaiset. Sivuston käyttäjien tiedot ja omistetut pelit päivitetään Steamin ohjelmointirajapintaa käyttäen kerran tunnissa. Näistä koostetaan statistiikkaa, joista näkee esimerkiksi montako omistajaa kullakin tietokannassa olevalla pelillä on, tai mitä kukin sivuston käyttäjä omistaa.

Pelien hintatiedot tarkistetaan automaattisesti tällä hetkellä kahdelta sivustolta: g2a.com sekä kinguin.net. Koska kummallakaan ei ole julkista ohjelmointirajapintaa, käytän omia libcurl-kirjastoa käyttäviä haravointiskriptejä tiedon keräämiseen.

2.5 Käyttäjiltä ja ulkopuolisilta sivustoilta kerätty tieto

Tärkeää oli, että käyttäjiltä syötteitä hyväksyvät lomakkeet olisivat mahdollisimman turvallisia, eikä esimerkiksi SQL-injektio tai XSS-hyökkäys niitä käyttäen olisi mahdollista.

SQL-injektioiden välttämiseksi tietokantakyselyissä käytettiin hyväksi valmisteltuja lauseita (engl. prepared statements), joista kerron lisää luvussa 6.

XSS-hyökkäyksien estämiseksi kaikkeen tulostettavaan tekstiin käytetään PHP:n htmlentities- tai htmlspecialchars -funktiota, jolloin niihin mahdollisesti sijoitettua

haittakoodia ei suoriteta selaimessa. XSS-hyökkäyksistä ja niiden estämisestä kerron myös lisää luvussa 6.

3 SIVUSTON OMINAISUUDET

Tässä luvussa käsitellään tekemäni verkkosivuston ominaisuuksia.

3.1 Navigaatiopalkki

Sivuston navigaatiopalkin sisältävä ”header.php”-tiedosto sisällytetään jokaiseen sivulataukseen PHP:n ’require_once’ -käskyllä. Näin siihen tehtävät muutokset päivittyvät samantien jokaiselle sivulle, eikä koodia tarvitse turhaan toistaa.

Navigaatiopalkki generoidaan dynaamisesti käyttäjälle istuntotiedoissa olevien tietojen perusteella, jolloin esimerkiksi kirjautumispainike näkyy vain uloskirjautuneille käyttäjille, ja adminvalikko vain adminoikeudet omaaville käyttäjille.

3.2 Automaattinen tiedonhaku ja ajoitetut käskyt

Suurin osa sivustolla olevasta tiedosta on automaattisesti kerättyä. Tiedon keräämiseen käytetään palvelimella ajettavia ajoitettuja käskyjä eli cron jobeja.

Käyttäjien Steam-tietojen päivitys tehdään cron jobilla, joka hakee käyttäjän tiedot Steamin web-ohjelmointirajapintaa käyttäen. Jos Steamin palvelimella oleva tieto poikkeaa tietokannassa olevasta, muutokset lisätään tietokantaan. Lisättyjen ja poistettujen pelien tietokantamerkinnöissä on mukana aikaleima, jonka avulla tiedosta voidaan koostaa käyttäjälle vaikkapa aikajana pelikirjaston muutoksista.

Cron jobeja käytin myös aikaisemmin mainittujen hintatietojen keräämiseen, sekä sivustolla olevan statistiikan laskemiseen. Raskaimmat tehtävät tehdään taustalla, jotta sivulataukset olisivat nopeita eivätkä turhaan hidastuisi niiden takia.

3.3 API

Sivustolla on oma API, jonka avulla ohjelmistokehittäjät voivat pyytää tiettyjä tietoja sivuston tietokannasta JSON-muodossa sen sijaan, että käyttäisivät hitaita

haravointiskriptejä saman tiedon keräämiseen. Tietoja voi käyttää esimerkiksi selainlisäosissa, jotka näyttävät sivuston statistiikkaa suoraan käyttäjän Steam-profiilissa.

{"success":true,"steamid":"7656119796954xxxx","recent_activity":[{"appid":"612290","status":"new","name":"ASTA Online"},{"appid":"561970","status":"new","name":"REALITY"},{"appid":"607270","status":"new","name":"Jack's Gang"},{"appid":"598340","status":"new","name":"Lament"},{"appid":"585860","status":"new","name":"Bunker 58"},{"appid":"545490","status":"new","name":"Akuya"},{"appid":"522590","status":"new","name":"Near Midnight"},{"appid":"598440","status":"new","name":"Roots of Insanity"},{"appid":"446250","status":"new","name":"Silver Bullet: Prometheus"},{"appid":"388910","status":"new","name":"Sentinels"},{"appid":"556010","status":"new","name":"Neon Arena"},{"appid":"604190","status":"new","name":"Storm Riders"},{"appid":"497570","status":"new","name":"Building the Great Wall of China 2"},{"appid":"494290","status":"new","name":"WARZONE"},{"appid":"618360","status":"new","name":"Aesthetic Melody"}]}

Kuva 2. JSON-muotoinen vastaus API-pyyntöön, jossa kysyttiin viimeisimpiä muutoksia tietyn käyttäjän pelikirjastoon.

API on toteutettu siten, että validoidun GET-pyynnön perusteella suoritetaan tietty tietokantahaku, jonka tulos näytetään JSON-muodossa. Tulos pidetään palvelimen kiintolevyllä 20 minuuttia, jonka aikana kaikille samanlaisille pyynnöille näytetään sama tieto. Näin varmistetaan se, että tietyn pyynnön toistaminen ei rasita palvelinta turhilla tietokantahauilla.

```
API information
----

https://[redacted]com/api?action=GetUserStats&steamid=steamid
returns stats for user and some basic info

# return format:

* success: true / false
* steamid
* steamname
* appcount - count of apps in users library, with free titles
* removed_games_total - the total of counts for removed game categories (delisted, disabled, retail only, etc.)
* stats - array that contains each special category and their respective counts for the user
        it contains:
        * category_id -internal permanent id for the category
        * category - category title (delisted, purchase disabled, etc). Not permanent, titles may change without notice
        * count - user total for apps belonging in this category
        * category_total - total count of apps in this category


https://[redacted].com/api?action=GetUserRecentActivity&steamid=steamid
returns apps recently added / removed from users library, if not set private

# return format:

* success: true / false
* steamid
* recent_activity - array that contains recently activated / deleted games for the given user, newest first
        it contains:
        * appid
        * status - 'new' for apps added to users library, 'deleted' for apps deleted from it
        * name - app name
```

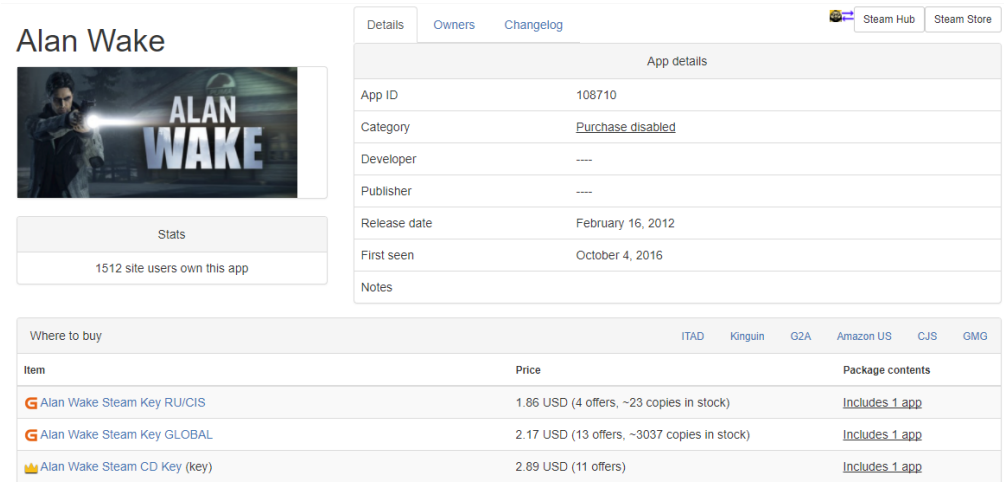
Kuva 3. API-metodit listataan ohjesivulla, jossa kerrotaan myös niiden hyväksymät parametrit sekä vastauksen rakenne.

3.4 Käyttäjän profiilisivu

Jokaiselle sivustolle rekisteröityneellä käyttäjällä on profiilisivu, johon on koostettu käyttäjään liittyvää tietoa. Tällä hetkellä sivulla näkyy esimerkiksi, kuinka monta peliä käyttäjä omistaa, ja montako niistä ei ole enää ostettavissa Steamista. Lisäksi sivulla näkyy lista uusimmista muutoksista käyttäjän pelikirjastossa, eli viimeisimmät poistetut ja lisätyt pelit. Käyttäjä voi halutessaan piilottaa tämän listan asettamalla sen yksityiseksi sivuston yksityisyysasetuksista.

3.5 Pelin profiilisivu

Jokaisella tietokannassa olevalla pelillä on oma profiilisivunsa, johon on koostettu peliin liittyvää tietoa. Automaattiseen tiedonhakuun käyttämäni skriptit ovat näiltä osin vielä jonkin verran kehitysvaiheessa, joten tällä hetkellä sivulla näytettyyn tietoon kuuluvat lista käyttäjistä, joilta kyseinen peli löytyy pelikirjastosta sekä näiden yhteismäärä, muutoslogi pelin kategoriamuutoksista, sekä automaattisesti koostettu hintavertailu kauppakohteista G2A sekä Kinguin -sivustoilla.

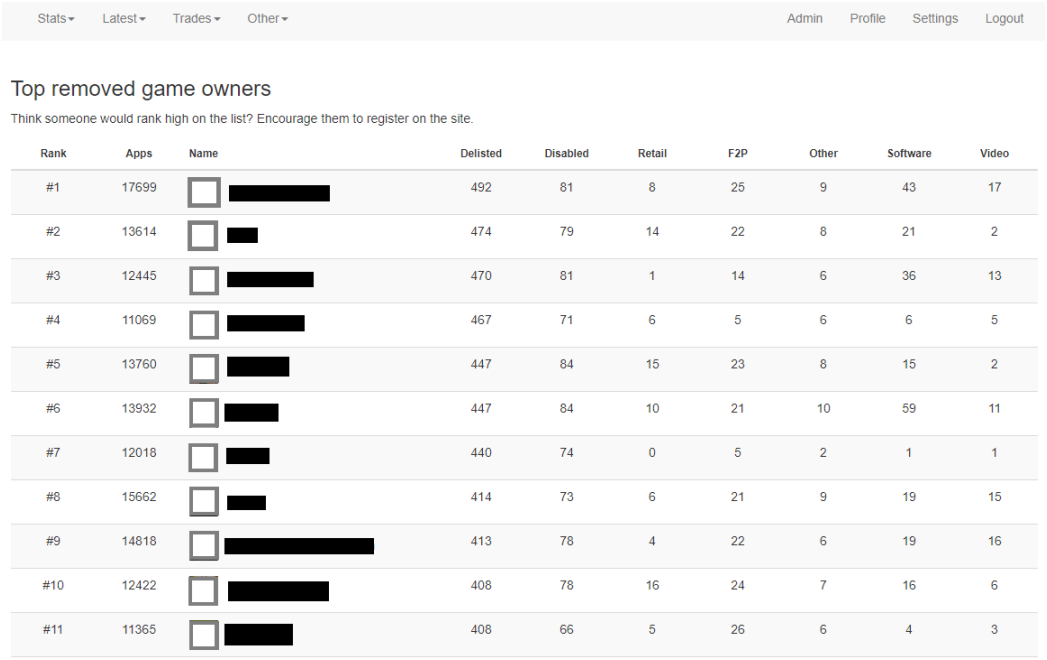


Kuva 4. Yksittäisen pelin profiilisivu.

3.6 Top-lista

Keräilijöillä on usein vähän näyttämisen halua, ja siitä syystä yksi ensimmäisistä sivustolle lisätyistä ominaisuuksista olikin automaattisesti koostettu top-lista. Lista päivittyy cron jobin avulla useita kertoja päivässä, ja sisältää harvinaisempien

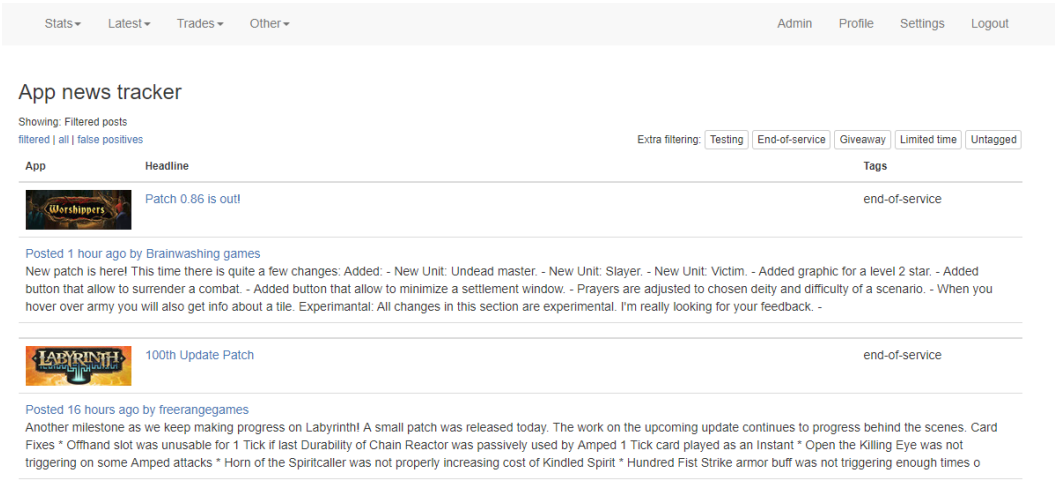
pelien määrän perusteella lajitellun listan sivuston käyttäjistä. Kuten muissakin taulunäkymää käyttävissä sivuissa, käyttäjä voi vapaasti lajitella taulun haluamaansa kentän perusteella klikkaamalla ylätunnistetta.



Kuva 5. Top-listan sisältävä taulu

3.7 Steam-uutisten suodattaminen

Joskus pelijulkaisijat julkaisevat Steamissa uutisen, jossa ilmoittavat tietyn tuotteen poistamisesta Steamista etukäteen. Tästä syystä lisäsin sivun, johon kerään cron jobilla uutiset, joissa esiintyy tiettyjä tuotteen poistamiseen liittyviä sanoja. Sataprosenttista tarkkuutta ei tällä metodilla saavuteta, mutta sadoista päivittäisistä uutisista seulaan tarttuu vain pieni osa, josta merkitykselliset uutiset löytyvät nopeammin.



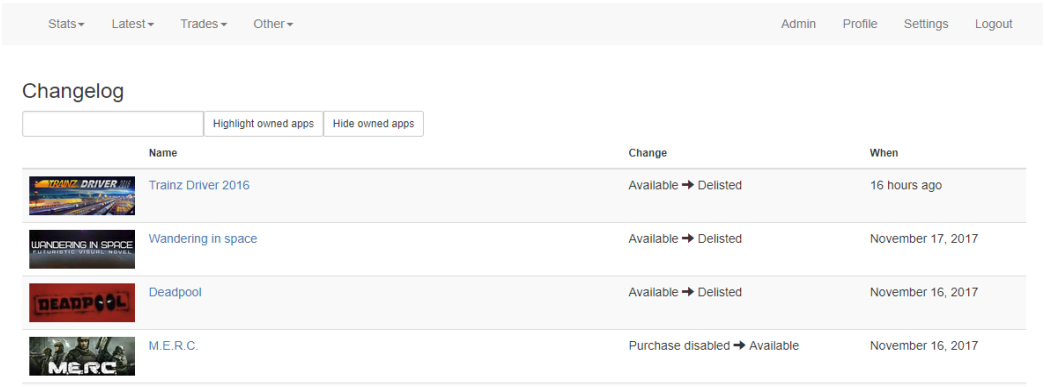
Kuva 6. Uutislistaus.

Uutiset keräävä cron job toimii siten, että se haravoi tietyin väliajoin uusimpien uutisten ID-numerot Steamin sivuilta, ja kerää sen jälkeen uutisten sisällön Steamin ohjelmointirajapinnasta. Tämän jälkeen palvelimella oleva skripti hakee uutisten sisällöstä ja otsikoista tiettyjä avainsanoja tai -lauseita ja merkitsee löydetyt näytettäväksi uutissivulla.

3.8 Muutoshistoria

Muutoshistorian 200 viimeisintä tietuetta näytetään Changelog-sivulla. Kun jonkun pelin kategoria muuttuu sivuston tietokannassa, muutos kirjataan samalla muutoshistorian sisältävään tauluun. Tietueeseen kuuluu pelin ID-numero, muutoksen aikaleima, vanha kategoria sekä uusi kategoria.

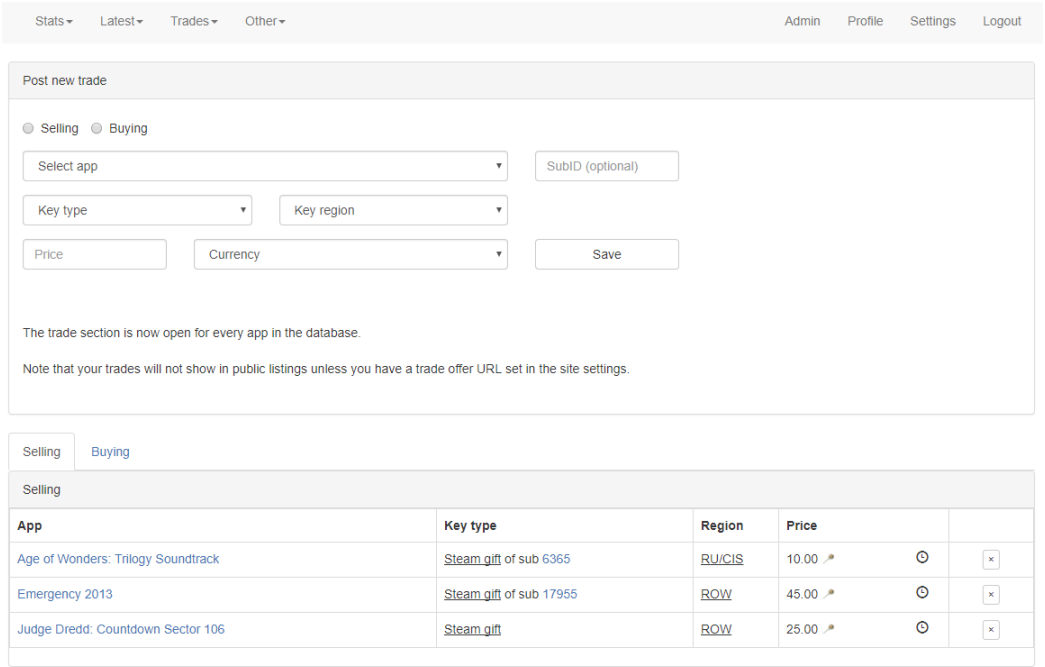
Käyttäjä voi korostaa listasta omistamansa pelit nappia painamalla samaan tapaan kuin sivuston pääsivulla. Aikaleima näytetään päivämääränä, jos muutoksesta on kulunut yli 24 tuntia, muussa tapauksessa näytetään montako tuntia sitten muutos on tehty.



Kuva 7. Lista viimeisimmistä kategoriamuutoksista.

3.9 Vaihtokauppalistaukset

Pelien vaihtaminen ja myyminen on monille olennainen osa Steam-pelien keräilyä, joten vaihtokauppalistaus oli luonnollista lisätä sivustolle. Nykyiset vaihtoihin erikoistuvat sivustot ovat lähinnä keskustelupaikkoja, joille ihmiset listaavat myytävänä olevat tuotteensa, yleensä ilman selkeästi merkittyjä hintoja.



Kuva 8. Vaihtokauppalistauksen lisäysnäkymä.

Pyrin omassa vaihtokauppalistauksessani eliminoimaan ostajalle hankalat hinnoittelemattomat pelit tekemällä hinnan antamisesta pakollista. Toinen ostajille

mieluinen ominaisuus on sivu, joka suodattaa listasta jo omistetut pelit automaattisesti. Tiedonsyötön kannalta järjestelmän pitäisi olla tietoturvallinen, ja kaiken tiedon oikeellisuus tarkistetaan syötön yhteydessä. Ajaxilla lähetettävissä vaihtojen poisto- sekä lisäyslomakkeissa on mukana vaihtuva tarkistusmerkkijono (token) joka estää CSRF-hyökkäykset.

Koska vaihtokauppaosio oli suunnitelusta toteutukseen vain noin päivän työ, siinä on suunnitteluvirheitä, joiden takia käyttettävyys on jonkin verran puutteellista. Ainakin pelilistausten lisäämisessä ja hakuominaisuuksissa olisi vielä parantamisen varaa, ja siksi aionkin vielä joskus suunnitella koko järjestelmän uudelleen.

3.10 Muutosloki

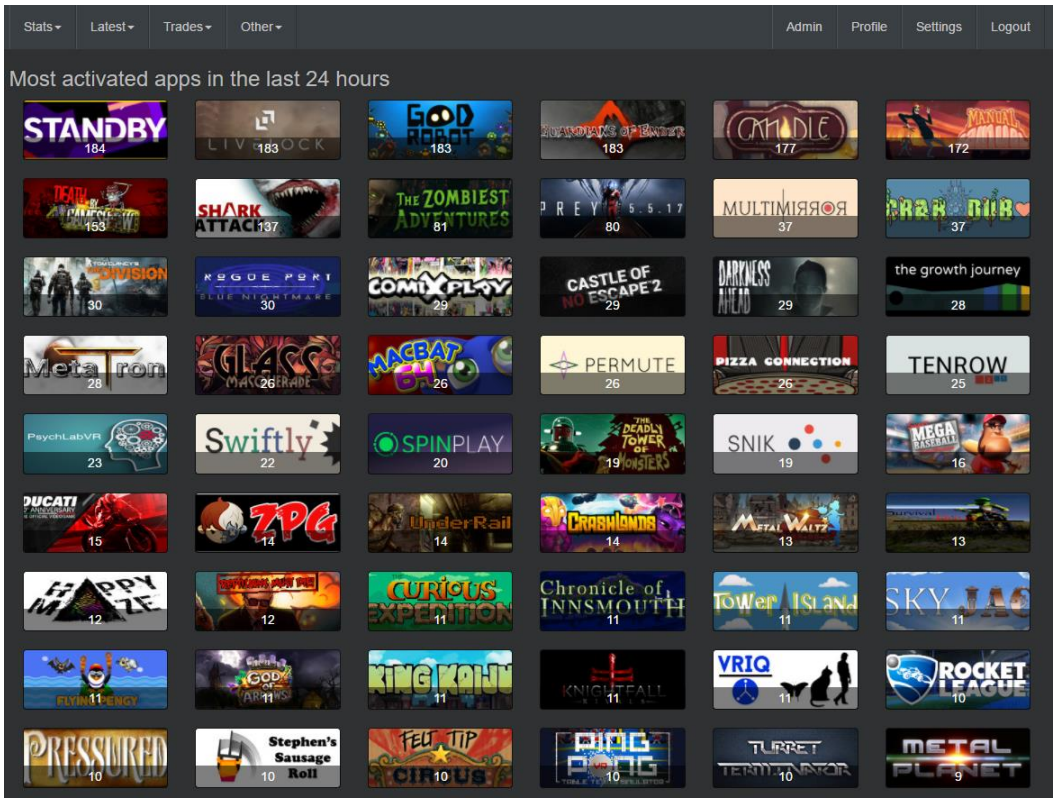
Käyttäjien pelikirjaston muutoksia näytetään kolmella eri sivulla.

Yhdellä sivulla näytetään käyttäjätileiltä lähiaikoina poistetut pelit, jolloin voi helposti nähdä, onko tietty peli poistettu jostain syystä samanaikaisesti useilta eri käyttäjältä. Syitä pelien poistamiseen ovat esimerkiksi beta-testauksen loppuminen tai laajat huijaukset, joissa käyttäjät ostavat tietämättään varastettuja pelejä jotka poistetaan tileiltä myöhemmin.

Yhden sivun tarkoitus on näyttää viimeisimmät aktivoinnit peleille, jotka eivät enää ole myynnissä. Näin käyttäjät näkevät nopeasti, jos tietty haluttu peli on hetkellisesti saatavilla jostain.

Yksityisyyttään vaalivat käyttäjät voivat poistaa itsensä näiltä sivuilta muokkaamalla yksityisyysasetuksiaam. Yksityiseksi merkityn käyttäjän aktivoinnit näkyvät sivulla nimettömänä.

Kolmas käyttäjätietoa hyödyntävä sivu näyttää eniten hankitut pelit viimeisen 24 tunnin ajalta.



Kuva 9. Viimeisen 24 tunnin aikana eniten hankittujen pelien lista.

4 KÄYTETYT TEKNOLOGIAT JA TYÖKALUT

Järjestelmän luomiseen käytettiin useita eri teknologioita. Käyttöliittymän tekemiseen käytettyihin teknologioihin lukeutuvat HTML, CSS, JavaScript sekä jQuery. Ulkoasun luomiseen käytettiin lähinnä Bootstrapin valmiita elementtejä. Järjestelmän ominaisuudet toteutettiin käyttämällä PHP-ohjelmointikieltä sekä MySQL-relaatiotietokantaa. Tässä luvussa esittelen työn toteutuksessa käytössä olevia teknologioita sekä apuna käytettyjä ohjelmistoja.

Ohjelmointityöhön ei tarvitse erityisiä työkaluja, vaan sen voi periaatteessa tehdä alusta loppuun vaikka pelkästään Notepad-tekstieditoria käyttäen. Työn saa kuitenkin sujumaan paremmin käyttämällä apuna erilaisia ohjelmistoja. Näistä tärkein oli ehdottomasti WampServer, jonka avulla koodin testaus onnistui jo kotikoneella.

4.1 HTML

HTML on merkintäkieli, jota käytetään verkkosivujen näyttämiseen. HTML-dokumenteilla on standardisoitu rakenne, joka takaa niiden lähes samanlaisen toimivuuden eri selaimilla. HTML-elementtejä ovat esimerkiksi lomakkeet (form), otsikot (header) sekä div, jolla sivu voidaan jakaa eri osiin. (Suehring & Valade 2014, 89-95)

HTML5 on uusi versio HTML-standardista, joka lisäsi kieleen nykyaikaisia käytettävyyttä parantavia ominaisuuksia, kuten lomaketietojen tietotyypit ja niiden validoinnin sekä video- ja äänitiedostojen upottamisen (engl. embed) ja soittamisen. Aikaisemmin videota ei voinut esittää selaimessa käyttämättä jotain selain-pluginia, kuten esimerkiksi Adobe Flashia. (W3schools, n.d.)

4.2 CSS

CSS-tyylitiedostoja (engl. Cascading Style Sheets) käytetään HTML-sivujen ulkoasun muokkaamiseen. Ennen sivun jokaiselle elementille jouduttiin tekemään tyyli manuaalisesti, CSS mahdollistaa saman tai useamman tyylin asettamisen useampaan elementtiin samanaikaisesti. (Suehring & Valade 2014, 121-122)

CSS3 on standardin uusin versio, jossa uusina ominaisuuksina ovat esim. animaatiot, 2D/3D-efektit, sekä reunukset ja varjostukset. (W3schools, n.d.)

4.3 PHP

PHP (alunperin *Personal Home Page*, nykyään *PHP: Hypertext Preprocessor*) on Rasmus Lerdorfin luoma, erityisesti dynaamisten ja interaktiivisten verkkosovellusten tekemiseen suunniteltu palvelinpuolen komentosarjakieli (engl. scripting language).

Kun palvelimelta pyydetään PHP:ta sisältävä sivu, se lähettää PHP-tulkille (engl. PHP interpreter) käskyn esiprosessoida sivu. PHP-tulkki suorittaa PHP-tiedoston sisältämän koodin ja palauttaa palvelimelle ainoastaan HTML-koodista koostuvan valmiin sivun, joka puolestaan välitetään käyttäjälle. (Suehring & Valade 2013, 272)

PHP valittiin työssä käytetyksi ohjelmointikieleksi lähinnä siksi, että suosionsa ansiosta sille löytyy paljon esimerkkejä ja dokumentaatiota, joiden avulla oli helppo päästä alkuun. Lisäksi oli yksinkertaista ja nopeaa asentaa käyttövalmis testipalvelin kotikoneelle käyttäen WAMP-ohjelmistopakettia, johon kuuluu mm. Apache, PHP sekä MySQL-palvelin.

4.4 MySQL

MySQL on Oracle Corporationin vuodesta 2009 omistama relaatiotietokantajärjestelmä, jota kehittää ruotsalainen MySQL AB. Relaatiotietokannassa tieto sijoitetaan useisiin eri tauluihin, jotka koostuvat riveistä (engl. row) ja sarakkeista (engl. column). Näillä tauluilla voi olla relaatiomallin mukaisesti viittauksia toisiin tauluihin.

Viittauksia käytetään vähentämään tallennetun tiedon määrää. Tavoitteena on, että jokainen asia tallennettaisiin tietokantaan ainoastaan kerran.

Esimerkiksi kaupan tilauksia sisältävään tauluun ei tarvitse lisätä uudelleen jokaisen asiakkaan tietoja, vaan asiakas voidaan liittää tilaukseen viittaamalla

asiakkaan ID-tunnuksella asiakastiedot sisältävään tauluun. (Suehring & Valade 2013, 450)

4.5 JavaScript

JavaScript on vuonna 1995 julkaistu komentosarjakieli, joka kuuluu HTML:n ja CSS:n ohella webin ydinteknologioihin. Kaikissa moderneissa verkkoselaimissa on JavaScript-tuki.

JavaScriptiä käytetään yleisesti tekemään verkkosivuista dynaamisempia. Sitä voidaan käyttää esimerkiksi sivun muokkaamiseen ilman että sitä ladataan kokonaan uudelleen.

4.6 jQuery

jQuery on John Resigin alunperin vuonna 2006 julkaisema JavaScript-kirjasto, joka on suunniteltu yksinkertaistamaan JavaScriptin käyttöä. jQuery on erittäin suosittu JavaScript-kirjasto, ja sitä käyttää jopa yli 70 prosenttia maailman verkkosivuista. (W3Techs, 2017)

jQuery sisältää CSS-valintamoottori (engl. CSS selector engine) Sizzlen, jolla voi helposti etsiä ja valita sivulta haluamiaan DOM-elementtejä. Lisäksi siihen kuuluu monia erilaisia metodeita eri käyttötarkoituksiin kuten JSON-jäsennys (engl. parsing), Ajax sekä tapahtumiin reagoivia metodeja (engl. event listener).

4.7 Bootstrap

Bootstrap on alunperin Twitterille luotu, ja vuonna 2011 julkaistu avoimen lähdekoodin front-end ohjelmistokehys. Se sisältää HTML, CSS ja JavaScript – pohjaisia komponentteja, joiden tarkoitus on nopeuttaa verkkosovelluksen käyttöliittymän kehitystä. Bootstrap on erityisen hyödyllinen, kun halutaan tehdä responsiivisia verkkosivuja. (Otto 2011)

Bootstrapin käyttöönotto on yksinkertaista, sivulle täytyy vain lisätä sen bootstrap.css sekä bootstrap.js -tiedostot. Tämän jälkeen voi omassa html-koodissa käyttää Bootstrapin verkkosivuilla olevasta dokumentaatiosta löytyviä valmiita

elementtejä. Verkkosivuilta löytyy myös kustomointityökalu, jolla elementtien värit voi muokata mieleisekseen.

4.8 WampServer

WampServer on kokoelma Windows-ympäristössä toimivia ohjelmistoja, jotka muodostavat dynaamisia verkkosivuja paikallisesti ajavan WWW-palvelimen. Se sisältää Apachen, PHP:n sekä MySQL:n. Ohjelmiston avulla voi kotikoneesta tehdä nopeasti kehitysympäristön, jossa omaa koodia pääsee testaamaan ennen oikealle palvelimelle siirtämistä.

Vastaavia kokoelmia ovat esimerkiksi XAMPP (Cross-Platform, Apache, MariaDB, PHP & Perl), WIMP (Windows, IIS, MySQL, PHP) ja LAMP (Linux, Apache, MySQL & PHP). Kokoelmista on myös erilaisia versioita, jotka voivat sisältää esimerkiksi PHP:n sijaan tai lisäksi Perlin tai Pythonin.

4.9 MySQL Workbench

MySQL Workbench on Oracle Corporationin kehittämä MySQL-tietokantojen suunniteluun ja hallintaan tarkoitettu ohjelmisto. Se sisältää monia hyviä ominaisuuksia MySQL-tietokannan hallintaan, kuten esimerkiksi graafisen MySQL-editorin, jonka avulla tietokannan voi luoda viiteavaimineen kirjoittamatta yhtäkään SQL-komentoa. Samalla editorilla voi myös muokata palvelimella olevaa tietokantaa esimerkiksi SSH-yhteyden kautta. Ohjelmistolla suunnitellun tietokannan voi tallentaa vaihtoehtoisesti SQL-komentosarjana, jonka ajamalla MySQL luo tietokantaan kuuluvat taulut. (Oracle Corporation, n.d.)

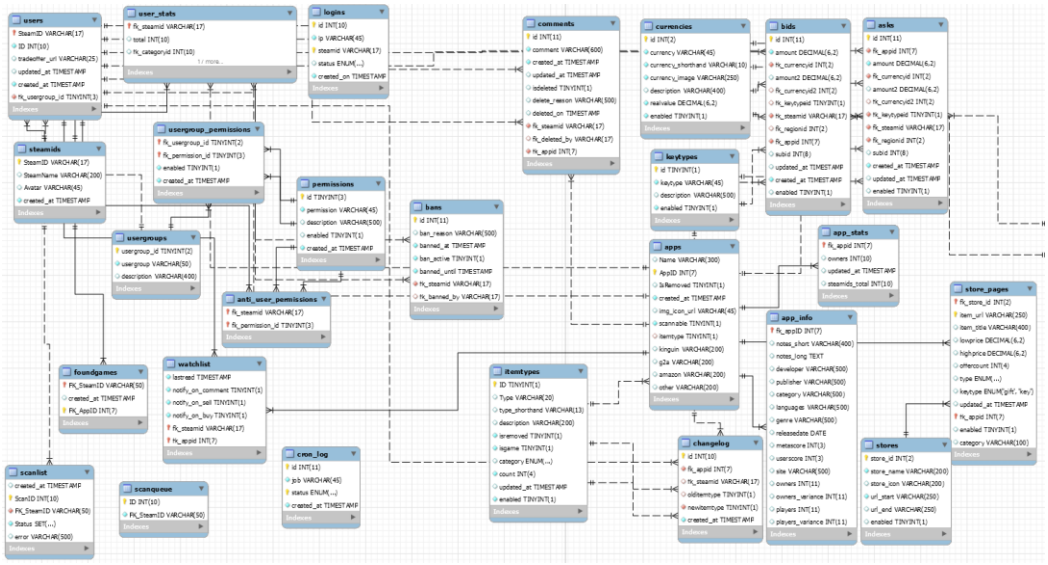
Tässä työssä olevat tietokannan rakennetta esittelevät kuvat on myös otettu MySQL Workbenchin avulla.

5 TOTEUTUS

Tässä luvussa käsitellään asioita, jotka liittyvät projektityön toteutukseen.

5.1 Tietokanta

Kun olin kartoittanut sivuston tarvitseman tiedon käyttämällä apuna käyttötapauksia ja listaa vaadittavista ominaisuuksista, aloin suunnittelemaan järjestelmän käyttämää MySQL-relaatiotietokantaa. Tietokannan rakenteen (engl. database schema) suunnittelussa ei kannata hätiköidä, koska se voi olla yksi järjestelmän kriittisimmistä osista. Huonosti suunniteltu tietokanta heikentää järjestelmän käytettävyyttä sekä tekee tietokannan muuttamisesta ja uusien tietojen lisäämisestä ja muokkaamisesta todella työlästä.



Kuva 10. Järjestelmän tietokanta.

Hyvän tietokantarakenteen saamiseksi on kehitetty vaiheittainen malli, jota kutsutaan tietokannan normalisoinniksi (engl. database normalization). Sillä pyritään tukemaan tiedon eheyttä, ja minimoidaan turhan datan määrä. Normalisoidussa tietokannassa tieto on jaettu useampiin tauluihin, mikä osaltaan vähentää tiedon eheyden vaarantamisen riskiä. (Microsoft, 2013)

Ensimmäisessä normaalimuodon (1NF) ehdot täyttävässä taulussa mikään yksittäinen taulu ei saa sisältää toistuvia ryhmiä toisiinsa liittyvää tietoa, vaan ne

tulisi jokainen siirtää omaan tauluunsa. Missään kentässä ei saa myöskään olla useampia arvoja, kuten numerolistaa. (Microsoft, 2013)

Näiden ehtojen täyttämiseksi en esimerkiksi tallentanut yksittäisten pelien julkaisijalistaa samaan tauluun, jossa on yleistiedot pelistä, vaan julkaisijatiedolle on luotu oma taulunsa. Erillisen taulun käyttö mahdollistaa useamman julkaisijan asettamisen samalle pelille lisäämättä niitä kaikkia yhteen kenttään. Tiedon hakeminen yksittäisiin kenttiin sijoitetuista listoista olisi hankalaa ja tehotonta.

Käytännössä siis tieto on jaettu useisiin tauluihin, ja viiteavaimia käytetään runsaasti. Tämä mahdollistaa sen, että tietojen muokkaus tai poisto myöhemmin ei vaikuta negatiivisesti tietokannan eheyteen tai pakota muokkaamaan useita tietueita samanaikaisesti.

Toisen normaalimuodon (2NF) ehtojen täyttämiseksi missään taulussa ei saa olla arvoja, jotka voivat liittyä useampaan tietueeseen. Jokaisella taululla tulee myös olla perusavain. Kolmannen normaalimuodon (3NF) ehtojen mukaan missään taulussa ei saa olla tietoa, joka ei liity taulun perusavaimeen.

Tietokantaa ei kannata aina normalisoida kokonaan, koska useat pienet taulut voivat vaikuttaa negatiivisesti tietokannan suorityskykyyn ja tehdä hakujen kirjoittamisesta työläämpää. (Ullman 2012, 172-176) Oma tietokantani rikkoo toisen ja kolmannen normaalimuodon ehtoja mm. aikaleima (engl. timestamp) sarakkeen vuoksi. Näille en tehnyt omaa taulua tietokantahakujen yksinkertaistamiseksi, ja koska niissä oleva tieto ei ole tietokannan eheyden säilyttämisen kannalta relevanttia.

5.2 Tietokannan rakenne

Tässä osiossa selvitän järjestelmää varten luodun tietokannan rakennetta ja kerron tarvittaessa miksi ratkaisuun on päädytty. Suurimmasta osasta tauluja löytyy updated_at ja created_at -kentät, joihin tallennetaan nimensä mukaisesti aikaleima rivin viimeisimmästä päivityksestä tai luomisesta. Enabled-kenttä toistuu myös monessa taulussa, sitä käytetään tiettyjen rivin ”poistamiseen” tietokantahauista.

5.2.1 Käyttäjätiedot

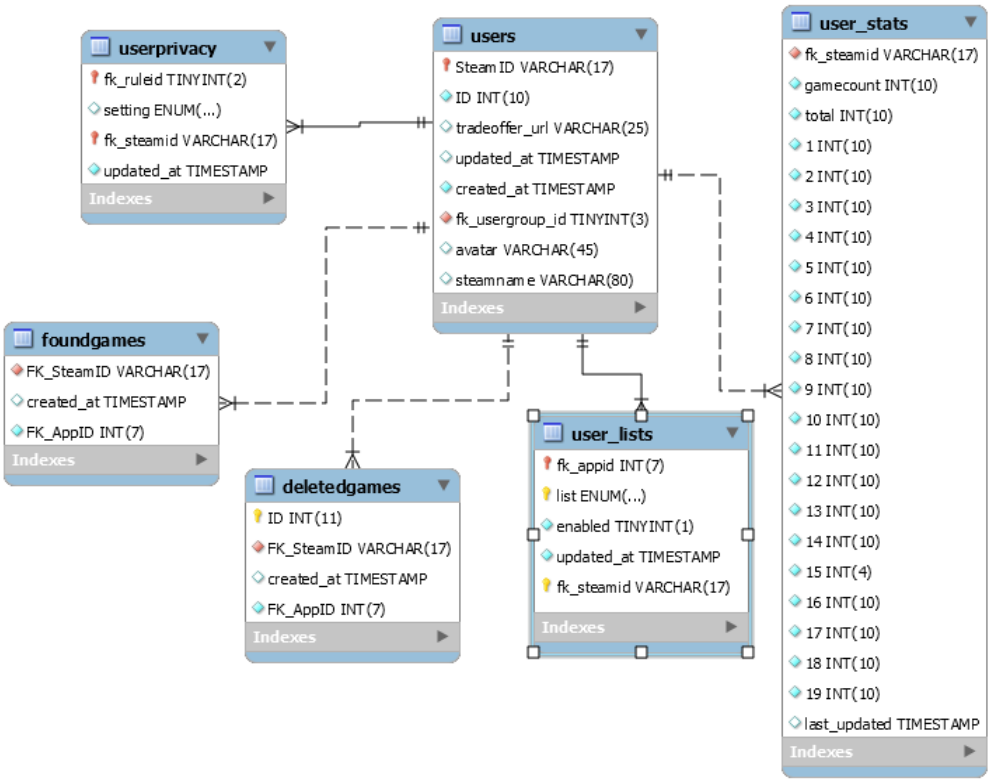
Yksittäisiin käyttäjätileihin liittyvä tieto on kuvassa 12 näkyvissä kuudessa taulussa. Näistä *users* sisältää perustietoa käyttäjästä, kuten Steam-tunnuksen ID-numeron (SteamID), URL-osoitteen vaihtotarjousten lähettämistä varten (tradeoffer_url), käyttäjäikonin (avatar) osoitteen Steamin palvelimella, sekä Steam-tunnuksen nimen (steamname). Lisäksi tähän tauluun merkitään mihin käyttäjäryhmään käyttäjä kuuluu (usergroup id). Muiden taulujen viitatessa tiettyyn käyttäjään käytetään viiteavaimena SteamID-kenttää.

Userprivacy-taulusta löytyvät kaikkien käyttäjien yksityisyysasetukset. Asetuksia on tällä hetkellä viisi erilaista, ja ne hallinnoivat käyttäjätiedon julkista näkyvyyttä järjestelmän eri osissa. Jos käytettäisiin yksittäistä globaalia yksityisyyssääntöä, voisi tämän tiedon sijoittaa vaikka *users*-tauluun muun tiedon ohella.

Jokainen sääntö tallennetaan eri riville, ja yksi rivi koostuu säännön ID-numerosta (ruleid), käyttäjän Steam-tunnuksesta (SteamID), sekä valitusta asetuksesta (setting), joka voi olla joko ”public” tai ”private”. Pääavaimena taulussa on Steam-tunnuksen ja säännön ID-numeron yhdistelmä. Useasta kentästä koostuva pääavain estää kätevästi saman tiedon moninkertaisen lisäämisen, ja päällekkäisen tiedon löytyessä vanha rivi päivitetään uudella tiedolla.

Foundgames-taulu sisältää käyttäjän omistamien pelien AppID-numerot. AppID on Steamin järjestelmässä tuotteelle asetettu juokseva numero, joka toimii käytännössä pääavaimena jota voi käyttää esimerkiksi tietyn pelin tietojen hakemiseen. Jokaista käyttäjiltä löytyvää peliä varten tietokantaan lisätään uusi pelin numeron sekä käyttäjätunnuksen sisältävä rivi, koska yhdellä käyttäjällä voi olla käyttäjätilillään jopa tuhansia pelejä. *Deletedgames* on edellistä vastaava taulu, mutta siihen kirjataan nimensä mukaisesti pelit, jotka on käyttäjän pelilistasta jostain syystä poistettu. Nämä kaksi taulua voisi periaatteessa myös yhdistää, mutta *Foundgames*-taulun suuren koon (yli viisi miljoonaa riviä 1600 käyttäjää kohden) takia päädyin käyttämään erillistä taulua.

User_stats-tauluun tallennetaan cron jobilla lukumääriin perustuvaa käyttäjäkohtaista statistiikkaa, jotta pelejä ei tarvitsisi laskea suurista tauluista uudelleen jokaiselle sivulataukselle. Numeroidut kentät vastaavat eri pelikategorioita, total-kenttä niiden summaa. Gamecount-kentästä taasen löytyy käyttäjän pelien kokonaismäärä. En ole kovin tyytyväinen tämän taulun rakenteeseen, koska mahdollisille uusille kategorioille joutuu manuaalisesti lisäämään tauluun uuden kentän. Tietokantahakujen yksinkertaistamisen vuoksi päädyin kuitenkin pitämään tämän ratkaisun.



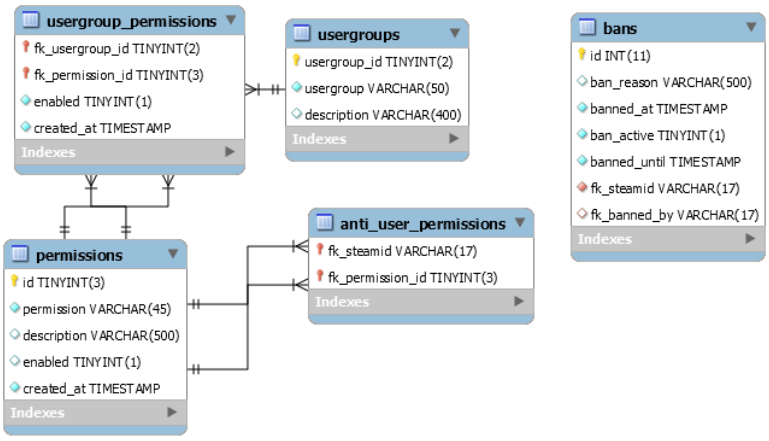
Kuva 11. Käyttäjätileihin liittyvät taulut.

User_lists-taulua käytetään tallentamaan erilaisia käyttäjäkohtaisia listoja. Tällä hetkellä tauluun tallennetaan esimerkiksi käyttäjän toivelistalle ja mustalle listalle asettamat pelit. List-nimiseen ENUM-kenttään sijoitetaan listan tyyppi, joka mahdollistaa usean listan tallentamisen samaan tauluun. Taulun pääavain koostuu käyttäjätunnuksen, pelin ID-tunnuksen sekä list-kentän arvon yhdistelmästä. Enabled-kenttää käytetään tiettyjen rivien ”poistamiseen” tekemättä varsinaista DELETE-käskyä.

5.2.2 Käyttäjäryhmien hallinnointi

Käyttäjäryhmien oikeuksia hallinnoidaan neljällä taululla. Jokainen käyttäjä kuuluu yhteen käyttäjäryhmään, joka määrittelee käyttäjän oikeudet sivustolla. *Permissions*-taulu sisältää mahdolliset käyttöoikeudet ja niiden kuvaukset, *usergroups*-taulu taas käyttäjäryhmien id-tunnukset ja nimen. Nämä kaksi taulua liittyvät *usergroup_permissions*-tauluun, joka sisältää listan jokaisen käyttäjäryhmän oikeuksista.

Jos tietyltä käyttäjältä halutaan evätä tietty oikeus, lisätään siitä merkintä *anti_user_permissions*-tauluun. Jos käyttäjälle annetaan täydellinen porttikielto, hänen tunnuksensa siirretään käyttäjäryhmään ”banned”, jolla ei ole mitään oikeuksia. Porttikiellosta lisätään tieto *bans*-tauluun, jossa olevan tiedon perusteella porttikielto voidaan poistaa tietyn ajan kuluttua automaattisesti.

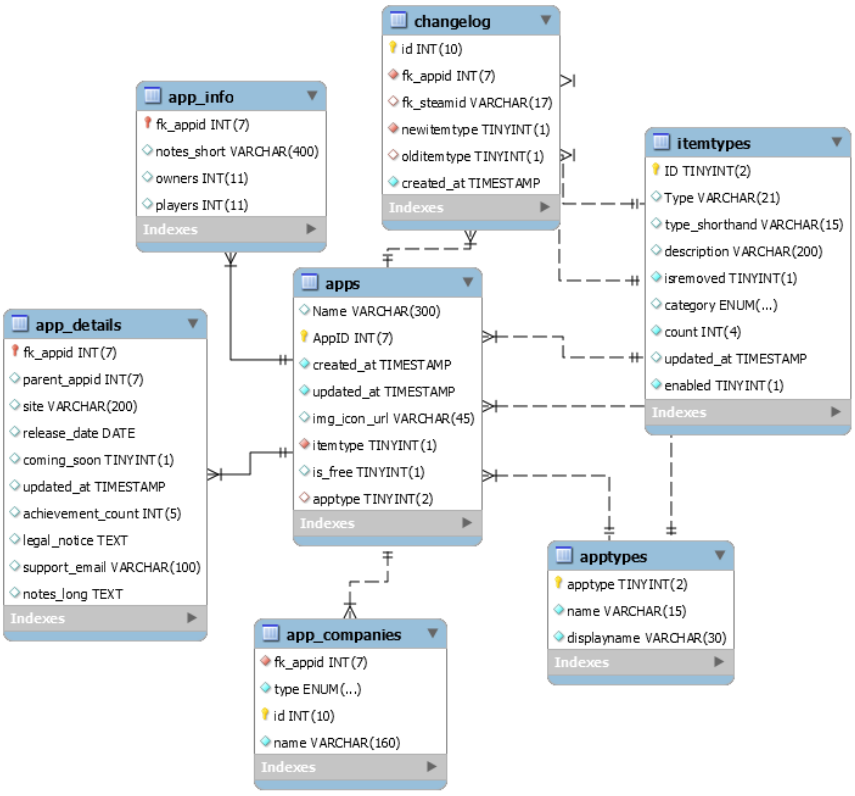


Kuva 12. Käyttäjäryhmiin liittyvät taulut.

5.2.3 Pelien ja muiden tuotteiden tiedot

Peleihin liittyvää tietoa säilytetään kuudessa eri taulussa. *Apps*-taulu sisältää perustiedot pelistä, ja sen pääavaimena olevaa AppID-kenttää käytetään viiteavaimena muissa tauluissa. Img_icon_url -kentässä pidetään pelin kuvakkeen osoitetta Steamin palvelimella, is_free-kenttä taas kertoo, onko peli ilmainen vai ei. Itemtype-kenttä viittaa *Itemtypes*-tauluun, joka sisältää sivustolla peleille asetetut kategoriat. Apptype-kenttä kuvaa tietyn tuotteen tuotetyyppiä (peli,

hyötyohjelmisto tai video). Se viittaa *Apptypes*-tauluun joka sisältää tällä hetkellä ainoastaan tuotetyypeille asetetut nimet.



Kuva 13. Peleihin liittyvät taulut.

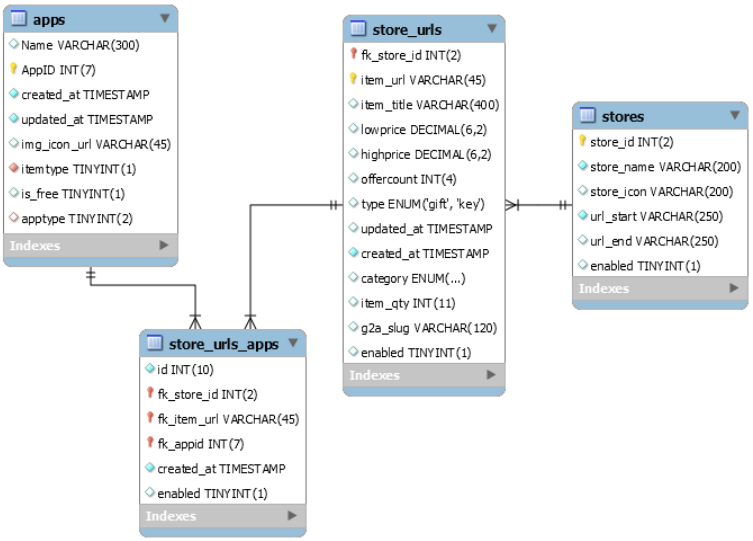
Itemtypes-taulu sisältää sivuston omat peleille asetettavat kategoriat. Kategorioita on useita, ja esimerkiksi Steamista poistetut pelit, elokuvat sekä ohjelmistot on jaettu kolmeen eri kategoriaan. Lisäksi oma kategoria löytyy esimerkiksi vain tietyissä maissa myytäville, beta-, ja ainoastaan ennakkotilausbonuksena tarjotuille peleille.

Description-kenttä sisältää kategorian kuvauksen, Type ja type_shorthand -kentät taas kategorian nimen sekä pienessä tilassa käytettävän lyhyemmän version nimestä. Isremoved-kenttä kertoo ovatko kategoriassa olevat tuotteet vielä myynnissä, vai poistettuja. Category-kenttään määritellään, ovatko tuotteet pelejä, videoita vai hyötyohjelmistoja. Count-kentässä on cron jobilla ajoittain laskettu tuotteiden lukumäärä kategoriassa. *App_info* ja *app_details* -taulut sisältävät tarkempaa tietoa tuotteista. Owners-kentästä löytyy tietyn tuotteen omistavien

käyttäjien määrä, ja players-kentästä moniko näistä on tuotetta käyttänyt. Parent_appid-kenttä sisältää tuotteen isäntätuotteen, jos tuote on lisäsisältöä (engl. DLC) jollekin toiselle tuotteelle. Loput *app_details* -taulun kentistä sisältävät lisätietoa joka on haettu tuotteen kauppasivulta, esimerkiksi julkaisupäivän ja verkkosivun osoitteen.

Changelog-taulussa pidetään lokia pelien kategoriamuutoksista. Muutoksista kirjataan aikaleima sekä uusi ja edelinen kategoria.

App_companies -taulussa pidetään listaa pelin julkaisija- ja kehittäjätiedoista. Type on enum-tyyppinen kenttä joka hyväksyy vain arvot ”dev” (developer) ja ”pub” (publisher). Näin molemmat listat voidaan pitää samassa taulussa ilman että tiedon haettavuus kärsii.



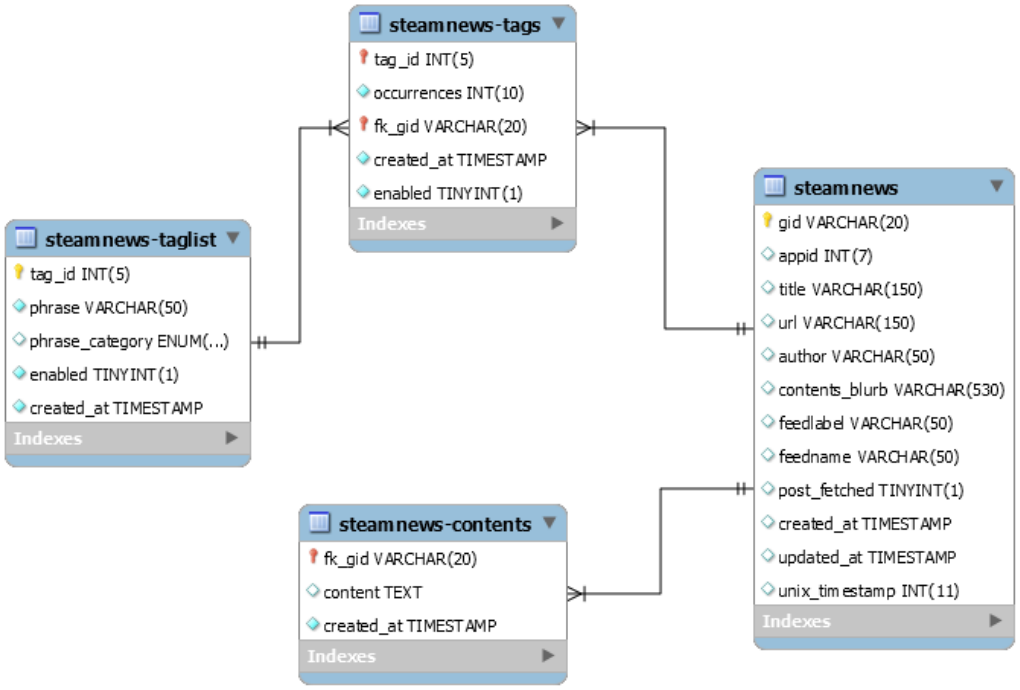
Kuva 14. Pelien hintatiedot sisältävät taulut.

Pelien hintatiedot sisältävät taulut viittaavat myös edellä mainittuun *Apps*-tauluun. *Stores*-taulu sisältää tiettyyn kauppaan liittyvää tietoa, kuten kaupan kuvakkeen, nimen sekä URL-osoitteen.

Store_urls-taulu sisältää yksittäisiä tuotetunnuksia ja niiden tarkat tiedot hinnasta myynnissä olevaan kappalemäärään. *Store_urls_apps*-taulu viittaa *store_urls*-tauluun ja listaa tietyn tuotetunnuksen alla myytävät pelit. Listaus on tarpeellinen, koska yksi tuotetunnus voi olla useammasta pelistä koostuva paketti.

5.2.4 Tuoteuutisia sisältävät taulut

Steamnews-taglist-taulussa pidetään listaa sanoista ja lauseista, joita etsitään uutisista. Sinne lisätään pääosin lauseita, jotka esiintyvät usein uutisissa, joissa ilmoitetaan tietyn tuotteen tuen lopettamista (engl. sunsetting).



Kuva 15. Pelien uutisia varten luodut taulut.

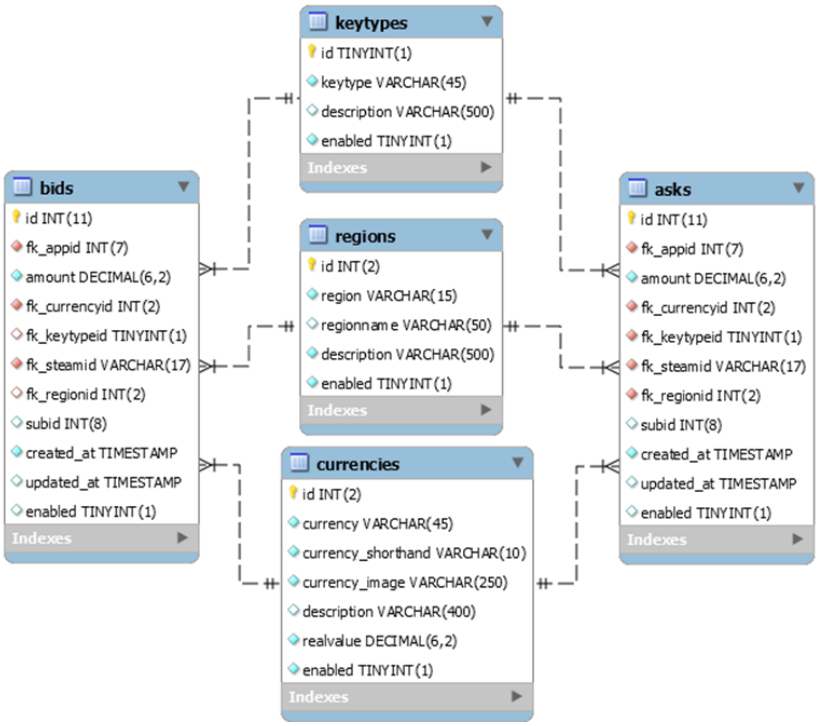
Steamnews-tags-tauluun tallennetaan sellaisten uutisten tunnukset, joista *steamnews-taglist*-taulussa olevia lauseita on löydetty.

Steamnews-taulu sisältää yksittäisten uutisten tiedot, kuten esimerkiksi minkä pelin uutinen on kyseessä, uutisen otsikon, pienen pätkän uutisesta sekä uutisen kirjoittajan nimen. Pääavaimena käytetään Steamin uutiselle antamaa gid-arvoa.

Steamnews-contents-taulu sisältää yksittäisten uutisten kokonaiset tekstit. Tekstit on sijoitettu omaan tauluunsa, jotta muiden taulujen käyttö ei hidastuisi niiden mahdollisesti suuren koon takia.

5.2.5 Vaihtokauppalistauksiin liittyvät taulut

Vaihtokauppalistauksiin liittyvä tieto on viidessä eri taulussa.



Kuva 16. Vaihtokauppalistauksia varten luodut taulut.

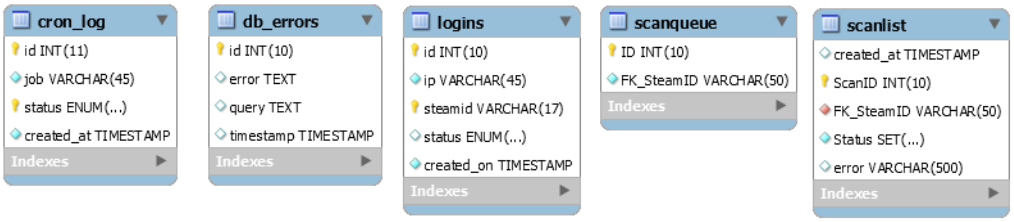
Käyttäjien tekemät vaihtokauppalistaukset tallennetaan tauluihin *asks* ja *bids*. *Asks* sisältää myynti-ilmoitukset, *bids* osto-ilmoitukset. Taulut ovat sisältämänsä tiedon puolesta samat. Taulut eroavat toisistaan siten, että osto-ilmoituksissa useampi kenttä on valinnainen täyttää, ja uniikit indeksit (engl. unique index) ovat erilaiset. Indeksejä käytin varmistamaan, että käyttäjät pystyvät lisäämään ainoastaan yhden ilmoituksen samalle tuotteelle. Edellä mainitut taulut viittaavat *Regions*, *keytypes* ja *currencies*-tauluihin, jotka sisältävät vaihtoehdot tuotteen aluelukitukselle, tuotteen tyypille (aktivointikoodi tai erilaiset steam-lahjat) sekä valuutoille, jolla tuotteen maksu halutaan suorittaa.

5.2.6 Lokitietoa tai tilapäistä tietoa sisältävät taulut

Nämä viisi taulua eivät liity keskeisesti järjestelmän toimintaan. Niihin tallennetaan tietoa, joka voi esimerkiksi auttaa löytämään järjestelmässä olevia bugeja.

Cron_log-tauluun lisätään merkintä epäonnistuneista cron jobeista. *Db_errors*-tauluun taas epäonnistuneet tietokantakyselyt. *Scanlist*-taulusta löytyvät epäonnistuneet käyttäjäprofiilipäivitykset. *Scanqueue*-taulua käytetään jonona käyttäjäprofiilien päivitykselle.

Logins-taulusta löytyy lista sivustolle kirjautuneista käyttäjistä. Kirjautumisen yhteydessä kirjataan käyttäjän ip-osoite, sekä tieto siitä onnistuiko kirjautuminen.



Kuva 17. Lokitietoa sisältävät taulut.

5.3 Tietokantahakujen optimointi

Tietokannan optimointia ei aina tule ajateltua suunnitteluvaiheessa, eikä pahimpia ongelmakohtia välttämättä huomaakaan ennen sivuston laajempaa testausta. Työssäni ensimmäinen ongelmakohta oli statistiikan laskeminen taulusta, johon oli talletettu kaikki pelit, jotka sivuston käyttäjät omistivat. Taulun koko oli tuolloin noin 1,6 miljoonaa riviä, ja hakujen suorittamiseen meni jopa kymmeniä sekunteja. Hakua pystyi optimoimaan esimerkiksi hakukriteereitä säätämällä ja liitettyjen taulujen järjestystä muuttamalla, mutta suurin apu oli tiedon jakamisesta useampaan tauluun.

Käytän asiaa kuvaavana esimerkkinä blogikirjoituksia sisältävää taulua. Taulua suunnitellessa voi kuulostaa järkeenkäyvältä sijoittaa kaikki kirjoitukseen liittyvä tieto samaan tauluun. Tietoa ei kuitenkaan kannata sijoittaa samaan tauluun sen enempää, kun hakujen kannalta on oleellista. Esimerkiksi uusimpien kirjoitusten listaamiseen ei tarvittaisi kun otsikko, kirjoituksen ID, kirjoitusaika sekä ehkä pieni osa kirjoituksesta. Kokonaisen blogikirjoituksen voi tallentaa toiseen tauluun, ja hakea sen ID:n perusteella, kun käyttäjä avaa kirjoituksen. Tietokantaa suunnitellessa ei välttämättä ajattele, että kenttä jota ei käytä hakuperusteena

vaikuttaisi hakunopeuteen, mutta pitkät tekstikentät taulussa voivat aiheuttaa jopa tyhjinä sen, että haku tehdään palvelimen kiintolevyllä, eikä paljon nopeammin toimivassa RAM-muistissa.

Oma tietokantahakuni sisälsi myös paljon laskuja ja muuta ylimääräistä, näiden tulokset sisällytin uuteen tauluun, jonka päivitin vartin välein cron jobilla. Näin niitä ei tarvitsisi tehdä uudestaan aina, kun sivu ladataan. Lisäksi tiedon perusteella luotu sivu pidetään palvelimella muistissa (engl. cache) parisen minuttia, jotta sama sivu voidaan näyttää usealle käyttäjälle tekemättä useita tietokantahakuja. Nykyisin pelilistat sisältävä taulu on paisunut jo yli viiteen miljoonaan riviin. Sivulataukset ovat kuitenkin säilyneet tarpeeksi nopeina.

5.4 Apache mod_rewrite ja yksinkertainen front controller

Mod_rewrite on Apache HTTP-palvelinohjelmaan sisältyvä moduuli, jolla voi uudelleenohjata palvelimelle saapuvia pyyntöjä. Sitä käytetään usein muuttamaan sivuston URL-osoitteet luettavampaan muotoon. Kun normaalisti vaikkapa verkkokaupan tuotesivun osoite olisi ”kauppa.fi/tuotteet.php?id=395”, mod_rewriteä käyttämällä siitä saadaan helposti esimerkiksi ”kauppa.fi/tuote/395”. Tästä on hyötyä paristakin syystä. Selkeät osoitteet ovat mieluisempia sivuston käyttäjille, ja hakukoneet eivät välttämättä indeksoi sivuja, joihin ei pääse ilman GET-muuttujia.

Käytin mod_rewriteä siihen, että kaikki palvelimelle saapuvat pyynnöt ohjattaisin tietylle PHP-prosessille, joka puolestaan luo pyynnön perusteella oikean sivun käyttäjälle.

Mod_rewriten käyttöönotto onnistuu yksinkertaisimmillaan sijoittamalla seuraavat komennot www-kansion juuressa olevaan .htaccess tiedostoon.

```
RewriteEngine On
RewriteBase /

RewriteRule ^(img|sitemap\.xml|gif|jpg|png|jpeg|css|js|swf)$ /$1
[L,NC]
RewriteCond %{REQUEST_FILENAME} !-f [OR]
RewriteCond %{REQUEST_FILENAME} !-d [OR]
```

```
RewriteCond %{REQUEST_FILENAME} !-l
RewriteRule ^.*$ controller.php [L]
```

Näin kaikki palvelimelle tulevat pyynnöt ohjautuvat controller.php prosessiin. Controller.php puolestaan käy läpi saapuneen pyynnön ja voi jakaa sen osiin seuraavasti:

```
$req_url = strtok($_SERVER['REQUEST_URI'], '?');
$req_url = strtok($req_url, '&'); // poistetaan pyynnöstä GET-muuttujat
$requestURI = explode('/', $req_url); // jaetaan pyyntö osiin
$scriptName = explode('/',$_SERVER['SCRIPT_NAME']);
for($i= 0;$i < count($scriptName);$i++) // poistetaan skriptiin osoittava osa jos sellainen löytyy
{
        if ($requestURI[$i] == $scriptName[$i])
                {
                    unset($requestURI[$i]);
                }
}
$command = array_values($requestURI); // lopuksi kootaan taulukko (engl. array) pyynnössä olevista käskyistä
```

Kuva 18. Palvelinpyynnön käsittely

Esimerkiksi URLista "kauppa.fi/tuote/500/tiedot" tulisi siis seuraavanlainen taulukko:

Array ([0] => tuote [1] => 500 [2] => tiedot)

Taulukkoa voidaan sitten käyttää vaikkapa yksinkertaisessa switch-lauseessa kokoamaan pyynnön mukainen sivu:

```
switch($command[0])
{
case "tuote":
        if (!isset($command[1]) || !is_numeric($command[1]))
        {throw new Exception('404');}
        else
        {
                $page_options['tuote_id'] = $command[1];
                require_once 'tuote.php';
        }
```

Kuva 19. Esimerkki pyynnössä olevien käskyjen käsittelystä.

5.5 Rekisteröinti, Steam OpenID

Palveluun rekisteröityminen toteutettiin käyttäen OpenID-menetelmää. Steam tarjoaa OpenID-tunnisteita, joilla voidaan yhdistää palvelun käyttäjä tiettyyn Steam-tiliin. Käyttäjän ei siis tarvitse luoda palveluun erillistä käyttäjätunnusta, vaan voi käyttää sisäänkirjautumiseen jo olemassa olevaa Steam-tiliään.

OpenID-tunnistautuminen toimii yksinkertaistettuna siten, että palvelu ohjaa käyttäjän Steamin (tai jonkun muun OpenID-toimittajana (engl. OpenID provider) toimivan palvelun) kirjautumissivulle (tai pyytää lupaa kirjaumiseen jos käyttäjä on jo kirjautunut sisään aikaisemmin). Onnistuneen autentikoinnin jälkeen Steam ohjaa käyttäjän takaisin pyynnön lähettäneelle palvelimelle, antaen samalla palvelulle käyttäjän Steam-tunnuksen ID-tunnisteen.

Tunnuksen salasana ja muu tieto annetaan ainoastaan OpenID-toimittajan palvelimelle, eikä sitä luovuteta eteenpäin OpenID:tä käyttävälle sovellukselle.

6 VERKKOSOVELLUKSEN TIETOTURVA

Jotta käyttäjät voivat käyttää verkkosovellusta, sen on oltava yhteydessä Internetiin. Tästä syystä on tärkeää varmistaa, että sovellus on mahdollisimman tietoturvallinen, ja että sen käyttäjät voivat suorittaa vain ennalta määrättyjä toimintoja. Jos ohjelmistossa on haavoittuvuuksia, pahimmassa tapauksessa hyökkääjä saa koko palvelimen tietoineen haltuunsa. Tässä luvussa käsitellään erilaisia tietoturvaongelmia, joita hyökkääjä voi käyttää hyväkseen.

6.1 Syötteiden validointi

Verkkosovelluksia käsittelevissä teksteissä yksi sääntö toistuu usein; älä koskaan luota käyttäjän syöttämään tietoon.

Tämä tarkoittaa sitä, että mitään ulkopuolista ei tule käyttää sellaisenaan missään, jossa epämääräinen syöte voi aiheuttaa harmia. Syöte on myös validoitava tarpeiden ja kontekstin mukaisesti jokaisessa osassa ohjelmaa.

Esimerkiksi jos käyttäjä joutuu valitsemaan pudotusvalikosta jotain, jonka odotetaan olevan tietokannassa tiettyä valintaa kuvaava kokonaisluku, voidaan syötettä vastaanotettaessa tarkistaa palvelimella vain, että kyseessä on todellakin kokonaisluku. Syvemmällä ohjelmassa, ennen tietokantaan syöttämistä voidaan sitten tarkistaa, että luvulle on sitä vastaava tieto tietokannassa.

Yleisesti tehokas validointitapa on esimerkiksi säännöllisillä lausekkeilla (engl. regular expression) toteutettu ”whitelist”, jolloin jokainen osa lomaketta hyväksyy tietyt sallitut merkit, eikä mitään muuta. Huomioitavaa on, että varsinainen validointi on aina tehtävä palvelinpuolella. Selaimessa toimiva, esimerkiksi javascript tai html-pohjainen validointi on helposti kierrettävissä eikä suojaa sovellusta käytännössä ollenkaan. Sitä tulee käyttää vain ohjeistuksena sovelluksen käyttäjälle. (Brady 2017)

6.2 SQL-injektio

SQL-injektio on yleinen tietoturva-aukko, jonka avulla hyökkääjä voi lisätä omia komentojaan tietokannassa suoritettavaan kyselyyn. Haavoittuvainen SQL-kysely login-sivulla voisi näyttää esimerkiksi tälläiseltä:

```
SELECT user_id FROM users

WHERE username = '$username' AND password = '$password'
```

Koska muuttujat liitetään suoraan kyselyyn, hyökkääjä voi kirjautua vaikkapa käyttäjänimellä “admin' --“, jolloin kyselyn salasanaa vertaava osa muuttuu kommentiksi:

```
SELECT user_id FROM users

WHERE username = 'admin' -- AND password = '$password'
```

Hyökkääjä pystyy tämän jälkeen kirjautumaan sivustolle admin-tunnuksella tietämättä sen salasanaa. SQL-hyökkäys voi olla erittäin tuhoisa, eikä aina rajoitu käyttäjätunnuksien kaappaamiseen. Sillä voidaan mahdollisesti myös tuhota sovelluksen tietokanta, muokata tietoa tai varastaa sen sisältö. (Brady, 2017)

SQL-injektiolta suojautuminen on onneksi yksinkertaista PHP:ssä. Jos tietokannan käsittelyyn käytetään PDO-rajapintaa, voidaan jokainen kysely luoda käyttäen ”valmisteltuja lauseita” (engl. prepared statements). Tällä tarkoitetaan sitä, että kyselyn jokainen muuttuva parametri korvataan kysymysmerkillä tai parametrin nimellä. Käyttäjän syöttämä arvo lisätään kyselyä suoritettaessa tiettyyn kohtaan kyselyä, eikä tietokanta voi erehtyä suorittamaan arvossa mahdollisesti olevia ylimääräisiä komentoja, koska sitä ei käsitellä lainkaan komentona. (The PHP Group, n.d.)

```
try
{
        $con = new PDO("mysql:host=$host;dbname=$database", $dbuser, $dbpass);
        $con -> setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $con -> setAttribute(PDO::ATTR_EMULATE_PREPARES, false);

        $sql = "INSERT INTO users (username, password, email) VALUES (?,?,?)";
        $values = array($user['id'], $user['passwd'], $user['email']);
        $query = $con->prepare($sql);
        $query->execute($values);
}

catch (PDOException $e)
{
        echo "Error: " . $e->getMessage();
}
```

Kuva 20. Yksinkertainen esimerkki rivin lisäämisestä tauluun kun käytetään PDO-rajapintaa.

6.3 Cross-Site Request Forgery

CSRF on hyökkäysmenetelmä, jossa hyökkääjä pyrkii saamaan käyttäjän tekemään jonkin autentikointia vaativan toiminnon ilman tämän suostumusta. Jos käyttäjä saadaan esimerkiksi avaamaan linkki hyökkääjän ylläpitämälle sivustolle, jossa oleva skripti lähettää pyynnön käyttäjän nimissä toisen sivuston palvelimelle. Jos käyttäjä on hyökkäyshetkellä kirjautunut hyökkäyksen kohteena olevalle sivustolle, eikä sivustolla käytetä suojakeinoja CSRF-hyökkäyksien estämiseksi, hyökkääjän väärentämä pyyntö suoritetaan käyttäjän sisäänkirjautumista käyttäen.

Hyökkäyksellä voidaan suorittaa mikä tahansa ylimääräistä vahvistusta tarvitsematon pyyntö, esimerkiksi käyttäjän tunnuksien poisto, tietojen poistaminen tai viestien lähettäminen.

Yleinen tapa suojautua CSRF-hyökkäyksiltä on sisällyttää lomakkeisiin piilotettu kenttä, joka sisältää käyttäjälle yksilöllisen avaimen (engl. token), jota säilytetään käyttäjän evästeissä tai istunnon (session) tiedoissa. Avain voidaan myös muuttaa tietyin väliajoin, jotta sen arvaaminen olisi vaikeampaa. Pyyntöihin sisällytettyä avainta verrataan käyttäjän istuntotiedoissa olevaan avaimeen ja jos se on väärä, pyyntö jätetään suorittamatta palvelimella.

6.4 Cross-Site Scripting (XSS)

Cross-Site Scripting on ehkä yleisin verkkosovelluksissa esiintyvä haavoittuvuus. XSS-hyökkäyksessä hyökkääjä pyrkii sijoittamaan verkkosivulle ylimääräistä HTML- tai javascript-koodia. Aukko johtuu usein validoimattomasta syötteestä tai GET-parametrissa olevasta tiedosta, joka esitetään sivulla sellaisenaan.

Aukon kautta sivulle lisätty javascript-koodi voi esimerkiksi varastaa käyttäjän evästeet, uudelleenohjata eri sivustoille, tai suorittaa muuta haitallista koodia sivustolla vierailevien käyttäjien selaimissa. (Nixon, 2015, 259-260)

XSS-hyökkäyksien estämiseksi on toimivia tapoja. PHP-pohjaisessa verkkosovelluksessa voidaan kaikkeen sivulla esitettävään epäluotettavaan sisältöön käyttää htmlentities-funktiota, joka neutralisoi mahdollisen syötteessä olevan koodin. Funktio toimii siten, että se muuttaa erikoismerkit niiden HTML-entiteeteiksi, jolloin selain ei enää voi pitää niitä ajettavana koodina. (Brady, 2016)

On huomioitavaa, että htmlentities ei suojaa muuta kuin HTML-kontekstissa olevaa tietoa. Jos epäluotettavaa tietoa halutaan sijoittaa esimerkiksi Javascript-koodin sisälle, on se käsiteltävä eri tavalla, ennen kuin se on turvallista. (Williams, Manico & Mattatall 2016)

6.5 Käyttäjien hallinta

Useimmiten osa verkkopalvelun ominaisuuksista halutaan rajata tietylle osalle käyttäjistä, jolloin vaaditaan palveluun rekisteröimistä. Kirjautumisen avulla voidaan myös pitää tietyt tallennetut tiedot luottamuksellisina.

Käyttäjän autentikointiin käytetään tavanomaisesti käyttäjänimeä sekä salasanaa. Käyttäjä syöttää verkkosovellukseen kirjautumistietonsa ja sovellus vertaa tietoja tietokannassa oleviin tietoihin. Jos vastaavat tiedot löytyvät tietokannasta, käyttäjä kirjataan sisään. (Keary, Manico, Goosen, Krawczyk, Neuhaus & Morales 2016)

HTTP-protokollan tilattomuudesta johtuen käyttäjän tunnistamiseen käytetään normaalisti evästeitä tai istuntoja (engl. session). Käytännössä tämä toimii siten,

että käyttäjän laitteella sijaitsevaan evästeeseen tallennetaan käyttäjän istunnon ID, kun taas kriittisemmät istuntoon littyvät tiedot (esim. käyttäjän yksilöivä ID) pysyvät tallessa palvelimella. Näin istunnon tietoja ei pääse vapaasti manipuloimaan asiakaspuolelta.

Vaihtoehtoja tavalliselle käyttäjänimellä/salasanalla tunnistautumiselle ovat mm. OpenID, OAuth, mobiilivarmenne, ym. (Keary, Manico, Goosen, Krawczyk, Neuhaus & Morales 2016)

6.6 Salasanojen tallentaminen tietokantaan

Jos kirjautumiseen käytetään salasanoja ja ne halutaan tallentaa tietokantaan, tulee tietoja tietomurtojen yleisyyden vuoksi varastoida tietovarkauden mahdollisuutta silmälläpitäen.

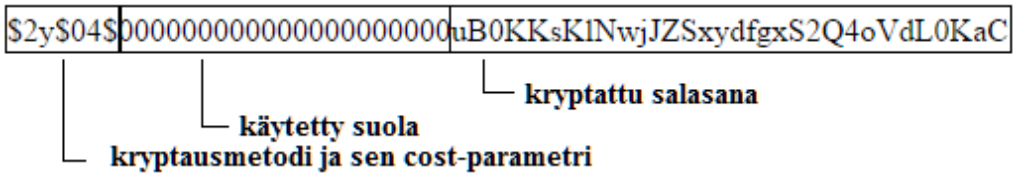
Salasanojen kannalta tämä tarkoittaa sitä, että ne tulee salata asianmukaisesti yksisuuntaista salausta käyttäen, jotta onnistunut hyökkääjä ei saa haltuunsa selkokielistä listaa salasanoista. Selkokielisen salasanalistan vuotaminen ei olisi sinällään välttämättä vaarallista, mutta käyttäjät voivat usein käyttää samaa salasanaa myös muissa palveluissa, jolloin myös näiden käyttäjätunnukset vaarantuvat. (Steven & Manico 2016)

Huonosti kryptattujen salasanojen ”avaamiseen” on olemassa listoja (engl. rainbow table), jotka sisältävät selkokielisen salasanan ja sen kryptatun version eli tiivisteen (engl. hash). Tämän vuoksi salasanaan tulee ennen kryptausta lisätä myös yksilöllinen merkkijono eli suola. Suolan avulla helpostikin arvattavasta lyhyestä salasanasta tulee huomattavasti pidempi ja siten vaikeampi avata. Valmiita salasanalistoja ei voi myöskään käyttää kryptauksen avaamiseen, jos sattumanvaraisesti generoitu suola on lisätty. (Steven & Manico 2016)

Salasanan kryptaamiseen voidaan käyttää esimerkiksi PHP:n password_hash -funktiota. Funktio generoi salasanalle yksilöllisen suolan tai käyttää siihen ennaltamääriteltyä merkkijonoa. On parempi antaa funktion generoida automaattisesti joka salasanalle oma suola, koska se ja kryptaamiseen käytetyt asetukset sijoitetaan joka tapauksessa funktion luoman hashin alkuun. Tästä syystä

suolaa ei tarvitse myöskään erikseen tallentaa tietokantaan. Kirjoitushetkellä password_hash -funktio käyttää oletuksena kryptaukseen bcrypt-algoritmia. Oletusasetus on suunniteltu siten, että uudemmat PHP-versiot voivat käyttää oletuksena automaattisesti eri algoritmia, jos bcryptistä myöhemmin luovutaan. (The PHP Group, n.d.)

Kirjautumisvaiheessa käyttäjän syöttämän salasanan hashia voi verrata tietokannassa olevaan PHP:n password_verify -funktion avulla. Salasanan vertaamiseen ei ainakaan aloittelijan kannata tehdä omaa funktiota, koska salasanan vertaamiseen kuluneesta ajasta voi mahdollisesti päätellä miten lähellä oikeaa salasanaa ollaan, jos sitä ei ole tehty oikein. Password_verify -funktio ei ole altis ajoitushyökkäyksille. (The PHP Group, n.d.)



Kuva 21. password_hash -funktion luoma tiiviste salasanasta ’salasana’ nollista koostuvalla suolalla.

6.7 Istunnon kaappaus

Istunnon kaappaamiseen on useita eri tekniikoita. Yksi yleisimmistä tavoista XSS-hyökkäyksen lisäksi on ohjata käyttäjä sivustolle linkillä, joka asettaa samalla käyttäjän istunnolle tietyn ID:n. Tätä hyökkäystä kutsutaan nimellä Session Fixation. Hyökkääjän käyttämä linkki voisi näyttää esimerkiksi tältä: http://www.example.com?PHPSESSID=1c23b45a6

Linkkiä klikattuaan käyttäjä siirtyy osoitteeseen example.com ja asettaa samalla istunnon ID:ksi merkkijonon ”1c23b45a6”. Myöhemmin hyökkääjä voisi käyttää samaa istunnon tunnistetta, ja jos käyttäjä kirjautui sivustolle sisään, päästä sivustolle tämän tunnuksilla. (Owasp.org, N.D.) Käyttäjän istunnon suojaamiseksi tämäntyyppisiltä hyökkäyksiltä tulee SessionID generoida uudelleen kirjautumisen jälkeen. PHP-sovelluksissa tämän voi tehdä session_regenerate_id -funktiolla.

Funktiolle on kuitenkin huomattava antaa parametri TRUE, tai muuten vanhaa istuntoa ei tuhota palvelimelta.

Jos istunnot halutaan suojata vielä paremmin, niihin voi lisätä muita käyttäjään liittyviä tietoja, kuten IP-osoite tai selaimen versionumero. Kun käyttäjä saapuu sivustolle, voi heidän IP-osoitettaan tai muuta tietoa verrata istunnossa oleviin tietoihin. Jos jokin on muuttunut, käyttäjän voi kirjata ulos tuhoamalla istunnon. (Siles 2016)

Kuvassa 12 näytän, miten tämän voi toteuttaa käytännössä.

```
Kirjautumishetkellä käyttäjän IP-osoite tallennetaan
RIPEMD-128 algoritmilla hashattuna yhdeksi istunnon muuttujista:
$_SESSION['check'] = hash('ripemd128', $_SERVER['REMOTE_ADDR']);

Myöhemmin jokainen sivulataus tekee seuraavan tarkistuksen:
session_start();
if (isset($_SESSION['check']))
{
        // jos IP-osoite on muuttunut alkuperäisestä,
        // kutsutaan different_user -funktiota:
        if ($_SESSION['check'] != hash('ripemd128', $_SERVER['REMOTE_ADDR']))
        different_user();
}

Funktio tuhoaa käyttäjän istunnon, kirjaten tämän ulos:
function different_user()
{
        $_SESSION = array();
        setcookie(session_name(), '', time() - 2592000, '/');
        session_unset();
        session_destroy();
}
```

Kuva 22. Käyttäjän IP-osoitteen muuttumisen havaitseminen ja istunnon tuhoaminen PHP:tä käyttäen.

6.8 Istuntojen säilyttäminen palvelimella

On kriittistä pitää istuntotiedostojen tallennussijainti hakemistossa, joka ei ole julkisesti saatavilla. Oletuksena PHP tallentaa tiedostot palvelimen tmp-hakemistoon. Tämä ei ole turvallista, jos samalla palvelimella on useita käyttäjiä, joilla on kirjoitusoikeudet samaan tmp-hakemistoon.

Hyvä varotoimi on joka tapauksessa asettaa tallennushakemisto manuaalisesti käyttäen PHP:n session.save.path -asetusta. (The PHP Group, n.d.)

```
ini_set('session.save_path', $_SERVER['DOCUMENT_ROOT'] . '/../sessions');
```

Kuva 23. Istuntojen tallennushakemiston polun osoittaminen webrootin yläpuolella sijaitsevaan ’sessions’-hakemistoon PHP-sovelluksen config-tiedostossa.

6.9 Tiedon eheyden varmistaminen käyttäen transaktioita

Jos tietokantaan lisätään tietokokonaisuus, joka sijoitetaan usealle eri riville tai eri tauluihin, on tiedon eheyden varmistamiseksi hyvä käyttää transaktioita. Transaktioita tukee kirjoitushetkellä ainakin InnoDB-tietokantamoottori (engl. storage engine).

MySQL-transaktio voi sisältää useita SQL-lauseita, jotka suoritetaan vain väliaikaisesti, kunnes kaikki transaktioon kuuluvat lauseet sekä transaktion lopetuskomento on onnistuneesti suoritettu. Jos jokin kyselyistä epäonnistuu, voidaan tietokannalle antaa tiedon tallentavan commit-komennon sijaan rollback-komento joka palauttaa tietokannan transaktion aloittamista edeltäneeseen tilaan. Näin vältytään keskeneräisen tietokokonaisuuden päätymiseltä tietokantaan.

Esimerkkinä tilanteesta, jossa halutaan välttää epäeheää tietoa käytän pankin tietokannassa olevaa tilisiirtoon liittyvää tietokokonaisuutta. Ilman transaktioita voisi tietokantaan päätyä virheen takia vain osa suoritetuista kyselyistä, jolloin asiakkaan tililtä voitaisiin poistaa siirrettävä summa, ilman että sitä lisättäisiin vastaanottajan tilille. Transaktioita käyttämällä voidaan olla varmoja siitä, että mistään tietokokonaisuudesta ei voi jäädä puuttumaan tärkeitä osia. (Nixon 2015, 223-225)

6.10 Verkkosovelluksen kehittäminen ja virheraportointi

PHP:n virheraportointi on sovelluksen kehitysvaiheessa erittäin hyvä apu, mutta jos sovellus on käytettävissä julkisesta verkosta, ei sen saisi enää antaa käyttäjälle tarkkoja virheilmoituksia tai stack traceja. Virheilmoitusten sisällön avulla hyökkääjä voi päätellä sovelluksen asetukset, rakenteen ja jopa päästä käsiksi käyttäjätunnuksiin tai tietokantaan. Käyttäjälle tulee näyttää korkeintaan geneerisiä virheilmoituksia, jotka eivät paljasta liikaa järjestelmän toiminnasta.

Näkyvät virheilmoitukset voi siis tuotantoympäristössä (engl. production environment) kytkeä pois päältä, ja sen sijaan kirjoittaa ne lokitiedostoon. Lokitiedosto tulee tallentaa webrootin ulkopuolelle, tai vähintään suojata jollain pääsynhallintamekanismilla, jotta ulkopuoliset eivät pääse lukemaan sitä. (Owasp.org 2016)

Virheilmoituksien pois kytkemiseen on useita tapoja. Jos php.ini -asetustiedostoa ei pääse palvelimella muokkaamaan, voi oman sovelluksen asetustiedostossa kytkeä näkyvät virheilmoitukset pois päältä vaikkapa seuraavasti, jolloin ne kirjataan vain lokitiedostoon:

```
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);
ini_set('log_errors', 1);
ini_set('display_errors', 0);
```

Vaihtoehtoisesti voi virheilmoitukset piilottaa myös Apachen avulla. Se onnistuu lisäämällä .htaccess-tiedostoon seuraavat komennot:

```
# piilota virheilmoitukset
php_flag display_startup_errors off
php_flag display_errors off
php_flag html_errors off
php_value docref_root 0
php_value docref_ext 0
# kirjaa ilmoitukset virhelokiin
php_flag  log_errors on
php_value error_log /home/username/php-logs/php_errors.log
```

7 JOHTOPÄÄTÖKSET

Opinnäytetyötä tehdessä tuli selväksi, että kohtuullisen tietoturvallisuuden saavuttaminen PHP-ohjelmointikieltä käyttäen ei ole erityisen vaikeaa. Itselläni ei ollut ohjelmoinnista juurikaan kokemusta ennen työn aloittamista, mutta tietoa löytyi verkosta runsaasti, ja aukkoja omassa osaamisessa oli sen avulla helppo täyttää.

Verkossa (ja miksei myös kirjoissa) olevan tiedon kanssa kannattaa olla tarkkana, koska tieto voi vanhentua yllättävänkin nopeasti tekniikan kehittyessä. Esimerkiksi vanhoissa kirjoissa usein suositeltu MD5-hashaus on jo auttamattoman vanha ja huono tapa salata mitään. Yksi ongelma suositun ohjelmointikielen käyttämisessä on myös se, että koska asioista kirjoittaa laajempi skaala ihmisiä, mukaan mahtuu myös paljon huonoja ratkaisuja. Verkoista löydettyyn tietoon kannattaa siten kohdistaa riittävästi lähdekriittisyyttä.

Verkkosovellusten tietoturva on erittäin laaja kokonaisuus käsitellä, eikä tähän työhön oikeastaan tullut kuin lyhyt pintaraapaisu siitä. Riippuu kuitenkin oman sovelluksen skaalasta, käyttötarkoituksesta ja tallennetun tiedon luottamuksellisuudesta, millaisia tietoturvaratkaisuja kannattaa harkita käyttävänsä. Tässä työssä käsitellyt tietoturvaan liittyvät asiat ovat mielestäni tärkeitä lähes kaikille verkkosovelluksille, jotka ovat julkisessa käytössä.

8 LÄHTEET

Bach, C. N.D. Tablesorter 2.0 documentation. Viitattu 21.3.2017. http://tablesorter.com/docs

Brady, P., 2016. Cross-Site Scripting (XSS). Survive The Deep End: PHP Security. Viitattu 29.10.2016. http://phpsecurity.readthedocs.io/en/latest/Cross-Site-Scripting-(XSS).html

Brady, P., 2017. Injection Attacks. Survive The Deep End: PHP Security. Viitattu 20.3.2017. http://phpsecurity.readthedocs.io/en/latest/Injection-Attacks.html#sql-injection

Brady, P., 2017. Input Validation. Survive The Deep End: PHP Security. Viitattu 19.3.2017. http://phpsecurity.readthedocs.io/en/latest/Input-Validation.html

CSS3 Introduction. N.D. W3schools.com. Viitattu 19.2.2016. http://www.w3schools.com/css/css3_intro.asp

Description of the database normalization basics. 2013. Microsoft. Viitattu 21.2.2016. https://support.microsoft.com/en-us/kb/283878

HTML data-* Attributes. N.D. W3schools.com. Viitattu 23.3.2017. https://www.w3schools.com/tags/att_global_data.asp

HTML5 Video. N.D. W3schools.com. Viitattu 19.2.2016. http://www.w3schools.com/html/html5_video.asp

Improper Error Handling. Open Web Application Security Project. N.D. Owasp.org. Viitattu 11.11.2016. https://www.owasp.org/index.php/Improper_Error_Handling

Keary, E., Manico, J., Goosen, T., Krawczyk, P., Neuhaus, S. & Morales, M. 2016. Authentication Cheat Sheet. Open Web Application Security Project. Viitattu 3.11.2016. https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Introduction

MySQL Workbench: Visual Database Design. N.D. Oracle Corporation. Viitattu 1.11.2017. https://www.mysql.com/products/workbench/design/

Nixon, R. 2015. Learning PHP, MySQL & JavaScript With jQuery, CSS & HTML5 (4. painos). O'Reilly Media.

Otto, M. 2011. Bootstrap from Twitter. Viitattu 6.9.2016. https://blog.twitter.com/2011/bootstrap-twitter

PHP: password_hash. N.D. The PHP Group. Viitattu 3.11.2016. http://php.net/manual/en/function.password-hash.php

PHP: password_verify. N.D. The PHP Group. Viitattu 4.11.2016. https://secure.php.net/manual/en/function.password-verify.php

PHP: Prepared statements and stored procedures. N.D. The PHP Group. Viitattu 22.2.2017. http://php.net/manual/en/pdo.prepared-statements.php

PHP: session_save_path. N.D. The PHP Group. Viitattu 7.11.2016. https://secure.php.net/manual/en/function.session-save-path.php

Session Fixation. Open Web Application Security Project. N.D. Owasp.org. Viitattu 5.11.2016. https://www.owasp.org/index.php/Session_fixation

Siles, R. 2016. Binding the Session ID to Other User Properties. Session Management Cheat Sheet. Open Web Application Security Project. Viitattu 7.11.2016. https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Binding_the_Session_ID_to_Other_User_Properties

Steven, J. & Manico, J. 2016. Password Storage Cheat Sheet. Open Web Application Security Project. Viitattu 4.11.2016. https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

Suehring, S. & Valade, J. 2013. PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies. Hoboken, New Jersey. John Wiley & Sons, Inc.

Ullman, L. 2012. PHP and MySQL for Dynamic Web Sites (4. painos). Berkeley. Peachpit Press.

Usage of JavaScript libraries for websites. 2017. Viitattu 9.6.2017. https://w3techs.com/technologies/overview/javascript_library/all

Williams, J., Manico, J. & Mattatall, N. 2016. XSS (Cross Site Scripting) Prevention Cheat Sheet. Open Web Application Security Project. Viitattu 2.11.2016. https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet