Mikko Yrjänä

Verkkopalvelun suojaus haittaliikenteeltä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

5.3.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Mikko Yrjänä Verkkopalvelun suojaus haittaliikenteeltä 27 sivua 5.3.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja(t)	Yliopettaja Janne Salonen

Insinöörityön tarkoituksena oli kehittää sovellus, jolla voitaisiin hallita ja valvoa verkkosivuille saapuvia vierailijoita. Opinnäytetyössä käydään läpi tarpeet sovellukselle, hieman sovelluksen suunnittelua, sovelluksen ohjelmointiin käytetyt tekniikat sekä varsinaiset keinot, joilla sovellus erottelee haitallisen liikenteen tavallisesta.

Työtä pyrittiin kuvaamaan laajamittaisesti menemättä liian yksityiskohtaisesti ohjelmointipuolen asioihin, jotta sovelluksen kokonaiskuva hahmottuisi paremmin työn edetessä. Ensimmäisenä asiana käytiin läpi, millaiseen ympäristöön sovellusta tarvitaan, jonka jälkeen tutustuttiin sovelluksessa käytettäviin tekniikoihin sekä sovelluksen varsinaiseen rakenteeseen. Viimeisenä esiteltiin hieman, miten sovelluksen ohjaus tuotannossa tapahtuu.

Työn tuloksena asiakkaalle valmistui toimiva ja kattava sovellus, joka vastasi toivottua tulosta kiitettävästi.

Avainsanat	Tietoturva, PHP, Haittaliikenne, Verkkoliikenteen valvonta

Author(s) Title Number of Pages Date	Mikko Yrjänä Protecting an online service from fraudulent traffic 27 pages 5.3.2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data Networks
Instructor(s)	Janne Salonen, Principal Lecturer

This thesis covers the design and functionality of a PHP-based software made to protect online services from fraudulent visitors. The thesis explains the needs for the software, a little about the software design, technologies used to develop the software and the actual methods the software uses to differentiate between legitimate and fraudulent traffic.

The thesis describes the work done from a general perspective without explaining the programming side in much detail. The first part of the thesis deals with the environment for which the software will be designed and the clients campaign structures to help better understand the need for the software.

The second part covers the technologies used in the software and some of their history along with small examples of their use, after which the structure of the software is explained along with the most important part of the thesis, the actual detection modules. Finally the maintenance and use of the software is explained.

The end result from the thesis was a comprehensive application which met the client's needs perfectly.

Keywords	Information security, PHP, Fraudulent traffic, Web traffic monitoring

Sisällys

Lyhenteet

HTML	*HyperText Markup Language*. WWW-sivuilla käytetty merkintäkieli.
PHP	*PHP: Hypertext Preprocessor*. Yleinen ohjelmointikieli Web-palvelinympäristöissä.
SQL	*Structured Query Language*. IBM:n kehittämä standardoitu kyselykieli tietokannan kanssa kommunikointiin.
LAMP	*Linux, Apache, MySQL, Perl / PHP / Python*. Suosittu ohjelmistopaketti joka sisältää tarvittavat puitteet verkkosivujen ja -sovellusten julkaisemiseen.

1 Johdanto

Tämän insinöörityön tarkoituksena on suunnitella ja kehittää verkkoliikennettä valvova web-sovellus yritykselle, joka toimii pääosin Internetin kautta myymällä ja markkinoimalla erilaisia tuotteita. Tuotteita myydään pääasiassa tavallisille kuluttajille eikä myyntiä ole kohdistettu yrityksille lainkaan.

Tarve työlle tulee yrityksen kärsimistä suurista määristä haitallista ja asiatonta verkkoliikennettä sivuilleen, mikä ilmenee muun muassa väärinä tilaustietoina sekä kilpailijoiden kopioidessa kampanja- ja tilaussivujen sisältöä. Väärien tilaustietojen läpikäyminen aiheuttaa ylimääräistä työtä, ja kilpailijoiden kopioimat markkinointisivut heikentävät sivuilla saavutettua kilpailuetua.

Opinnäytetyössä suunnitellaan PHP-pohjainen sovellus, jolla haitallista liikennettä pystytään merkittävästi vähentämään ja hallitsemaan halutulla tavalla. Työssä selvitetään, mitä asiakas haluaa sovellukselta, millä tekniikoilla haitallista verkkoliikennettä voidaan erotella tavallisesta liikenteestä ja miten kokonaisuus tulee toimimaan. Työssä keskitytään sovelluksen suunnitteluun sekä ymmärtämiseen takertumatta kuitenkaan liian yksityiskohtaisesti ohjelmointitason asioihin.

Työ toteuttiin lähes kokonaan PHP:lla käyttäen kuitenkin hyväksi myös JavaScriptiä yhdessä moduulissa sekä MySQL:ää tietokantapuolen käsittelyssä. Mitään erityistä kehitysympäristöä ei työssä käytetty, vaan työkaluina toimivat pääasiassa Sublime Text 3 ja VIM (VI Improved) -tekstieditorit Windowsissa ja Linuxissa. Sovelluksen testauksessa ja ohjelmointivirheiden etsinnässä käytettiin PHP:n ja Nginx:in virhelogeja sekä koodiin tehtyjä välivaiheita.

Insinöörityön aihe valittiin mielenkiintoisen projektin sattuessa sopivasti kohdalle sekä ylipäätään mielenkiinnosta asiakkaan alaa kohtaan. Tarkoituksena oli ymmärtää asiakkaan Internet-markkinoinnin parissa päivittäin kohtaamia haasteita paremmin sekä pyrkiä ratkaisemaan näistä suurin ongelma haitallisen verkkoliikenteen kanssa.

2 Kampanjoiden ymmärtäminen

Ennen kuin sovellusta lähdetään kehittämään, on olennaista ymmärtää, mihin, missä ja miten sovellusta halutaan tarkalleen käyttää. Kehittämisen kannalta olennaisia asioita ovat muun muassa, minkä lähteiden kautta verkkosivuille tulee liikennettä ja miten asiakkaan kampanjat ylipäätään toimivat.

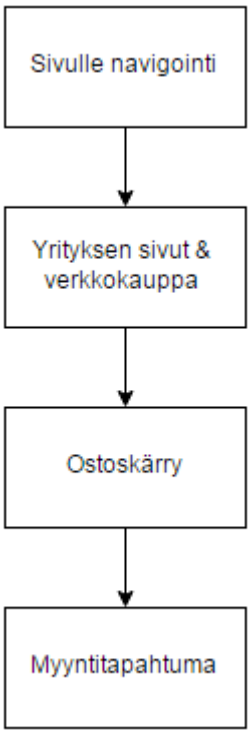
2.1 Kampanjoiden rakenne

Asiakkaan Internet-kampanjat koostuvat useammasta eri sivusta, jotka yhdessä muodostavat sivustokokonaisuuden, jolla houkutellaan asiakasta ostamaan tuote tai palvelu. Myyntiprosessi on hieman erilainen riippuen siitä, minkälaisen lähteen kautta asiakas saapuu myyntisivuille. Erilaisia lähteitä ovat esimerkiksi orgaaninen liikenne, maksettu liikenne ja sähköpostilistan kautta tuleva liikenne. Näistä ainoastaan orgaanisen liikenteen sivukokonaisuus eroaa maksetun ja sähköpostitse saadun liikenteen sivustoista.

2.1.1 Orgaaninen verkkoliikenne

Orgaaninen verkkoliikenne on sivulle tulevia vieraita, jotka saapuvat kaikista muista lähteistä kuin mainoskampanjoista. Tämmöisiä lähteitä ovat esimerkiksi Googlen tavalliset hakutulokset verkkosivun sisällön täsmätessä vierailijan hakutermiin ja erilaisten Facebook-linkitysten kuten jakojen ja tykkäyksien kautta sivulle löytäneet kävijät. Asiakkaan aikaisempien kokemusten mukaan orgaanisten lähteiden kautta tulevat kävijät etsivät usein lisää tietoa tuotteista ja palveluista, milloin järkevintä on tarjota runsaasti tietoa yrityksestä ja tarjonnasta sen sijaan, että ohjattaisiin nämäkin vierailijat suoraan myyntisivustolle, joka ei itsessään tarjoa runsaasti tietoa yrityksestä ja sen palveluista sekä tuotteista.

Ylläpitämällä yrityksen verkkosivuja, joista löytyy runsaasti tietoa, sekä kattavan verkkokaupan avulla parannetaan myös yrityksen näkyvyyttä hakukoneissa, jonka avulla orgaanista liikennettä löytää sivuille paremmin.

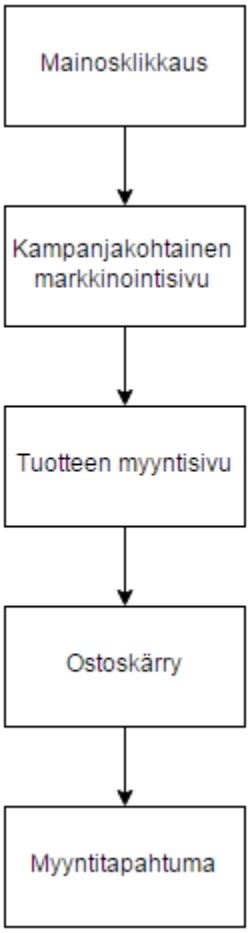


Kuva 1. Myyntiprosessi orgaaniselle liikenteelle

2.1.2 Maksettu verkkoliikenne

Maksettu verkkoliikenne koostuu vierailijoista, jotka päätyvät sivustolle klikattuaan ensin mainosta. Mainoksia voidaan näyttää esimerkiksi Googlessa ja Facebookissa kohdistaen niitä erilaisiin haluttuihin yleisöihin. Markkinointikampanjat on suunniteltu yksinomaan ostavia asiakkaita varten, joten maksetun verkkoliikenteen kautta tulevat vierailijat ohjataan suoraan markkinointisivuille tavallisen verkkokaupan tai yrityksen sivujen sijaan.

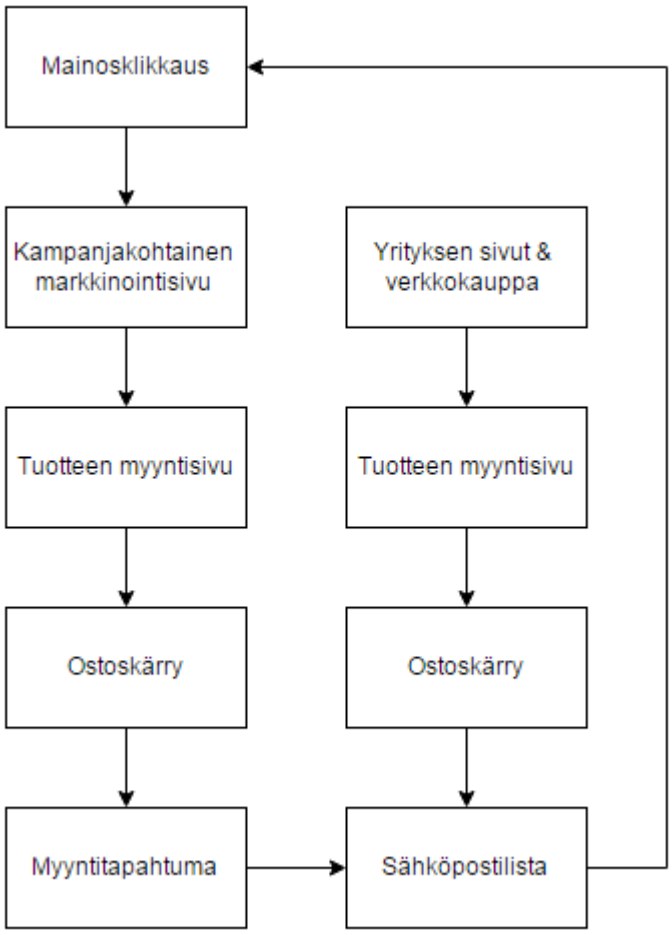
Mainoksen mainostaessa esimerkiksi teollisuustietokonetta asiakas on huomannut tehokkaimmaksi ohjata mainoksen klikkaaja suoraan teollisuustietokoneen markkinointisivulle, esimerkiksi perinteisen verkkokaupan sijaan.



Kuva 2. Myyntiprosessi maksettujen lähteiden kautta saapuville kävijöille

2.1.3 Sähköpostilistaliikenne

Sähköpostilistan kautta saapuneet vierailijat ovat aikaisemmin joko ostaneet jonkun tuotteen tai rekisteröityneet järjestelmään mutta jättäneet tuotteen ostamatta syystä tai toisesta. Näille asiakkaille markkinoidaan uusia tuotteita ja palveluita tai mahdollisesti lisäpalveluita jo olemassa olevaan tilaukseen käyttämällä hyväksi asiakkaiden aikaisemmin antamia sähköpostiosoitteita sekä asiakkaasta saatuja tietoja, kuten asuinmaata ja tilaushistoriaa.



Kuva 3. Asiakkaiden päätyminen sähköpostilistalle ja listan toimintaperiaate

3 Käytetyt tekniikat

Sovellusta kehittäessä tärkeää oli sen helppo siirrettävyys ja käytettävyys usealla eri palvelimella sekä toiminta mahdollisimman monessa selaimessa. Tästä johtuen kehityksessä pitäydyttiin tiukasti yleisimmissä ja laajasti tuetuissa tekniikoissa. Tekniikoiksi valittiin HTML, PHP, MySQL ja JavaScript. Kehitysympäristössä otettiin käyttöön uusin Debian GNU/Linux 7.7, koodinimeltään ”wheezy” ja HTML-palvelimeksi Nginx sekä PHP-tulkiksi PHP-FPM.

3.1 HTML

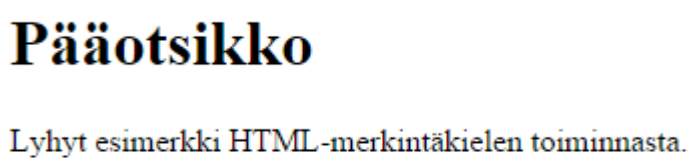
HTML on fyysikko Tim Berners-Leen vuonna 1990 luoma merkintäkieli, jota käytetään Internet-sivujen tekemiseen. HTML-tiedostot ovat käytännössä tekstitiedostoja, jotka sisältävät HTML-syntaksin mukaisia kuvausohjeita Internet-sivun ulkoasun ja rakenteen esittämistä varten. [1.]

Internet-selaimet, kuten Internet Explorer, Firefox, tai Google Chrome lukevat HTML-tiedostoja ja tulkkaavat niitä visuaalisiksi Internet-sivuiksi.

Lyhyt esimerkki HTML-merkintäkielellä toteutetusta verkkosivusta:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=utf-8">
</head>
<body>
<h1>Pääotsikko</h1>
Lyhyt esimerkki HTML-merkintäkielen toiminnasta.
</body>
</html>
```

Kun Internet-selain tulkitsee edellämainitun esimerkin, tuloksena on sivu:



Kuva 4. Esimerkki HTML-merkintäkielen toiminnasta

3.2 PHP

PHP on tanskalais-grönlantilaisen Rasmus Lerdorfin vuonna 1994 luoma komentosarjakieli, joka tarjoaa helpon tavan luoda dynaamista sisältöä web-sivuille [2.]. Esimerkiksi HTML:llä toteutettu kalenteri näyttää aina samaa päivää, kunnes HTML-tiedostoon käsin päivitetään uusi päivämäärä. PHP:n avulla pystytään dynaamisesti valitsemaan nykyinen päivämäärä aina kun verkkosivulla vieraillaan, ja täten näyttää aina oikea päivämäärä.

PHP-sovellusten rakentaminen on yksinkertaista, sillä palvelimelle asennettu PHP-moduuli tulkitsee koodin suoraan selväkielisestä tiedostosta ilman, että koodia tarvitsee erikseen kääntää kone- tai tavukoodiksi. PHP on todella laajasti tuettu eri käyttöjärjestelmillä ja alustoilla.

Perinteinen ”Hei Maailma!” -tervehdys toteutettuna PHP:lla:

```
<?php
$tervehdys = ”Hei Maailma!”;
print $tervehdys;
?>
```

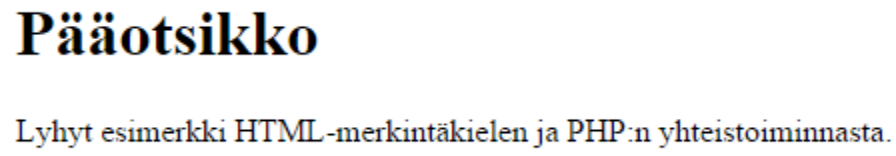
PHP-koodin käyttö yhdessä HTML:n kanssa on varsin yksinkertaista; kaikki ”<?php”- ja ”?>” -merkkijonojen sisällä oleva sisältö tulkitaan palvelimen puolella PHP-koodiksi jättäen selaimelle tulkittavaksi PHP-sovelluksen valmiin tulosteen sekä HTML-koodin.

```
<?php
$otsikko = “Pääotsikko”;
$teksti = “Lyhyt esimerkki HTML-merkintäkielen ja PHP:n
yhteistoiminnasta.”;
?>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=utf-8" >
</head>
<body>
<h1><?php echo $otsikko;?></h1>
<?php echo $teksti;?>
</body>
</html>
```

Palvelimen suorittaessa PHP-koodin selain näkee ainoastaan sovelluksen valmiin tulosteen, eli tässä tapauksessa:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=utf-8" >
</head>
<body>
<h1>Pääotsikko</h1>
Lyhyt esimerkki HTML-merkintäkielen ja PHP:n
yhteistoiminnasta.</body>
</html>
```

Joten Internet-selaimen tulostama sivu on selkokielinen:



Kuva 5. Esimerkki HTML ja PHP -kielien yhteistoiminnasta

3.3 MySQL

Työn tietokantaohjelmistoksi valittiin MySQL pitkälti suuren suosionsa ja hyvän dokumentoinninsa ansiosta. MySQL sisältyy myös suosittuun avoimen lähdekoodin ohjelmistopakettiin, joka tunnetaan nimellä LAMP.

MySQL on vuonna 1996 suomalaisen Michael Wideniuksen ja ruotsalaisen David Axmarkin yrityksen MySQL AB:n julkaisema, maailman toisiksi käytetyin avoimen lähdekoodin relaatiotietokantaohjelmisto. Ohjelmiston nimi tulee Michael Wideniuksen tyttären ”My”:n ja SQL-kyselykielen nimien yhdistelmästä [3.]. Vuonna 2008 MySQL AB myytiin Sun Microsystemsille, ja vuonna 2009 Oracle osti edelleen Sun Microsystemsin.

MySQL-tietokantoihin tehdään erilaisia operaatioita käyttäen SQL-lauseita, joilla dataa voidaan hakea, tallettaa, muokata ja poistaa. Tietokanta koostuu eräänlaisista tauluista, joiden sisältämä kenttien rakenne voidaan suunnitella käyttötarkoitusta varten. Eri taulujen välillä voi olla relaatioita, eli yhteyksiä kenttien välillä.

Sovelluksessa MySQL:ää käytetään PHP:n kanssa jatkuvasti datan tallennukseen ja lukuun. Kaikki sisääntulevat klikkaukset ja sovelluksen tekemät päätökset yksittäisten klikkauksien kohdalla tallennetaan tietokantaan mahdollista myöhempää tarkkailua varten. Myös osa sovelluksen käyttämistä moduuleista, kuten IP-estolista hakee estetyt IP-osoitteet tietokannasta.

3.4 JavaScript

JavaScript on dynaaminen komentosarjakieli, jonka avulla Internet-sivuille voidaan lisätä dynaamista toiminnollisuutta, kuten vaikka lomakkeiden automatisoitua virheiden

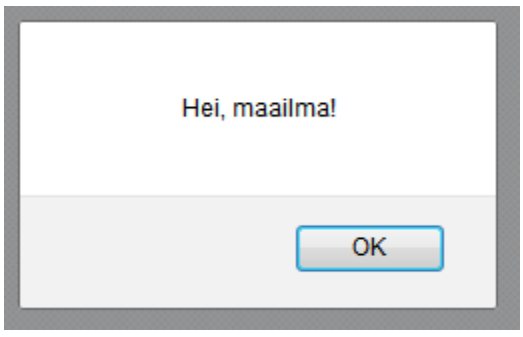
tarkistusta. Ensimmäinen versio JavaScript-moottorista julkaistiin vuonna 1995 Brendan Eichin, Netscape Communications Corporationin työntekijän luomana Netscape Navigator -selaimessa. [4.]

JavaScript sekoitetaan usein Java-ohjelmointikieleen samankaltaisen nimensä vuoksi, mutta ero näillä kahdella on huomattava. JavaScriptia käytetään useimmiten HTML:n lisänä, jotta voidaan tarjota lisäominaisuuksia, mihin HTML ei yksin pystyisi, ja Java taas on ohjelmointikieli, jolla voidaan tehdä kokonaisia ohjelmia ja pelejä, joita ajetaan Java Runtime Environment -ympäristön sisällä. Sekä JavaScriptiä että Javaa voidaan käyttää nettisivuilla, mutta Java-ohjelmat tarvitsevat aina Java Runtime Environmentin käyttäjän tietokoneella, kun JavaScriptiä tukee lähes kaikki selaimet sellaisenaan. [5.]

JavaScriptillä toteutettu ”Hei, maailma!” -ilmoitus, käyttäen aiemman HTML-esimerkin koodia pohjana.

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=utf-8" >
</head>
<body>
<script>
       alert('Hei, maailma! ')
</script>
</body>
</html>
```

Kun JavaScript:iä tukeva selain avaa sivun, hyppää esiin ilmoituslaatikko, jossa esimerkiksi on määritelty viesti, ”Hei, maailma!”:



Kuva 6. JavaScript:llä toteutettu ”Hei, maailma!” -ilmoituslaatikko

Jos JavaScript olisi kytketty pois päältä tai ei tuettu selaimessa, näkyisi ilmoituslaatikon sijaan vain tyhjä sivu. Kuitenkin esimerkiksi HTML:llä kirjoitettu koodi suoritettaisiin kuitenkin normaalisti, mikä on olennainen asia huomioida JavaScript-koodia käytettäessä sivujen lisänä.

4 Sovellus

Sovelluksen suunnittelussa huomioitiin erityisesti monipuoliset käyttöympäristöt ja tarve sovelluksen joustavuudelle. Koska sovelluksen käyttäjät ja ylläpitäjät ovat IT-alan ammattilaisia, ei graafiselle käyttöliittymälle nähty välitöntä tarvetta. Asetusten hallinta ja moduulien kytkeminen päälle tai pois tehdään muokkaamalla alustustiedostoa. Estolistoja ylläpidetään suorittamalla SQL-lausekkeita tarpeen mukaan.

4.1 Rakenne

Sovelluksen rakenteen suunnittelussa haluttiin ottaa huomioon erityisesti sovelluksen helppo laajentaminen myöhemmin esimerkiksi moduuleita lisäämällä sekä samanaikaisen käytön mahdollistaminen useammalla eri sivulla mahdollisimman vaivattomasti. Joustavuus ja laajennettavuus tuli myös huomioida tietokantaa suunnitellessa.

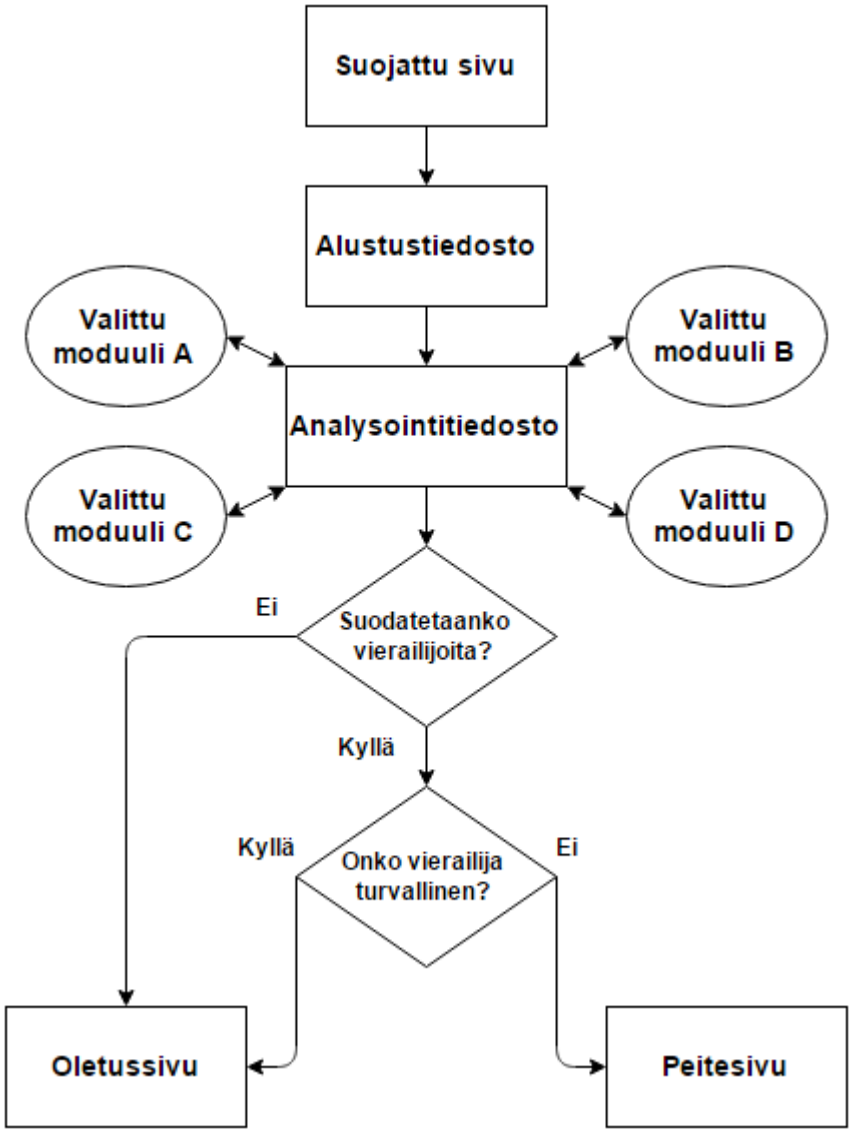
4.1.1 Sovelluksen rakenne

Ratkaisuna päädyttiin jakamaan sovellus kahtia luomalla varsinaisesta analysointitiedostosta erillinen alustustiedosto, joka sisältää kaikki sivukohtaiset asetukset. Alustustiedostossa myös määritellään halutut tarkistusmoduulit, joita analysointitiedosto käyttää klikkauksen aitouden selvittämiseen. Alustuskoodi voidaan sisällyttää niin moneen verkkosivuun kuin on tarpeen, ja käyttää esimerkiksi vain yhtä analysointitiedostoa WWW-palvelimen juuressa, jota kaikki sivustot kutsuvat yhdessä erilaisine asetuksineen.

Alustustiedosto lähettää klikkauksesta saamansa datan ja käyttäjän määrittelemät asetukset edelleen analysointitiedostolle, joka ajaa saadun datan alustustiedostossa valittujen tarkistusmoduuleiden läpi, tallentaa tuloksen tietokantaan ja palauttaa alustustiedostolle vastauksen, oliko klikkaus aito vai ei.

Jokaisen kampanjan ollessa erilainen, on tärkeää, että sovellusta pystytään aluksi käyttämään yksinkertaisesti pelkästään klikkausdatan tallentamiseen ilman, että suodatetaan klikkauksia lainkaan, jotta voidaan saada tietoa kävijöistä estämättä kuitenkaan aitoja klikkauksia vahingossa. Tällä tavalla voidaan ennen suodatuksen käynnistämistä kerätä kampanjaan saapuvista kävijöistä tarpeeksi tietoa oikeanlaisten suodatusasetusten määrittämiseksi.

Kun tarpeeksi klikkausdataa kävijöistä on kertynyt, voidaan päivittää tarvittavat asetukset ja halutut estomoduulit alustustiedostoon ja käynnistää suodatustoiminta.



Kuva 7. Sovelluksen rakenne ja toiminta yksinkertaistettuna

4.1.2 Tietokannan rakenne

Tietokannasta suunniteltiin mahdollisimman yksinkertainen. Kaikki, mitä sovellus tarvitsee, on jaettu kolmeen tauluun: ”daily_data”, ”ip_blacklist” ja ”org_blacklist”.

Nimensä mukaisesti ”daily_data” -taulu sisältää kaiken päivittäisen datan, mitä sovellus kerää klikkauksista. Jokainen klikkaus sisältää paljon dataa, joka tallennetaan yksityiskohtaisesti tietokantaan sopivan kolumnin alle.

```
mysql> SELECT COLUMN_NAME FROM information_schema.columns WHERE
table_name = 'daily_data';
+-------------+
| COLUMN_NAME |
+-------------+
| ip          |
| browser     |
| city        |
| state       |
| country     |
| org         |
| b_tz        |
| ip_tz       |
| tzoffset    |
| time_s      |
| proxy_hdrs  |
| blocked     |
| b_reason    |
+-------------+
13 rows in set (0.00 sec)
```

Kolumnien nimet antavat hieman viitettä siitä, mitä dataa mistäkin löytää:

- ”ip”-kolumni sisältää klikkaajan IP-osoitteen.
- ”browser”-kolumni sisältää selaimen nimen.
- ”city”-kolumni sisältää kaupungin nimen.
- ”state”-kolumni sisältää osavaltion joissain tapauksissa.

- ”country”-kolumni sisältää maatunnisteen.
- ”org”-kolumni sisältää ISP:n tai organisaation nimen.
- ”b_tz”-kolumni sisältää selaimen aikavyöhykkeen.
- ”ip_tz”-kolumni sisältää IP-osoitteesta lasketun aikavyöhykkeen.
- ”tzoffset”-kolumni sisältää selaimen ja IP-osoitteen aikavyöhykkeiden erotus.
- ”time_s”-kolumni sisältää klikkauksen ajankohdan.
- ”proxy_hdrs”-kolumni sisältää välityspalvelimen mahdollisesti lähettämät tunnisteet.
- ”blocked”-kolumni kertoo estettiinkö klikkaus vai ei.
- ”b_reason”-kolumni sisältää kaikkien niiden moduleiden nimet jotka merkkasivat klikkauksen haitalliseksi.

Taulu ”ip_blacklist” koostuu vain kahdesta kolumnista:

```
mysql> SELECT COLUMN_NAME FROM information_schema.columns WHERE
table_name = 'ip_blacklist';
+-------------+
| COLUMN_NAME |
+-------------+
| startip     |
| endip       |
+-------------+
2 rows in set (0.00 sec)
```

IP-estolistamoduulia varten tehdyssä taulussa on kaksi kolumnia, jotta voidaan estää erilaisia IP-alueita. Kun halutaan estää IP-alue, ”startip” -kolumniin laitetaan estettävän IP-alueen pienempi IP-osoite, esimerkiksi 127.0.0.0, ja ”endip” -kolumniin estettävän alueen suurempi IP, esimerkiksi 127.0.0.255. Tässä esimerkissä kaikki IP-osoitteet väliltä 127.0.0.0 - 127.0.0.255 tulisivat estetyiksi moduulin toimesta.

”org_blacklist” -taulu sisältää kaikkien niiden organisaatioiden ja palveluntarjoajien nimet, jotka halutaan estää.

```
mysql> SELECT COLUMN_NAME FROM information_schema.columns WHERE
table_name = 'org_blacklist';
+--------------+
| COLUMN_NAME  |
+--------------+
| organization |
+--------------+
1 row in set (0.00 sec)
```

Taulussa on vain yksi kolumni, johon estettävän organisaation tai palveluntarjoajan nimi syötetään estämistä varten.

4.2 Tarkistusmoduulit

Tarkistusmoduulit ovat sovelluksen tärkeimmät komponentit; moduulien vastuulla on selvittää, oliko varsinainen klikkaus aito vai ei. Koska aitojen klikkauksien tunnistaminen haitallisista klikkauksista kuten automaattisten bottien klikkauksista voi olla hyvinkin vaikeaa, on tärkeää käyttää useampaa tarkistusmoduulia yhdessä, jotta voidaan tunnistaa kaikki mahdolliset haitallisen liikenteen merkit.

Jokainen tarkistusmoduuli palauttaa aina vastauksena joko ”tosi” tai ”epätosi” riippuen saadusta tuloksesta. Sovellus tarkistaa kaikkien valittujen moduuleiden palauttamat tulokset ja yhdenkin moduulin palauttama ”epätosi” tarkoittaa, että klikkauksesta löytyi jotain moduulien sääntöjen vastaista, milloin sovellus määrittelee klikkauksen haitalliseksi.

Moduuleita haluttiin mahdollisimman monipuolisesti, jotta tunnistamisprosessista saataisiin mahdollisimman tarkka ja myös jotta sovellusta voitaisiin mukauttaa jokaiselle kampanjalle parhaiten.

4.2.1 Sallitut maat

Kampanjoiden ollessa usein maakohtaisia, milloin esimerkiksi tiettyä tuotetta tai palvelua halutaan myydä ja markkinoida vain Suomessa asuville, tarvitaan moduuli,

jonka avulla voidaan erotella halutusta maasta saapuneet klikkaajat, ja ne, jotka tulevat jostain muualta.

Klikkaajan maan selvittämiseen tarvitaan palvelu, joka sisältää IP-osoitteiden sijaintitietoja. Yksi hyvä tämmöinen palvelu on projektiin valittu MaxMind:in tarjoama GeoIP2 -tietokanta, joka tarjoaa monipuolisesti tietoa IP-osoitteista. Varsinaisen IP-osoitteen selvittämiseksi riittää PHP:hen sisäänrakennettu muuttuja.

Klikkaajan IP-osoite selvitetään lukemalla PHP:n sisäänrakennetun $_SERVER[’REMOTE_ADDR’] -muuttujan arvo, minkä jälkeen muuttujasta saatu IP tarkistetaan MaxMindin tarjoamaa tietokantaa vasten. Kun IP:n maa on selvitetty, tarkistetaan, löytyykö maata alustustiedostossa määritettyjen sallittujen maiden listalta, jonka perusteella Moduuli sitten tekee päätöksen.

GeoIP2 -tietokanta kattaa 99.9999 % käytetystä IP-avaruudesta. [6.] Kuitenkin varmuuden vuoksi tapauksissa, joissa tietokannasta saatu vastaus ei kerro klikkaajan maata, moduuli on ohjelmoitu olettamaan klikkauksen tulevan estetystä maasta.

4.2.2 HTTP header -kentät

Internet-selaimen ladatessa web-palvelimelta sivua selain lähettää HTTP-tunnisterivin (header field), joka sisältää tietoa selaimesta sekä selaimen käyttöympäristöstä. Tunnisterivin avulla voidaan esimerkiksi pyytää palvelinta käyttämään tietyn tyyppistä pakkausmenetelmää dataa lähettäessä. [7.]

Moduulissa kiinnitetään huomiota erityisesti kolmeen header-kenttään: User-Agentiin, Viaan ja X-Forwarded-Foriin.

Näistä kentistä User-Agent kertoo web-palvelimelle, minkälainen selain käyttäjällä on, jotta sisältöä voidaan tarpeen mukaan mukauttaa selaintyypille. Esimerkkejä erilaisista User-Agent -tunnisteriveistä:

Googlen robotti, joka indeksoi Internet-sivuja Googlen hakukonetta varten

```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

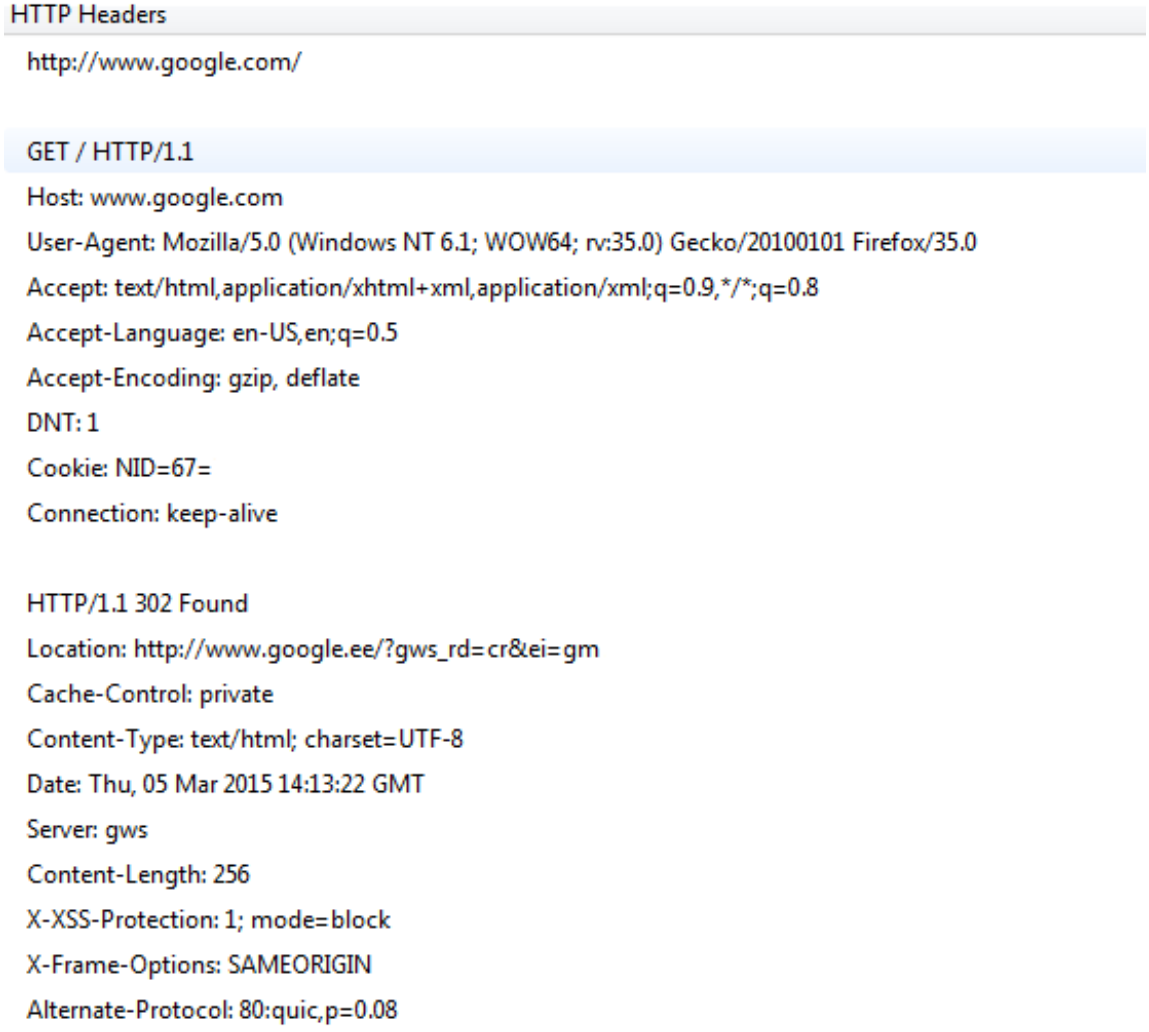
Yleinen tiedonsiirto-ohjelma Wget

```
Wget/1.9.1
```

Microsoftin Internet Explorer 11

```
Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
```

Käyttämällä esimerkiksi ”Live HTTP Headers” -lisäosaa Firefox-selaimessa voidaan nähdä selaimen lähettämät tunnisterivit kohdesivulle:



Kuva 8. Firefox-selaimen lähettämät HTTP-tunnisteet pyydettäessä google.comia ilman välityspalvelinta

Internetissä liikkuu paljon erilaisia robotteja, joiden tehtävä on käydä nettisivuja läpi esimerkiksi Googlen hakutuloksia varten. Robotteja varten on olemassa robottien rajausstandardi, eli Internet-palvelimen juurihakemistoon sijoitettava tiedosto, jolla voidaan määrittää, mitä sivuston osia mikäkin robotti saa tai ei saa tutkia. Valitettavasti

moni robotti ei huomioi rajaustiedostoja lainkaan, minkä takia erillisen moduulin tekeminen nähtiin aiheelliseksi. [8.]

Kuten aikaisempaankin moduuliin, myös User-Agentin lukemiseen löytyy PHP:n sisäänrakennettu muuttuja $_SERVER['HTTP_USER_AGENT']. Muuttujan arvon luettua moduuli suorittaa säännöllisen lausekkeen (regular expression) haun tarkistaakseen, täsmääkö User-Agent mihinkään kiellettyyn tietokannassa olevaan User-Agenttiin.

Via- ja X-Forwarded-For -kentät ovat välityspalvelimien usein käyttämiä header-kenttiä, jotka voivat sisältää tietoa esimerkiksi alkuperäisestä IP-osoitteesta ja välityspalvelimesta. Pelkkä kenttien olemassaolo HTTP-tunnisterivissä tarkoittaa yleisesti välityspalvelimen olevan käytössä, joten moduuli ainoastaan tarkistaa, lähetettiinkö kyseisiä kenttiä header-kenttien mukana eikä puutu erityisemmin kenttien mahdollisesti sisältämään dataan. [9.]

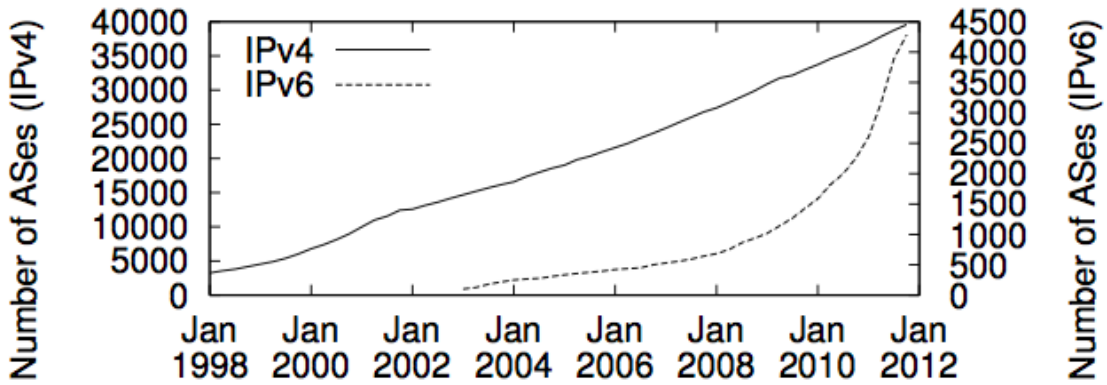
On tärkeää tietää, että web-palvelimelle lähetetyn header-datan muokkaaminen on kohtuullisen yksinkertaista, eikä pelkästään sen perusteella voida päätellä varmasti User-Agenttia tai välityspalvelimen läsnäoloa. Myöskään kaikki välityspalvelimet eivät lähetä Via- ja X-Forwarded-For -kenttiä käyttäjän anonyymiteetin suojaamiseksi.

4.2.3 IP-estolista

Jokaisella Internetiin kytketyllä laitteella on verkossa yksilöivä numerosarja, eli IP-osoite (”Internetin protokollaosoite”). IP-osoitteen avulla laitteen on mahdollista lähettää ja vastaanottaa tietoa toisen IP-osoitteellisen laitteen kanssa.

Työn tekohetkellä Internetin yleisimmin käytössä oleva IP-protokollan versio on neljäs, eli IPv4. IPv4-osoitteet koostuvat 32 bitistä, jotka on jaettu neljään 8 bitin sarjaan, eli oktetteihin. Helpomman käytön vuoksi IP-osoitteet esitetään yleisesti desimaaleina, joiden arvo on aina jotain välillä 0-255. IPv4:n ongelma nykyaikana on verkkoon kytkettyjen laitteiden määrä, sillä IP-osoitteita on nykyaikaan nähden hyvin rajallinen määrä (2^{32} = 4 294 967 296). Ratkaisu IP-osoitteiden rajallisuuteen on suosiotaan jatkuvasti kasvattava uusi IP-protokolla, joka tunnetaan paremmin nimellä IPv6. IPv6-osoiteavaruuden pituus on 128 bittiä verrattuna IPv4:n 32 bittiin, mikä mahdollistaa

siten todella paljon suuremman osoiteavaruuden (2^{128} = 340 282 366 920 938 463 463 374 607 431 768 211 456). [10.]



Kuva 9. IPv6 AS:ien (pääasiassa ISP-verkkojen) eksponentiaalinen kasvukäyrä kertoo IPv6:n nopeasta kasvusta [15.]

Koska IPv4 oli työn tekohetkellä suosituin käytössä oleva IP-protokolla, päätettiin tehdä estolista vain IPv4:n osoitteille. [11.]

Moduulina IP-estolista on hyvin yksinkertainen. Tietokannassa on taulu, joka sisältää ja johon lisätään estetyt IP-osoitteet. Moduuli tarkistaa klikkaajan IP-osoitteen taulua vastaan, ja jos IP-osoitteelle löytyy pari tai jos IP-osoite on estolistalla olevan IP-alueen sisällä, klikkaus estetään.

4.2.4 ISP ja Organisaatio -estolista

Tiettyjen ISP ja Organisaatioiden estämisen mahdollistava lista oli seuraava looginen lisäys moduuleihin. ISP:t eli Internet-palveluntarjoajat (Internet Service Providers) ovat kutsunimensä mukaisesti yrityksiä tai organisaatioita jotka tarjoavat asiakkailleen erilaisia Internet-palveluita, kuten Internet-liittymiä ja web-hotelleita.

Koska tiedetään, että oikeat klikkaajat ovat tavallisia käyttäjiä, jotka useimmiten käyttävät kuluttajille suunnattuja nettiliittymiä, halutaan tavallisesti estää esimerkiksi erilaisia Internet-palvelimia myyviä yrityksiä, sillä näitä saatetaan käyttää välitys- tai robottipalvelimina.

Moduuli hakee klikkaajan IP:n tiedot käyttäen MaxMindin tarjoamaa GeoIP2 ISP -tietokantaa, jonka jälkeen löydettyä ISP:tä tai organisaationimeä verrataan estolistaan, joka on yksinkertaisuudessaan tietokannassa oleva haitallisten ISP:n ja organisaatioiden nimiä sisältävä taulu.

Estolistaa päivitetään lisäämällä tai poistamalla tietokannan taulusta rivejä yksinkertaisilla SQL-lausekkeilla.

4.2.5 Project Honey Pot

Project Honey Pot on Internetin ensimmäinen hajautettu järjestelmä haitallisten robottien ja käyttäjien tunnistamiseksi. Kuka tahansa voi osallistua projektiin asentamalla hunajapurkin (”honeypot”) verkkosivuilleen, joka toimii luomalla tavalliselle käyttäjälle näkymättömiä ”ansoja”, joihin sitten robotit lankeavat ja jättävät jälkensä. [12.]

Hunajapurkkiin langenneiden IP-osoitteet ja selaintiedot lähetetään eteenpäin projektin palvelimelle, joka koostaa eri Internet-sivuilta saadun datan yhtenäiseksi, josta lasketaan uhka-arvio (”threat rating”) IP:lle, millä arvioidaan IP:n vaarallisuutta kokonaisuutena. Uhka-arvion arvo perustuu havaittujen sääntörikkomusten ja haitallisten tekojen yhteenlaskettuun määrään.

http:BL tai ”HTTP Blacklist” on Project Honey Pot:in luoma järjestelmä, joka tarjoaa Internet-sivujen ylläpitäjille mahdollisuuden käyttää projektin keräämää dataa epäilyttävistä ja haitallisista Internetin käyttäjistä hyväkseen. Moduuli suorittaa DNS-haun klikkaajan IP-osoitteelle käyttäen http:BL:n API:a joka lähettää vastauksena projektin keräämät tiedot IP:stä. [13.]

Esimerkiksi haku IP-osoitteelle 127.9.1.2 käyttämällä API-avainta nimellä ”avain” näyttäisi tältä:

Haku: `avain.2.1.9.127.dnsbl.httpbl.org`
Vastaus: `127.3.5.1`

Jossa hakupyynnön ensimmäinen osa ”avain” korvataan käytännössä henkilökohtaisella API-avaimella, seuraavana IP-osoitteen oktetit on käännetty

päinvastaiseen järjestykseen ja ”dnsbl.httpbl.org” määrittää listan, minkä avulla IP tarkistetaan. Jos IP-osoitetta ei löydy estolistalta, palvelu lähettää DNS-vastauksena ”NXDOMAIN” - DNS-osoitetta ei löydy. Jos osoite vastaa palvelussa jo olevaa IP:tä, on vastaus IP-osoitetta muistuttava numerosarja kuten tapauksen esimerkki ”127.3.5.1”. Tässä vastauksen ensimmäisen oktetin arvo ”127” tarkoittaa, että pyyntö onnistui, jokin muu arvo tarkoittaa pyynnössä olleen jonkinlainen ongelma.

Toinen oktetti tarkoittaa, kuinka monta päivää oli kulunut edellisestä havainnosta. Esimerkin tapauksessa viime havainnosta on kolme päivää.

Kolmas oktetti on Project Honey Pot:in antama uhka-arvio IP-osoitteelle, ja sen arvo voi olla mitä tahansa 0:n ja 255:n välillä. Alin arvo 0 tarkoittaa uhan olevan olematon ja 255 äärimmäisen uhkaava. Uhka-arvio on logaritmisen asteikon numero, jossa uhka-arvion arvo 25 vastaa suunnilleen 100 roskapostiviestin lähettämää IP:tä, arvo 50 IP:tä josta on lähetetty 10 000 roskapostiviestiä, arvo 75 IP:tä, josta on lähetetty 1 000 000 roskapostiviestiä ja niin edelleen.

Neljäs oktetti määrittelee klikkaajan tyypin neljästä eri vaihtoehdosta: 0 = hakukone, 1 = epäilyttävä, 2 = keräilijä ja 4 = kommentti-roskapostittaja. Neljännen oktetin arvolla voidaan myös viestiä klikkaajan täyttäneen useamman tyypin kriteerit. Esimerkiksi oktetin arvo 3 tarkottaa klikkaajan olevan epäilyttävä ja keräilijä, eli 1 + 2, arvo 5 epäilyttävä ja kommentti-roskapostittaja, arvo 7 epäilyttävä, keräilijä ja kommentti-roskapostittaja, ja niin edelleen. [14.]

4.2.6 Aikavyöhyketarkistus

Aikavyöhyketarkistuksessa verrataan klikkaajan IP-osoitteen perusteella saatua aikavyöhykettä klikkaajan tietokoneen kellon ajasta laskettuun aikavyöhykkeeseen. Jos aikavyöhykkeiden välillä on eroa esimerkiksi yli tunti (jotta sallitaan pieni virhemarginaali), voidaan pitää todennäköisenä, että klikkaaja käyttää välityspalvelinta.

Klikkaajan aikavyöhykkeen selvittämiseksi tarvittiin tekniikka, joka pystyisi lukemaan kellonajan klikkaajan selaimen kautta. Esimerkiksi PHP:n ollessa täysin palvelinpuolen kieli ja HTML:n rajoitteiden takia kummallakaan ei pysty suorittamaan käyttäjän selaimessa tarvitunlaisesti koodia, jolla saataisiin tarpeeksi tietoa aikavyöhykkeen selvittämiseksi.

JavaScriptin avulla klikkaajan aikavyöhykkeen noutaminen kuitenkin onnistuu käyttämällä getTimezoneOffset()-metodia. Moduuli hakee ensimmäiseksi klikkaajan aikavyöhykkeen IP-osoitteen perusteella käyttäen MaxMind:in GeoIP2 -tietokantaa. Tämän jälkeen selvitetään tietokoneen JavaScript-aikavyöhyke ja verrataan saatuja arvoja toisiinsa.

Koska kaikilla klikkaajilla ei välttämättä ole JavaScript-tukea selaimessa tai se on kytketty pois päältä esimerkiksi tietoturvasyistä, ja koska joissain harvinaisissa tapauksissa aikavyöhykkeitä ei onnistuta laskemaan geolokaatiotietokannan puutteellisten IP-tietojen takia, tarkistuksen epäonnistuessa mistä tahansa syystä moduuli on ohjelmoitu estämään klikkaus.

5 Sovelluksen käyttöönotto

Sovelluksen käyttäminen kampanjasivuilla vaatii asetusten määrittelemistä kampanjakohtaisesti sekä käytettävistä moduuleista riippuen enemmän tai vähemmän jatkuvaa ylläpitoa, jotta saavutetaan mahdollisimman tehokas suoja haitalliselta liikenteeltä ja vältytään vahingossa estämästä oikeiden kävijöiden pääsy sivustolle.

5.1 Klikkausdatan kerääminen

Ennen kuin klikkausten suodattaminen aloitetaan, halutaan kerätä tietoa sivuston vierailijoista vaikuttamatta käyttäjäkokemukseen vielä millään tavalla. Parhaiten dataa saadaan ottamalla kaikki tarkistusmoduulit käyttöön (jotta jokaisesta moduulista saadaan tietoja) ja kytkemällä suodatuksen väliaikaisesti pois päältä, milloin riippumatta moduulien palauttamasta vastauksesta vierailijat päätyvät aina oikealle sivulle, mutta tietokantaan tallentuu kuitenkin tiedot klikkauksesta ja moduuleista.

Kun kampanja on käynnissä ja sovellus on konfiguroitu toimimaan oikein, alkaa tietokantaan ilmestyä dataa. Dataa tulee yleensä runsaasti, sillä eri moduuleiden tulokset ja löydökset tallennetaan kaikki tietokantaan myöhempää tarkastelua varten.

Klikkausdataa on hyvä kerätä ja seurata erityisesti uuden kampanjan aloittamisen yhteydessä ainakin muutaman päivän verran ennen liikenteen suodattamisen

aloittamista, eikä klikkausdatan seuraamista kannata laiminlyödä ensimmäisten päivien jälkeenkään vaan seurata säännöllisesti sovelluksen toimintaa sekä saapuvaa dataa, jotta maksimoidaan suodattimen toiminta.

Kaikki kerätty klikkausdata saadaan helposti näkyviin suorittamalla SQL-lauseke:

```
mysql> SELECT * FROM daily_data;
```

Siinä ”daily_data” on taulu, johon kaikki klikkausdata tallennettaan sovelluksen toimesta. Koska klikkausdataa tulee usein paljon, on tärkeä osata navigoida tietokantaa tehokkaasti suorittamalla oikeanlaisia SQL-lausekkeita.

5.2 Klikkausliikenteen valvonta

Kun klikkausdataa alkaa kertymään, on sitä pyrittävä valvomaan jatkuvasti. Tehokkaassa valvonnassa olennaista on osata muodostaa oikeanlaisia SQL-lausekkeita, joiden avulla saadaan haluttu data näkyviin tarkastelua varten.

Hyvä esimerkkilauseke olisi esimerkiksi:

```
mysql> SELECT time_s, ip, org, b_reason FROM daily_data WHERE
b_reason NOT LIKE '';
```

Lauseke valitsee kaikki tietueet ”time_s”, ”ip”, ”org” ja ”b_reason” pöydästä ”daily_data”, joissa ”b_reason” -tietue ei ole tyhjä, eli toisin sanoen klikkaus on päätelty haitalliseksi jonkun moduulin toimesta. Tällä lausekkeella nähdään kellonaika, IP-osoite, klikkaajan organisaatio tai ISP sekä moduuli(t) jotka havaitsivat klikkauksessa jotain poikkeavaa.

Toinen käytännöllinen lauseke päivittäisen datan tarkasteluun on:

```
mysql> SELECT * FROM daily_data WHERE time_s LIKE "2014-12-24%";
```

Tämän lausekkeen SELECT valitsee kaikki tietueet pöydästä ”daily_data”, joiden päivämäärä on 24.12.2014, riippumatta kellonajasta. Lauseke on erittäin käytännöllinen päivittäisen datan yksityiskohtaiseen tarkasteluun, mistä johtuen sovelluksen käytännön testeissä lauseke on ehdottomasti ollut yleisimmin käytetty.

Erilaisia lausekkeita muodostamalla voidaan analysoida tallennettua dataa tehokkaasti, sekä tehdä järkeviä päätöksiä moduuleiden ja estolistojen suhteen.

5.3 Estolistojen ylläpito

Klikkausdatasta poimitut haitalliset IP-osoitteet tai -alueet sekä haitalliset organisaatiot tai ISP:t tulisi lisätä estolistoille aina kuin mahdollista sovelluksen suodatustehokkuuden parantamiseksi. Estolistojen tehokas hallinta on tärkeä osa niitä hyödyntävien moduuleiden toiminnassa, jonka takia yleisimmät komennot listojen muokkaamiseen on hyvä olla tiedossa.

5.3.1 ISP- ja Organisaatio -estolista

ISP- ja Organisaatio -estolista on yksinkertainen ylläpitää, sillä se koostuu ainoastaan yhdestä tietueesta, jossa on estettävän kohteen nimi. Täten SQL-lauseke, jolla lisätään uusi kohde listalle, on yksinkertainen:

```
mysql> INSERT INTO org_blacklist VALUES ('Esimerkki');
```

Eli lauseke lisää tunnisteen ”Esimerkki” estettävien listalle. Jos sen sijaan halutaan muokata olemassaolevaa tunnistetta, voidaan käyttää UPDATE-lausetta:

```
mysql> UPDATE org_blacklist SET organization = 'Tuhma' WHERE
organization = 'Kiltti';
```

Joka tässä esimerkissä muuttaisi ”Kiltti” -nimisen kohteen kohteeksi nimeltä ”Tuhma”.

Jos halutaan poistaa estolistalta nimi kokonaan, DELETE-lause toimii tarkoitukseen hyvin:

```
mysql> DELETE FROM org_blacklist WHERE organization = 'Tuhma';
```

5.3.2 IP-estolista

IP-estolista on vain hieman ISP- ja Organisaatio -estolistaa monimutkaisempi johtuen sen sisältämistä kahdesta eri tietueesta. IP-estolistan tapauksessa voidaan käyttää samaa SQL-lauseketta kuin ISP- ja Organisaatio -estolistaa päivittäessä, kunhan muistetaan antaa molemmille tietueille arvot:

```
mysql> INSERT INTO ip_blacklist VALUES ('127.0.0.3','127.0.0.3')
```

Esimerkissä asetetaan estolistalle vain yksi IP-osoite, 127.0.0.3. Jos haluttaisiin estää esimerkiksi kaikki IP-osoitteet väliltä 127.0.0.3 - 127.0.0.6, muokattaisiin lausekkeessa toisen IP-osoitteen viimeistä oktettia esimerkin osoittamalla tavalla:

```
mysql> INSERT INTO ip_blacklist VALUES ('127.0.0.3','127.0.0.6')
```

Näin IP:t 127.0.0.3 ja 127.0.0.6 sekä kaikki niiden väliltä (tässä tapauksessa IP:t 127.0.0.4 ja 127.0.0.5) tulisivat estetyksi.

6 Yhteenveto

Insinöörityön tuloksena syntyi tehokas ja monipuolinen sovellus, jolla asiakas pystyy pitämään kampanjamateriaalinsa paremmin suojassa haitallisilta vierailijoilta. Sovellus on ollut asiakkaan käytössä kesäkuusta 2014 lähtien hyvällä menestyksellä, eikä sovelluksen käytössä ole esiintynyt ongelmia. Asiakas on myös itse lisännyt sovellukseen joitain pienempiä ominaisuuksia, joista on ollut edelleen lisähyötyä sovelluksen käytössä.

Sovellus kirjoitettiin lähes kokonaan käyttämällä Sublime Text 2 -tekstieditoria Windows-käyttöjärjestelmässä, sekä testattiin Debian Linux:issa käyttämällä Nginx-webpalvelinta ja PHP-FPM -tulkkia ja vielä Ubuntu Linux:issa Apache2 ja mod_php:n kanssa. Sublime Text -tekstieditori valikoitui ystävän suosituksen kautta työhön, ja se soveltui suoraan palvelimelle ohjelmointiin loistavasti. Tulen varmasti käyttämään sitä jatkossakin vastaavanlaisissa töissä. Koodin siirtäminen palvelimelle Sublime Textistä sujui kätevästi käyttämällä WinSCP:tä tiedostojen siirtoon ja asettamalla Sublime Text WinSCP:n vakiotekstieditoriksi. Kun tällä tavalla tallensi sovelluksen tiedostoon muutoksen, WinSCP kopioi sen välittömästi palvelimelle tehden koodin suorittamisesta ja virheenetsinnästä huomattavasti nopeampaa ja helpompaa.

Sovellusta kehittäessä tuli huomattua erityisesti, kuinka tärkeää oli projektin huolellinen suunnittelu ja kuinka isoja hyötyjä työmäärän suhteen suunnittelusta saatiin, kun koodia ei tarvinnut kirjoittaa uudelleen erilaisten aivoituksien ansiosta kesken toteuttamisvaihetta. Erityisesti modulaarinen lähestymistapa sovelluksen estosääntöjen toteuttamisessa oli tärkeää lopputuloksen ja työmäärän kannalta, ja modularisuudesta tulee varmasti olemaan hyötyä myös erityisesti sovelluksen jatkokehityksessä.

Jatkokehitettävää sovelluksessa riittää. Esimerkiksi graafisen käyttöliittymän tekeminen sovelluksen hallintaan voisi olla merkittävä parannus, joka nopeuttaisi ja helpottaisi hallintaa huomattavasti. Uusien moduuleiden kehitys on lähes poikkeuksetta vain hyväksi, kunhan vain keksii lisää keinoja haitallisen liikenteen havaitsemiseksi. Joitain jo olemassa olevia moduuleitakin voisi kehittää eteenpäin. Esimerkiksi aikavyöhykkeen laskemisessa voitaisiin käyttää useampaa tekniikkaa kuten vaikka Flash:ia selaimen aikavyöhykkeen tarkistamiseksi ja IP-estolistassa voitaisiin hyödyntää esimerkiksi Spamhaus:in tai SORBS:in IP-estolistoja paremman kattavuuden saavuttamiseksi.

Sovelluksesta voisi myös tehdä automaattisesti ”oppivan”, jolloin se itsenäisesti suorittaisi jonkun näköistä päättelyprosessia tai heuristiikkaa haitallisten klikkausten havaitsemiseksi ja torjumiseksi sekä päivittäisi estolistoja esimerkiksi jonkinlaisen uhka-arvion perusteella.

Lähteet

1 Sir Tim Berners-Lee. 2012. Verkkodokumentti. World Wide Web Foundation. < http://webfoundation.org/about/sir-tim-berners-lee/ >. Luettu 12.10.2014.

2 History of PHP. 2014. Verkkodokumentti. The PHP Group. < http://php.net/manual/en/history.php.php >. Luettu 14.10.2014.

3 MySQL - suomalainen menestystarina. 2008. Verkkodokumentti. It-viikko. < http://www.itviikko.fi/ratkaisut/2008/01/16/mysql--suomalainen-menestystarina/20081483/7 >. Luettu 14.10.2014.

4 Browser History: Netscape. 2005. Verkkodokumentti. Blooberry / Brian Wilson. < http://www.blooberry.com/indexdot/history/netscape.htm >. Luettu 14.10.2014.

5 How is JavaScript different from Java? 2014. Verkkodokumentti. Oracle Corporation. < https://www.java.com/en/download/faq/java_javascript.xml >. Luettu 19.10.2014.

6 GeoIP2: Industry Leading IP Intelligence. 2014. Verkkodokumentti. MaxMind, Inc. <https://www.maxmind.com/en/geoip2-services-and-databases>. Luettu 2.11.2014.

7 Grigorik, Ilya. 2013. High Performance Browser Networking. O'Reilly Media, Inc.

8 The Web Robots Pages. 2014. Verkkodokumentti. robotstxt.org. <http://www.robotstxt.org/robotstxt.html>. Luettu 19.11.2014.

9 Forwarded HTTP Extension. 2014. Verkkodokumentti. Internet Engineering Task Force. <http://tools.ietf.org/html/rfc7239>. Luettu 19.11.2014.

10 Understanding IP Addressing. 2014. Verkkodokumentti. RIPE Network Coordination Centre. <https://www.ripe.net/internet-coordination/press-centre/understanding-ip-addressing>. Luettu 20.11.2014.

11 Dataset Comparison: IPv4 vs IPv6 traffic seen at the DNS Root Servers. 2014. Verkkodokumentti. Center for Applied Internet Data Analysis. <http://www.caida.org/research/policy/dns-country/>. Luettu 22.11.2014.

12 Project Honey Pot: Frequently Asked Questions. 2014. Verkkodokumentti. Unspam Technologies, Inc. < https://www.projecthoneypot.org/faq.php>. Luettu 5.12.2014.

13 Project Honey Pot: Http:BL Overview. 2014. Verkkodokumentti. Unspam Technologies, Inc. <https://www.projecthoneypot.org/httpbl.php>. Luettu 5.12.2014.

14 Project Honey Pot: Http:BL Application Programming Interface (API). 2014. Verkkodokumentti. Unspam Technologies, Inc. <https://www.projecthoneypot.org/httpbl_api.php>. Luettu 8.12.2014.

15 Measuring the Deployment of IPv6: Topology, Routing and Performance. 2012. Verkkodokumentti. Amogh Dhamdhere, Matthew Luckie, Bradley Huffaker. <http://www.caida.org/publications/papers/2012/measuring_deployment_ipv6/measuring_deployment_ipv6.pdf>. Luettu 22.12.2014.