Teemu Kantoluoto

Ohjelmiston päivityspakettien jakelujärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

7.11.2012

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Teemu Kantoluoto Ohjelmiston päivityspakettien jakelujärjestelmä 43 sivua 7.11.2012
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	lehtori Ilpo Kuivanen toimitusjohtaja Juha Lappi

Työn aihe oli kokonaisvaltaisen päivitysjärjestelmän rakentaminen Deltagon Group Oy:n sähköpostin salausjärjestelmälle. Työssä tutkittiin käytössä olevia päivitysjärjestelmiä ja niiden käyttömahdollisuutta kohdejärjestelmässä ja käytiin huolellisesti läpi toteutus, johon päädyttiin.

Työssä suunniteltiin päivitettävän aineiston pakkaaminen muotoon, jolla se voidaan siirtää ja helposti asentaa kohdepalvelimelle. Lisäksi suunniteltiin pakettien jakelupalvelimen toteutus ja siihen liittyvät työkalut sekä asiakaspalvelinten asennusprosessi ja työkalut.

Tarkoituksena oli yksinkertaistaa Deltagon Group Oy:n päivitysprosessia, joka yrityksen mukaan vie liikaa aikaa ja on altis virheille. Päivitystapahtuma pyrittiin saamaan mahdollisimman standardisoiduksi ja automaattiseksi.

Avainsanat	ohjelmistopäivitykset, paketinhallinta, perl, rpm

Author(s) Title Number of Pages Date	Teemu Kantoluoto Software Update Package Distribution System 43 pages 7 November 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Ilpo Kuivanen, Senior Lecturer Juha Lappi, Chief Executive Officer

The topic of this study was to produce a wholesome software update system for Deltagon Group email security server software. Some existing update systems and their applicability to the system were investigated. The selected method was then delved into and completed.

The study consists of designing how the updated software is to be packaged, distributed and installed on customer servers. This includes describing the process in each step and exploring and setting up the necessary tools.

The primary purpose of the study was to streamline the updating process of Deltagon Group, which up to this point was seen as exceedingly time consuming and error prone. Standardizing and automating as much of the updating process as possible was considered very important.

Keywords	software updates, package managing, perl, rpm

Sisällys

Lyhenteet

1 Johdanto 1

2 Työssä käytettyjä ohjelmistoja 1

2.1 Perl 1

2.2 MySQL 3

2.3 RPM Package Manager 4

2.4 Yellow Dog Updater, Modified (ohjelman virallinen nimi, komm. huom.) 5

3 Päivitysjärjestelmistä 6

3.1 Yleistä 6

3.2 Toteutustapoja 7

3.3 Päivitysjärjestelmien tietoturva 8

3.4 Tietoturvariskejä 9

3.4.1 Injektiohyökkäys 9

3.4.2 IP spoofing 10

3.4.3 Palvelunestohyökkäys - Denial of Service (DoS) 10

3.4.4 Cross-site Scripting (XSS) 10

3.4.5 Palvelimena esiintyminen 11

4 Deltagon Group Oy 11

5 Taustaa 12

5.1 Tilanne ennen järjestelmää 12

5.2 Päivitysjärjestelmän suunnittelu- ja päätösprosessi 13

6 Päivityspaketin luonti ja testaus 14

6.1 Yleistä 14

6.2 Luonti 14

6.2.1 Yleistä spec-tiedostosta 14
6.2.2 Deltagon D3 spec -tiedosto 16

6.3 Testaus 20

7 Päivityspalvelin 21

7.1 Yleistä 21

7.2 Päivityspalvelimen suojaus 22

7.3 Tekninen toteutus 23
7.3.1 Palvelimen käyttöjärjestelmän asennus 23
7.3.2 Yum-repositorion, apachen ja palomuurin konfigurointi 24
7.3.3 Päivitysraporttien vastaanottopalvelu 26
7.3.4 Järjestelmän tietokanta 26
7.3.5 Järjestelmän asetusten ylläpito 29
7.3.6 Järjestelmän web-hallinnointityökalu 30
7.3.7 Päivityspalvelimen ja kokoamispalvelimen paikka Deltagon Group Oy:n verkossa 31

8 Asiakaskone 32

8.1 Yleistä 32

8.2 Yum-ohjelman asetukset 32

8.3 Päivitysjärjestelmän asetustiedosto 34

8.4 Päivitysjärjestelmän kirjasto 35
8.4.1 Kirjaston perustoiminnot 35
8.4.2 IPC::Run-moduuli 37
8.4.3 Kirjaston käyttö 38
8.4.4 Muiden solujen päivitys 38

8.5 Päivitysraportit 39

8.6 Paketin asennuskäyttöliittymä 39

8.7 Automaattinen päivitys 42

9 Lopputulokset 42

Lähteet 44

Lyhenteet

CPAN	Comprehensive Perl Archive Network. Julkinen Perl -ohjelmistomoduuliarkisto.
IPC	Inter-process Communication. Ohjelmaprosessien välinen kommunikointi.
rpm	RPM Package Manager. Paketinhallintatyökalu.
yum	Yellow Dog Updater, modified. Rpm-paketteja käsittelevä paketinhallinta-ohjelmisto.

1 Johdanto

Tässä työssä pyrittiin ratkaisemaan Deltagon Group Oy:n lippulaivatuotteen päivittäminen yhtiön yritysasiakkaiden palvelimille tehokkaasti ja turvallisesti. Työssä tutkittiin erilaisia toteutustapoja ohjelmiston päivitykseen ja päädyttiin lopulta ratkaisuun, jossa yhdistettiin käyttöjärjestelmän päivitykseen käytettyjä ohjelmistoja omiin erikoistuneisiin työkaluihin, jotka rakennettiin tuotteen hallinnointityökaluun.

Aluksi tutustutaan erilaisiin päivitysjärjestelmäratkaisuihin ja tietoturvariskeihin, joita ratkaisussa tuli ottaa huomioon. Työn ratkaisun tarvitsema ohjelmiston paketointitapa suunnitellaan ennen tarvittavien ohjelmisto- ja palvelinratkaisujen vaiheita. Ratkaisun toteuttamista varten tarvittiin monia yleisesti käytössä olevia ohjelmistoja tukemaan palvelun tarjontaa ja tietosuojaa, ja niiden asetukset käydään läpi omissa vaiheissaan. Työn ohjelmisto- ja palvelinratkaisut kuvataan myös tarkasti omissa osioissaan.

2 Työssä käytettyjä ohjelmistoja

2.1 Perl

Perl (Practical Extraction and Report Language) on proseduraalinen skriptikieli, joka on alun perin suunniteltu tekstitiedostojen lukemiseen ja käsittelyyn. Kielen ensimmäinen ilmentymä julkaistiin vuonna 1987 ja siitä on sittemmin julkaistu yhteensä neljä uutta versiota. Uusin vakaa versio on 5.16.2 [1] ja työn alla on myös kokonaan uusi versio (Perl 6), joka perustuu tämän hetkiseen versioon (Perl 5) mutta muutokset ovat niin perustavaa laatua että alaspäin yhteensopivuus on mahdotonta. Perl 6, toisin kuin edeltäjänsä, on täysin oliomallinen kieli. Sen kehitys on kuitenkin vielä niin aikaisessa vaiheessa, ettei siitä ole täysin toimivaa versiota. Perl 5 on yleisesti käytössä oleva versio. Versio 5.8.8 oli pitkään julkisesti käytössä ollut vakaa versio, mutta 5.10.1 on korvannut sen Linux-jakelujen oletusversiona.

Ohjelmointikielenä Perl on hyvin löysä sääntöjensä ja syntaksinsa kanssa. Skriptikielenä se ei myöskään yllä suoritusnopeuksiltaan C-toteutuksien tasolle, mutta se ei myöskään vaadi kehittäjiltä samaa pilkuntarkkuutta esimerkiksi muistin hallinnan kanssa. Perliin on kuitenkin saatavilla niin sanotut ‘strict’- ja ‘warnings’-moodit, joissa koodia

suorittaessa tarkistetaan ennalta määritettyjä kriteerejä kuten muuttujien käyttö ja julistus oikeissa skoopeissa ja suoritus keskeytetään, mikäli näihin kriteereihin ei ylletä.

Perl on tunnettu sen CPAN (Comprehensive Perl Archive Network)-verkostostaan, jonne kehittäjät voivat julkaista tuottamiaan moduuleita. Vuosien saatossa sinne on kerääntynyt kattava määrä sekä yleiskäytännöllisiä että hyvin erikoistuneita moduuleita, joita voidaan Perl-lisenssillä käyttää omissa toteutuksissa. Joitain käytännölliseksi todettuja moduuleja on kytketty ydinpaketin yhteyteen, mutta oletuksena ohjelmistototeutuksen tarvitsemat moduulit haetaan ja päivitetään itsenäisesti.

Vaikka Perl 5 ei ole olio-ohjelmointikieli, CPAN-moduulien muoto muistuttaa olioita. Niitä kutsutaankin pseudo-olioiksi. Sillä olio-ohjelmointi on todettu hyväksi tavaksi mallintaa haluttuja toiminnallisuuksia, sen muotoa käytetään hyödyksi näillä pseudo-olioilla.

Perlissä on kolmen tyyppisiä muuttujia: skalaareja, taulukoita ja hajautustauluja. Skalaarimuuttujiin viitataan dollarimerkillä muuttujan alussa, taulukoihin at-merkillä (@) ja hajautustauluihin prosenttimerkillä. Yksi erittäin tärkeä muuttujien käsittelymuoto on referenssi, joka on skalaarimuuttuja ja joka osoittaa perusmuuttujan muistialueeseen. Tavallisesti kun funktiolle välitetään muuttuja, siitä luodaan kopio. Kun funktiolle välitetään referenssi, kopioidaankin muuttuja, joka osoittaa alkuperäisen muuttujan muistialueeseen. Näin funktioiden välillä voidaan viitata täysin samaan muuttujaan. Referenssien alkuperäiseen muuttujaan osoitetaan dollarimerkillä muuttujan nimen ja etumerkin välillä. Esimerkiksi skalaarireferenssin alkuperäiseen muuttujaan viitataan kahdella dollarimerkillä. Taulukkomuotoisien muuttujien alkioihin viitataan tavallisesti hakatai väkäsulkeilla heti muuttujan jälkeen. Referenssitaulukon alkioon osoitetaan laittamalla referenssimuuttujan ja sulkeismerkin väliin oikealle osoittava nuoli. Esimerkiksi taulukon *@many* alkioon 3 viitataan tavallisesti *$many[2]*, ja referenssiin *$many->[2]*.

Muuttujia voi olla myös useissa eri skoopeissa, joista tärkeimmät ovat nimiavaruuden ja paikallisen koodialueen skoopit. Nimiavaruuden skooppiin julistetaan muuttuja ò our ò etuliitteellä, esimerkiksi ò our $date = ð11.11.2012ñò Paikalliseen skooppiin muuttuja julistetaan ò my ò etuliitteellä vastaavasti. Paikallinen muuttuja näkyy myös kaikkiin eroteltuihin alueisiin, jotka on määritelty sen alueen sisällä. Esimerkiksi sisäkkäisissä ehtolauseissa ensimmäisessä alueessa määritelty muuttuja tunnetaan kaikissa sitä seuraavissa ehtolauseissa ja niiden alueissa.

Perlin moduulien tekemiseen on määritelty *best practices* -muotoilusuositukset, joita tässäkin työssä käytetään. Moduuli on käytännössä nimiavaruus, joka julistetaan package <nimi>; rivillä. Näin luodun nimiavaruuden funktioihin voidaan viitata kutsumalla ydinfunktiota *bless* pseudo-olion tietorakenteena toimivalle hajautustaulureferenssille. *Bless* tuottaa referenssin nimiavaruuteen, jolloin sen funktioita voidaan kutsua hajautustaulureferenssimuuttujaan viitaten. Hajautustaulu toimii silti myös tavallisena hajautustauluna, johon voidaan lisätä avain-arvo pareja, joita voidaan verrata olion instanssin muuttujiin. On suositeltavaa, että nimiavaruus tarjoaa funktion joka palauttaa *bless*atun muuttujan. Tämän funktion suositellaan olevan nimeltään *new,* mutta ehdoton vaatimus se ei ole.

Todellisuudessa hajautustaulureferenssin kautta kutsutut funktiot muutetaan nimiavaruuden funktioiden kutsuiksi siten, että niiden ensimmäiseksi parametriksi välitetään hajautustaulu itse. Siksi funktioita kirjoittaessa ensimmäinen käsiteltävä parametri on usein *$self,* eli hajautustaulureferenssi.

```
my $object = Namespace->new();
1) $object->function_call($variable);
2) Namespace->function_call($object, $variable);
```

Edellä olevan esimerkin kohdat yksi ja kaksi ovat teknisesti identtiset.

2.2 MySQL

MySQL on vapaan lähdekoodin relaatiotietokantaohjelmisto. Markkinoilla olevista relaatiotietokantaohjelmistoista se on käytetyin vapaan koodin vaihtoehto [2]. Ohjelmiston kehitys alkoi 1990-luvulla, ja sitä on jatkettu tähänkin päivään asti. Nykyisin tuotemerkin omistaa ohjelmistoyritys Oracle, joka on tunnettu muun muassa Java-ohjelmointikielestä. MySQL on suosittu varsinkin pienten, itsenäisten joukkojen ja yritysten tietokantavalintana avoimen saatavuutensa vuoksi. Erittäin suuri osa kaikista web-hakukoneista on toteutettu MySQL:llä.

MySQL-tietokannat sisältävät tauluja, jotka koostuvat yhdestä tai useammasta tietueesta, joiden tyypit määritellään luomisen yhteydessä. Tietokantaan syötettäessä dataa, annetun syötteen tulee olla kuhunkin tietueeseen sopivia syötteitä. Esimerkiksi kokonaislukutietue hylkää syöttötapahtuman, mikäli siihen yritetään syöttää merkkijono.

Tietokantojen ja taulujen käyttäjäoikeuksia pidetään yllä pääkäyttäjän skeematietokannan tauluissa. Näillä tiedoilla voidaan rajata käyttöoikeuksia kanta- tai taulukohtaisesti kullekin käyttäjälle. Esimerkiksi palvelinkoneessa, jossa toimii useita palveluita, jotka kaikki käyttävät MySQL-tietokantaa tietojensa tallentamiseen, käyttöoikeudet on rajattu siten, että kullekin palvelulle luodaan oma käyttäjänsä, jonka oikeudet rajataan vain palveluun liittyvien kantojen ja taulujen käyttöön. Joskus samalla palvelulla voi olla useita käyttäjiä eri toimintojen suoritukseen. Esimerkiksi tiedon lukuun voidaan käyttää käyttäjää, jolla on vain lukuoikeus kannan tauluihin.

Kun kantakomentoja suoritetaan siten, että ne sisältävät käyttäjäsyötettä, on mahdollista, että käyttäjäsyöte sisältää merkkejä, jotka sekoittavat tarkoitetun kantakomennon, ja tekevät siitä joko suorituskyvyttömän tai pahemmassa tapauksessa suorittavat jonkin toisen komennon.

```
SELECT * FROM taulu WHERE muuttuja=$muuttujan_arvo;
```

Edellä olevassa esimerkissä, jos $muuttujan_arvo -muuttujaan syötettäisiin ò; drop tablesò tämä aiheuttaisi hakulauseen suorituksen keskeyttämisen ja suorittaisi heti uutena komentona annetun òdrop tablesò Tätä kutsutaan injektiohyökkäykseksi. Suoritettava komento voi olla myös uusi hakulause, joka voi palauttaa jotain arkaluontoista tietoa hyökkääjälle.

Tätä varten MySQL tarjoaa työkaluja parametrien käsittelylle. Annettuun hakulauseeseen voidaan sijoittaa kysymysmerkkejä paikoille, joihin halutaan sijoittaa käyttäjäsyötettä. Hakulauseeseen voidaan sitten sitoa käyttäjäsyöte muuttujissa. Tämä aiheuttaa sen, että sidottu syöte välitetään erillisenä parametrina suoritettavalle kyselylle, eikä sitä tulkita ikinä osaksi kyselyä itseään.

2.3 RPM Package Manager

RPM Package Manager (rpm) on paketinhallintajärjestelmä, jota käytetään joissain Linux-käyttöjärjestelmissä ohjelmistojen asennukseen ja päivityksiin. Nimi rpm voi viitata pakettitiedostoon tai paketinhallintajärjestelmän ohjelmistoon. RPM-paketit luodaan spec-tiedostoissa määriteltyjen käsittelyohjeiden mukaan. Spec-tiedostoihin ja niiden käyttöön perehdytään osioissa 6.2.1 ja 6.2.2.

RPM-paketin asennus tapahtuu vaiheittain. Vaiheisiin kuuluu riippuvuuksien ratkaisu, paketoitujen tiedostojen purkaminen, asennustoimenpiteiden suoritus ja asennuksen siivous. Asennustoimenpiteet voidaan määritellä pakettia luotaessa spec-tiedostossa.

Asennetut RPM-paketit tallennetaan ohjelman ylläpitämään tietokantaan. Tätä tietokantaa käytetään muun muassa pakettien vaatimien riippuvuuksien olemassaolon tarkistukseen asennuksen yhteydessä. Tähän tietokantaan tallennetaan myös tiedot kaikista muuttuneista tiedostoista RPM-paketin asennuksen yhteydessä. Tällä tiedolla voidaan poistaa kaikki paketin tekemät muutokset eli käytännössä poistaa asennettu paketti.

RPM-pakettien nimeämistä varten on sovittu yhteiset muotoilusäännöt. Paketin nimen muoto on ònimi-versio-julkaisu.arkkitehtuuri.rpmò Yleisesti ottaen versiot ovat numerosarjoja, jotka on eroteltu pisteellä. Versiossakin voi kuitenkin olla tekstiä. Julkaisua voidaan käyttää kokonaisnumerona tai tarkoitetun käyttöjärjestelmän nimenä. Arkkitehtuurilla kerrotaan, mille järjestelmäarkkitehtuurille suoritettava käännetty koodi on tarkoitettu. Aina paketeissa ei kuitenkaan ole mitään arkkitehtuurikohtaista sisältöä, jolloin se voidaan asettaa arvoon ònoarchò

2.4 Yellow Dog Updater, Modified (ohjelman virallinen nimi, komm. huom.)

Yellow dog Updater, Modified (yum) on paketinhallintaohjelmisto, joka ylläpitää järjestelmän rpm paketteja. Sillä suoritetaan automaattisesti päivitysten hakeminen sekä asennus rpm-ohjelmalla. Yum-repositorioilla voidaan jaella ohjelmapäivityksiä, ja Linux-käyttöjärjestelmien julkaisijat ylläpitävät usein omaa repositoriotaan, josta on saatavilla vakaiksi todetut versiot yleisistä ohjelmistoista. Se on yleisesti käytössä useassa Linux-julkaisussa, kuten RedHat Linuxissa ja siihen pohjautuvassa CentOSissa.

Ohjelma käyttää repositoriopalvelimien tarjoamia xml-määritelmiä sisällöstään tuottamaan tulosteensa. Repositorion metadatatietoihin kuuluvat kaikki sen sisältämät paketit, niiden versiot ja pakettikohtaiset listat niiden sisältämistä tiedostoista. Repositoriolta voidaan tämän perusteella kysellä, että mikä paketti tulee ladata, jotta saadaan asennettua haluttu tiedosto. Tämä on hyödyllistä esimerkiksi silloin, kun jo asennettu ohjelma tuottaa virheviestejä puuttuvista kirjastoista eikä virheilmoituksesta voida päätellä tarvittavan paketin nimeä.

Ohjelmaa käytetään kutsumalla sitä konsolin komentoriviltä komennolla ò*yum*ò ja välittämällä haluttu toiminto kutsun parametreissa. Esimerkiksi saatavilla olevien pakettien listaus tapahtuu komennolla ò*yum list available*ò. Vastaavasti saatavilla olevat paketit ò*yum*ò-ohjelmasta listataan komennolla ò*yum list available yum*ò.

3 Päivitysjärjestelmistä

Ohjelmistojen täsmällinen ylläpito on tärkeää kaikissa tietokoneissa. Ohjelmistoihin julkaistaan päivityksiä jopa päivittäin, ja päivitysten sisältö ja tärkeys voivat vaihdella päivityksittäin ja ohjelmistoittain. Päivityksissä voidaan korjata pieniä huomaamattomia virheitä tai kriittisiä, toiminnan kannalta erittäin tärkeitä haavoittuvuuksia. Loppukäyttäjien tietokoneissa päivittämättömän ohjelmiston käyttäminen voi vaarantaa käyttäjän henkilökohtaisia tietoja, jos niitä syötetään haavoittuneeseen ohjelmaan. Internetselaimet ovat erityisen alttiita tietovuodoille, sillä ne ovat käytetyimpiä ohjelmistoja markkinoilla ja niitä käytetään suoraan tunnustautumiseen mahdollisesti arkaluontoistakin tietoa sisältäville sivustoille.

Yritysmaailmassa työntekijöiden päätteiden lisäksi on huolehdittava palveluja tarjoavien palvelinkoneiden ohjelmistojen ajantasaisuudesta. Heikkous jonkin kriittisen palvelinkomponentin palvelinohjelmistossa voi altistaa koko palvelimen rakenteen ja vuotaa arkaluontoista tietoa. Hyökkääjälle riittää tieto palvelimen käyttämästä ohjelmistoversiosta ja siihen tunnetusta tietoturvariskistä. Isoprofiilisten ohjelmistojen tietoturvariskeistä ilmoitetaan yleensä julkisesti noin viikon tai kuukauden päästä joko ohjelmiston julkaisijan tai julkisten tiedotustahojen toimesta. Ilmoituksen tultua julki, hyökkääjien on mahdollista selvittää julkisen tiedon perusteella, kuinka sitä voi hyödyntää ja etsiä kohteita, joissa kyseistä haavoittuvuutta ei ole vielä korjattu päivityksellä.

3.1 Yleistä

Lähestulkoon kaikissa ohjelmistoissa on jatkuvasti jotain korjattavaa tai paranneltavaa. Tämän vuoksi niihin tuotetaan usein erilaisia päivityksiä, jotka korjaavat edellisistä versioista löytyneitä virheitä tai lisäävät uutta toiminnallisuutta ohjelmistoon. Päivityksistä ei kuitenkaan ole ohjelmiston käyttäjille miltään hyötyä, jos he eivät saa päivitystä suo-

ritettua ohjelmistolleen. Tämän vuoksi on kehitetty erilaisia päivitysten jakelusta vastaavia järjestelmiä.

Lähes poikkeuksetta päivitysten toteuttamiseen tarvitaan jonkinlainen palvelinratkaisu, jolla pakettia levitetään. Palvelimena voi toimia oikeastaan minkälainen tietokone tahansa, kunhan sen toimintateho riittää tarvittavien palveluiden ylläpitoon ja vaadittavan käyttäjämäärän palveluun. Palvelimen tietoturvaohjelmistot ja jakeluohjelmistot sekä tarvittavat lisäkomponentit ovat yleisesti ottaen tarjolla jossakin muodossa kaikille yleisessä käytössä oleville käyttöjärjestelmille ja toimivat luottamuksellisuudeltaan ja teholtaan verrattavissa arvoissa. Kullakin käyttöjärjestelmällä on omia vahvoja ja heikkoja puoliaan, jotka voivat vaihdella vahvuudeltaan toisiinsa nähden eri versioiden välillä. Markkinatutkimusyritys IDC:n lehdistöjulkaisun mukaan markkina-osuudet palvelinmyynnissä jakautuvat suosituimpien palvelinkäyttöjärjestelmien Unix-, Linux- ja Windows-käyttöjärjestelmien kesken 45,8 % Windowsille, 24,2 % Unixille ja 18,4 % Linuxille [3].

Käyttöjärjestelmästä huolimatta tiedostojen jakelu niitä pyytäville tahoille on mahdollista toteuttaa luotettavasti käyttäen saatavilla olevia menetelmiä, jotka voivat vaihdella käyttöjärjestelmittäin tai olla saatavilla kaikille käyttöjärjestelmille.

3.2 Toteutustapoja

Yksinkertaisimmillaan päivitysjärjestelmät voivat olla web-sivuja tai tiedosto-palvelimia, joille päivityspaketteja asetetaan käyttäjien saataville. Käyttäjät hakevat päivityspaketit saatavilla olevilta palvelimilta käsin ja asentavat paketin sisällön erillisellä komennolla, kun paketti on ladattu päivitettävän koneen kiintolevylle. Tämänkaltaiset ratkaisut ovat riittäviä, kun kyseessä on ohjelmisto jonka ei tarvitse toimintansa kannalta olla aina päivitettynä uusimpaan versioonsa - eli melkein kaikki ohjelmistot, jotka pääsevät julkaistavaksi asti. Tämän ratkaisun heikkous on, että käyttäjien pitää tietää etsiä päivityksiä itse.

Nykyaikaiset ohjelmistot osaavat usein tarkistaa joltakin ylläpitopalvelimeltaan, onko siihen saatavilla uusia päivityksiä tai versioita. Ohjelmistot, jotka on suunniteltu olemaan jatkuvassa suorituksessa voivat suorittaa tarkistuksia tietyin väliajoin, esimerkiksi kerran päivässä. Ohjelmistot, joiden suoritus lopetetaan käytön jälkeen tarkastavat

yleensä päivitystilanteensa suorituksen alkaessa. Uuden version löytyessä ohjelmistot voivat toimia eri tavoilla ajantasaisuuden tarpeesta riippuen. Verkko-ohjelmistot, joiden palvelin- ja asiakasohjelmistojen tulee olla samoissa versioissa toimiakseen, eivät yleensä jatka ohjelmiston suoritusta mikäli päivitystä ei toteuteta. Yleisempää on, että käyttäjälle ilmoitetaan päivityksen saatavuudesta ja tälle tarjotaan mahdollisuutta joko hakea ja päivittää ohjelmisto automaattisesti tai avata käyttäjälle näkymä, josta tämä voi ladata päivityspaketin kiintolevylleen ja asentaa sen käsin.

Ajantasaisuutta vaativat verkko-ohjelmistot, joiden käyttäjämäärät ovat erittäin suuria voivat toteuttaa päivitysten jakelun käyttämällä Peer-to-Peer (P2P) tiedostonjako-ohjelmistoja, joissa tiedostojen siirto koneelta toiselle tapahtuu suoralla yhteydellä eri käyttäjien välillä. Esimerkiksi erittäin suositut Massively Multiplayer Online Roleplaying Game (MMORPG) -tyyppiset verkkopelit, joissa jopa miljoonat käyttäjät yhdistävät keskitetyille palvelimille, voivat hyödyntää P2P-teknologioita erittäin onnistuneesti, sillä kaikkien käyttäjien asiakasohjelmistojen täytyy olla yhteensopivia palvelinten sekä muiden käyttäjien ohjelmistojen kanssa, jotta järjestelmäkokonaisuus toimisi moitteettomasti. Tämä toteutustapa estää myös tehokkaasti päivitystä jakelevan yrityksen palvelimelle syntyvää verkkoliikenneruuhkaa, mikäli suuri käyttäjämäärä tarvitsee päivityksen yhtäaikaisesti. Jotta päivitysjärjestelmä olisi järkevää toteuttaa P2P-teknologialla, ohjelmistolla tulee olla tarve pitää kaikki asiakaskoneet samassa versiossa ja sillä pitää olla erittäin suuri käyttäjäkanta jotta päivityspaketin jakeluun ei synny katkoksia, kun P2P-verkon liikenteessä ei liiku ainuttakaan kappaletta osasta jaettavaa päivityspakettia.

3.3 Päivitysjärjestelmien tietoturva

Yleisesti ottaen päivitysjärjestelmät on tarkoitettu julkiseen käyttöön, jolloin päivityksistä vastaavien palvelimien tietoturvaratkaisuksi riittää perinteiset ja yleisessä käytössä olevat yksinkertaiset palvelinten tietoturvatoimenpiteet, kuten käyttäjätunnusten ja -oikeuksien hallinnointi ja tarpeelliset palomuurisäännöt, joilla estetään yksinkertaiset tunkeutumisyritykset. Tilanne monimutkaistuu, kun kyseessä on palvelin, jonka on tarkoitus jakaa päivityspaketteja vain luotetuille tahoille. Jaettavat tiedostot tai päivityspalvelin itse voi sisältää luottamuksellista tietoa, jonka vuotaminen julkisuuteen tai pahaaikeisille yksilöille voi altistaa palvelimen, sen omistajan sekä sen käyttäjät maineelli-

seen, rahalliseen tai jopa fyysiseen vaaratilanteeseen. Tällöin järjestelmän käyttöön tarvitaan jonkinlainen varmistus lataajan identiteetistä.

Perustietoturvan kannalta on tärkeää, että palvelimelle on toteutettu kovetus eli turhien palveluiden poiskytkentä tai niiden ohjelmakoodin poisto ja että palvelimen palomuurisäännöt on asetettu torjumaan yhdistämisyritykset porteista joissa ei odoteta kulkevan liikennettä. Muihin verkkopalveluita tarjoaviin palvelimiin verrattuna tiedosto- ja päivityspalvelimet ovat tärkeämpiä suojata huolella, sillä niiden tietoturvan vaarannuttua kaikki niiden käyttäjät voivat olla vaarassa jos jaettuihin tiedostoihin saadaan lisättyä vahingollista sisältöä. Tästä syystä on hyvin suositeltavaa että palvelimet on omistettuja tehtävälleen, jotta potentiaaliset tietoturvariskit rajautuvat tiedostojen jakoon liittyvään toiminnallisuuteen.

3.4 Tietoturvariskejä

3.4.1 Injektiohyökkäys

Injektiohyökkäyksessä hyökkääjä käyttää ohjelmiston syötteidenkäsittelyn puutteita hyväkseen ja voi päästä käsiksi tietoihin, joita käyttäjien ei ole tarkoitus kyetä saamaan. Pahimmassa tapauksessa haavoittuvuuden kautta voi päästä syöttämään suoritettavia komentoja ohjelman tietokannalle tai palvelimen käyttöjärjestelmälle. Näin voidaan aiheuttaa valtavaa vahinkoa palvelimelle ja sen omistavalle taholle.

Injektioiden torjunnan ratkaisu riippuu kyseessä olevasta alustasta ja ohjelmistosta. Ratkaisuihin liittyy kuitenkin aina syötteenkäsittelyn huolellinen tarkistus ja erotus suoritettavasta komennosta. Kun syötettä välitetään toiselle ohjelmalle, on suositeltavaa käyttää aidosti parametreja käyttävää ratkaisua. Mikäli parametriratkaisua ei ole saatavilla, ratkaisuksi voi riittää syötteen erikoismerkkien muuntaminen literaaleiksi merkeiksi, yleisesti niin sanotusti peittämällä ne, eli syöttämällä niiden eteen \-merkin. Tämä tulkitaan usein käsittelijöissä, joissa erikoismerkit tarkoittavat muuttujia, literaaliksi erikoismerkiksi muuttujan sijaan.

3.4.2 IP spoofing

IP spoofing eli saapuvien pakettien otsikkotietueessa ilmoitetun saapumisosoitteen väärentäminen on riski kaikissa järjestelmissä, joissa liikenne on rajattu IP-osoitteiden perusteella. Verkko-osoitteen väärennystä käytetään pääasiassa Denial-of-Service -hyökkäyksissä salaamaan paketteja lähettävän tahon oikea identiteetti, mutta sitä voidaan käyttää myös palvelimelle tunkeutumiseen, mikäli palvelin ei pysty tarkistamaan pitääkö ilmoitettu lähdeosoite paikkaansa. Tunkeutuminen on tällöin mahdollista esimerkiksi tilanteessa, jossa palvelin on asetettu olemaan hyväksymättä yhteyksiä muualta kuin sisäverkosta ja ulkoverkosta saapuvan paketin lähdeosoitetietue on väärennetty sisäverkon osoitteeksi. Jokaisen palvelimelle lähetetyn paketin tietueiden väärennys on työlästä, mutta kun kyseessä on palvelin, jonka toiminnallisuus ja sisältö on arkaluontoista tai luottamuksellista, on syytä nähdä vaivaa IP-osoitteiden oikeellisuuden tarkistamiseen.

3.4.3 Palvelunestohyökkäys - Denial of Service (DoS)

Palvelunestohyökkäysten tarkoitus on ruuhkauttaa palvelin turhilla palvelukutsuilla, jolloin kaikki palvelimen resurssit käytetään hyökkääjän kutsuihin sen sijaan, että muita asiakkaita palveltaisiin. Palvelunestohyökkäysten yksityiskohdat riippuvat palvelimen tarjoamista palveluista. Yleisesti ottaen kaikki julkiset verkkopalvelut ovat jollakin tapaa alttiita palvelunestohyökkäyksille. Kun hyökkääjiä on useita ja useista eri paikoista, puhutaan hajautetuista palvelunestohyökkäyksistä. Julkisissa palveluissa näiden torjuminen on erittäin hankalaa, jopa mahdotonta sillä hyökkäykset eivät tule yksistä tietyistä paikoista, joista voitaisiin estää liikenteen vastaanotto. Hyökkäysten lähteet voivat olla väärennettyjä osoitteita tai oikeiden viattomien käyttäjien koneita, jotka ovat altistuneet hyökkääjän levittämälle haittaohjelmalle, joka suorittaa hyökkääjän määrittämiä kyselyitä kohdepalvelimelle tämän käskystä.

3.4.4 Cross-site Scripting (XSS)

Cross-site Scriptingillä tarkoitetaan haavoittuvuutta verkkopalvelun syötteenkäsittelyssä tai esityksessä. Aina kun verkkopalvelu käsittelee käyttäjäsyötettä, joka näytetään takaisin käyttäjälle, voidaan syötteeseen upottaa html-komentoja, jotka käyttäjän selain tulkitsee standardiensa mukaisesti. Jos käyttäjäsyöte tallennetaan ja näkyy toisillekin käyttäjille, haavoittuvuus voi altistaa muut käyttäjät tietoturvariskeille. Pahimmissa ta-

pauksissa tallennettavaan syötteeseen voidaan upottaa toiminto, joka lähettää muiden käyttäjien tietoja ulkopuoliselle taholle.

Mikäli hyökkääjä tietää, että syötteen käsittelyssä käytetään muuttujia, hyökkääjä voi lisätä syötteeseensä näiden muuttujien tulosteita. Näin hyökkääjä voi saada tietää arkaluontoista tietoa palvelun toiminnasta.

Cross-site Scripting torjutaan yleisesti syötteenkäsittelyssä käsittelemällä kaikkia erikoismerkkejä html-entiteetteinä literaalien erikoismerkkien sijaan. Esimerkiksi perl-ohjelmointikielessä dollari-, at- ja prosenttimerkit ovat muuttujien aloitusmerkkejä ja niihin voidaan yrittää viitata käyttäjän syötteessä. Html-entiteetteinä ne ovat käsittelyssä vähintään neljä erillistä merkkiä.

3.4.5 Palvelimena esiintyminen

Kun käyttäjä yhdistää palvelimelle millä protokollalla tahansa, palvelimen oikeellisuudesta ei oletuksena ole mitään takeita. Esimerkiksi julkiset verkkoliikenteen ohjaukset on voitu saada osoittamaan väärälle palvelimelle. Tällöin yhdistäjä altistaa itsensä tietoturvariskille ja voi luovuttaa luottamuksellisia tietoja ulkoiselle taholle.

Yleisessä käytössä oleva tapa varmistua palvelimen oikeellisuudesta on varmennetut palvelinsertifikaatit, joita tarkistellaan yhteyttä luotaessa. Tietyt varmentajatahot varmentavat maksua vastaan pyydetyn sertifikaatin tiedot oikeiksi. Sertifikaattien suurin ongelma on se, että varmentajatahoihin pitää pystyä luottamaan. Varmentajia on monia eri luottamustasoja, joista jotkin eivät ole itse varmennusta varmempia ja joitain pidetään erittäin luotettavina. Täysin luotettavaa tapaa tunnistaa palvelin ei ole vielä tänä päivänä kehitetty.

4 Deltagon Group Oy

Tämä työ on toteutettu Deltagon Group Oy:lle. Deltagon Group Oy on suomalainen yhtiö, jonka asiakkaina on pääasiassa suuria ja keskisuuria suomalaisia yrityksiä. Yhtiö erikoistuu sähköpostin tietoturvaan ja tarjoaa tärkeimpänä tuotteenaan tietoturvallisen sähköpostin lähettämiseen ja vastaanottamiseen tarkoitettua D3-ratkaisua. Tuote asennetaan oletuksena omaksi palvelimekseen asiakkaan sisäverkkoon, eikä siihen

yhdistetä muita toiminnallisuuksia kuin saman tuoteperheen jäseniä. D3-ratkaisuun kuuluu hallinta-paneeli, jolla sen komponentteja voidaan asettaa palvelimen omistajan toimesta. Hallintapaneeliin tahdotaan työkalu palvelimen koodikannan päivitykselle, joka pyritään tässä työssä toteuttamaan. Palvelimet voivat koostua myös useasta solupalvelimesta, jotka kaikki tulee tuoda ajan tasalle samalla työkalulla.

5 Taustaa

5.1 Tilanne ennen järjestelmää

Tällä hetkellä yrityksen tarjoamien ohjelmistopalvelujen ylläpito ja ajantasallapito on toteutettu yrityksen tuottamien sisäisten päivitysohjedokumenttien mukaisesti manuaalisesti suoran SSH-yhteyden ylitse. Päivityksiin liittyy muutetun ja lisätyn koodikannan lisäksi erilaisia muutoksia järjestelmän asetusarvoihin, jotka tehdään käsin dokumentin ohjeiden mukaan. Myös kaikki uudet perl-moduulit, tietokantamuutokset ja muut koodikantaan liittymättömät muutokset tehdään käsin ohjeen mukaan. Päivitysohjedokumentit saattavat olla kymmenenkin sivun pituisia, ja osa toiminnoista saatetaan tarvita vain yhdelle asiakaspalvelimen monesta solusta. Kun tarkkuutta vaativia tehtäviä lisätään päivitysprosessiin, niiden tekemiseen käytettävä aika kasvaa ja monistuu jokaista päivitettävää kohdetta kohden.

Päivitysprosessi tahdotaan saada mahdollisimman automaattiseksi ja geneeriseksi, jotta asennusten ja päivitysten toteuttamiseen käytettävä aika saataisiin minimoitua ja eri asiakaspalvelinten väliset päivitystoimintatavat yhtenäistettyä. Yrityksen sisäisesti päätettiin, että päivitykset tullaan toteuttamaan RedHat-pohjaisten Linux-käyttöjärjestelmän RPM Package Manager -paketinhallintaohjelmiston standardien mukaisilla asennus- ja päivityspaketeilla. Näin ohjelmiston asennus asiakkaiden palvelinkoneille tapahtuu aina samalla tavalla ja saman prosessin mukaisesti, ja yrityksen kehitystiimin työpanos saadaan kohdistettua toimivan päivityspaketin luomiseen ja testaamiseen sen sijaan, että jokaisen asiakaspalvelimen päivityksen toteutukseen ja varmistamiseen käytetään paljon aikaa. Näin asennukseen liittyvät yksilölliset riskitekijät minimoidaan ja työaika saadaan panostettua kehitys- ja testausprosessin toteuttamiseen yrityksen omassa asiakasympäristöä simuloivassa testiympäristössä.

5.2 Päivitysjärjestelmän suunnittelu- ja päätösprosessi

Kun päivitysjärjestelmän toteuttaminen otettiin harkintaan yrityksen palavereissa, sen toteutusmahdollisuuksista tuotettiin raportti, jossa kuvattiin ja suunniteltiin mahdollisia toteutustapoja. Yrityksen työntekijät suunnittelivat tapoja toteuttaa HTTPS-kyselyillä toimiva päivitysten kysely- ja latauspalvelu, jossa kyselijät tunnistetaan kyselyn yhteydessä lähetetyllä asiakaspalvelimen SSL-sertifikaatin osasta luodulla avaimella, jota verrataan palvelimella sijaitsevaan vastakappaleeseen. Tarkistuksen onnistuttua palvelin lähettäisi vastauksena listan saatavilla olevista päivityksistä tai päivityspaketin tiedostona. Päivityspaketti asennettaisiin käsin tai käyttöliittymän kautta RPM-paketin asennus- tai päivityskomentoa käyttäen.

Mahdollisia olemassa olevia toteutusmahdollisuuksia tutkittaessa päädyttiin Yum-ohjelmalla toteutetun tiedostopankin eli repositorion ympärille toteutettavaan ratkaisuun. Yum eli ðYellow Dog Updater, modifiedò on avoimen lähdekoodin RPM-paketteihin perustuvien Linux-käyttöjärjestelmien paketinhallintaohjelma, jota Deltagon Group Oy:n asiakaspalvelimet käyttävät käyttöjärjestelmiensä ylläpitoon. On siis yksinkertainen ja luonnollinen ratkaisu käyttää samaa ohjelmistoa palvelimen tarjoamien sähköpostipalveluiden lähdekoodin ajantasaisuuden ylläpitoon.

Päivitysjärjestelmän implementointi päätettiin toteuttaa askeleittain alkaen päivityksen toteuttavan paketin suunnittelusta ja kokoamisesta. Kun päivityspaketteja pystytään tuottamaan ja niillä voidaan toteuttaa kaikki päivitykselle oleelliset toimenpiteet, niiden jakelu voidaan toteuttaa alustavasti vaikka yksinkertaisesti siirtämällä paketti asiakaspalvelimelle ja suorittamalla asennus käsin käyttäen RPM-pakettien asennusohjelmaa. Näin asiakasverkosto valmisteltaisiin päivityspakettien automaattiseen hakemiseen ja asentamiseen Deltagon Group Oy:n tarjoaman päivityspalvelimen avulla.

Päivityspakettien luonnin kiinteydyttyä keskitytään pakettien jakeluun yum-repositorion ja -asiakasohjelman avulla. Päivityspalvelin saatetaan yrityksen verkkokokoonpanoon siten, että asiakaspalvelimet pystyvät yhdistämään sinne yum-asiakasohjelmaa käyttäen. Päivityspalvelimelle siirretään valmistuneet päivityspaketit yrityksen sisäverkosta käsin. Asiakaspalvelimille sallitaan uloslähtevät yhteydet päivityspalvelimelle. Samalla luodaan päivityspalvelimelle yhdistämiseen tarvittavat asetustiedostot yum-ohjelmalle ja ajastetaan päivitysten kysely tasaisin väliajoin.

Tämän vaiheen jälkeen voidaan keskittyä kehittämään päivitysjärjestelmän lisätoiminnallisuutta ja muokattavuutta. Asiakaspalvelimet asetetaan pystymään suorittamaan päivityksiä eriasteisille päivityksille: virhekorjaukset ja toiminnallisuuspäivitykset. Asiakaspalvelinten ylläpitäjille tarjotaan käyttöliittymä päivitysten asennusta ja automatiikan asetuksia varten. Päivityspalvelimelle toteutetaan erilaisia valvontatyökaluja joilla päivitysten kulkua ja mahdollisia virheitä voidaan valvoa. Päivityspalvelimen yhteyteen toteutetaan työkaluja, joilla voidaan ilmoittaa asiakkaille uusien versioiden ja päivitysten saatavuudesta.

Järjestelmän tulee toimia jokaisella toteutusasteella. Toteutuksen pääpaino on päivitysten toiminnalla ja sillä, että asiakaspalvelimet saadaan päivitettyä tietoturvallisesti.

6 Päivityspaketin luonti ja testaus

6.1 Yleistä

Päivityspakettina toimii Redhat-pohjaisissa Linux-ympäristöissä käytettävät RPM Packet Manager -paketinhallintaohjelmalla luodut ohjelmapaketit, joissa suoritetaan kaikki ohjelmiston päivittämisprosessissa tapahtuvat toiminnot. Päivityspaketti luodaan ohjelmiston koodikannasta sekä joukosta järjestelmälle suoritettavista toiminnoista jotka muokkaavat ohjelmiston omia tai kolmannen osapuolen ohjelmistojen asetustiedostoja.

6.2 Luonti

RPM-paketit luodaan käyttöjärjestelmän rpmbuild-ohjelmalla. Tälle ohjelmalle välitetään ennalta luotu spec-tiedosto, jossa on määritelty paketin rakenne sekä asennuksen tai päivityksen yhteydessä suoritettavat toiminnot kussakin asennuksen vaiheessa.

6.2.1 Yleistä spec-tiedostosta

Spec-tiedosto koostuu otsikkotietueista sekä prosenttimerkkialkuisista ennaltamääritellyistä merkkijonoista, jotka merkitsevät kunkin asennusprosessin vaiheen suoritusosiota. Otsikkotietueissa kerrotaan päivityksen versio ja muita oleellisia tietoja

paketista sekä lyhyt kuvaus paketin sisällöstä. Asennusprosessin eri vaiheiden sisältö riippuu ohjelman toteutustavasta ja vaadittavien toimenpiteiden monimutkaisuudesta.

RPM-paketin luova rpmbuild-ohjelma suorittaa paketin luomisen yhteydessä tarvittavien alkutoimenpiteiden lisäksi spec-tiedostossa määritellyt skriptitoiminnot. Ohjelman tunnistamat skriptit suoritetaan paketin luonnin, asennuksen tai poistamisen tietyissä vaiheissa. Kullekin suorituksen vaiheelle on oma nimetty skriptinsa, jonka sisään voidaan kirjoittaa suoritettavat komentokehotetoiminnot.

Taulukko 1. RPM-paketin luomisprosessin skriptit

Suoritusvaihe	**Skripti**	**Kuvaus**
Alkutoimenpiteet	%prep	Suoritetaan ennen paketin sisällön kokoamista. Esimerkiksi tarpeellisten lähdekoodien purkaminen paketin rakennushakemistoon.
Paketin kokoaminen	%build	Paketin sisällön viimeistelevä vaihe. Usein yksinkertainen ‘make’-komennon suoritus, joka kääntää lähdekoodin suoritettavaksi ohjelmaksi.
Asennus	%install	Build-osiossa suoritetun ohjelmiston asentamiseen tarvittavat toimenpiteet. Usein yksinkertainen ‘make install’-komento.
Siivous	%clean	Paketin rakennuksessa käytettyjen hakemistojen ja tiedostojen poistaminen.

Alkutoimenpiteet-vaiheessa on käytettävissä myös kaksi hyödyllistä makrotoiminnallisuutta: %setup ja %patch. Nämä ovat ennalta määritettyjä joukkoja toimintoja, jotka voidaan suorittaa helposti makroa kutsumalla. %setup poistaa mahdolliset olemassa olevat tiedostot rakennushakemistosta, purkaa RPM-paketin pakatun lähdekoodin rakennushakemistoon, vaihtaa työhakemiston juuri puretun paketin luomaan hakemistoon ja lopulta asettaa hakemiston ja sen alihakemistojen tiedostojen ryhmä- ja omistajaoikeudet oletusarvoiksi. Kaikki %setup-makron toiminnot ovat myös ohitettavissa ja niihin liittyvät arvoparametrit (kuten työhakemiston nimi) ovat muutettavissa makroa kutsuttaessa käytettävissä olevilla parametreilla. [4.]

Spec-tiedostossa määritellään myös paketin sisällön asentamisen ja poistamisen toimenpiteet niille varatuilla skriptiosioilla. Kullekin toiminnolle on varattu alustus- ja viimeistelyskriptit, jotka suoritetaan paketin sisällön asennuksen tai poistamisen yhteydessä. Asennuksen alustustoimenpiteet suoritetaan %pre-skriptissä ja jälkitoimenpiteet %post-skriptissä. Harva ohjelma vaatii mitään alustustoimenpiteitä, mutta jälkitoimenpiteissä voidaan tehdä paljonkin muutoksia. Pakettia poistettaessa suoritetaan vastaavat %preun- ja %postun-skriptit, joissa siivotaan paketin jättämät jäljet järjestelmästä.

Alustus- ja viimeistelyskriptojen lisäksi asennuksissa ja poistamisissa järjestelmän kiintolevylle siirretään kaikki spec-tiedoston %files-osiossa listatut tiedostot. Tiedostojen tulee löytyä RPM-paketin kokoamishakemistopolusta alkaen samasta hakemistopolusta kuin mihin se on %files-listaan merkitty. Pakettia asennettaessa tiedostot kopioidaan vastaavaan absoluuttiseen polkuun. Esimerkiksi pakettia koottaessa %files-listalta löytyvä tiedosto ðetc/ohjelma.cfgò kopioidaan RPM-tiedoston sisään oletuskokoamishakemistoa käytettäessä tiedostosta ðusr/src/redhat/BUILD/ohjelma-1.0/etc/ohjelma.cfgò ja kopioidaan asennettaessa ðetc/ohjelma.cfgò-tiedostoksi. %files-listalle voidaan merkitä myös hakemistoja, jolloin pakettiin sisällytetään hakemisto ja kaikki sen alta löydetyt tiedostot ja hakemistot sisältöineen.

6.2.2 Deltagon D3 spec -tiedosto

Deltagon D3 -ohjelmiston RPM-tiedostoa luotaessa tarvitaan kustomoitu spec-tiedosto, jossa määritellään tarvittavat järjestelmämuutokset. Asennuksen eri vaiheissa tarvitaan useita tarkistuksia, lisäyksiä ja muutoksia järjestelmän muihin komponentteihin. Nämä toteutetaan kustomoidussa spec-tiedostossa.

Koska Deltagon D3 -ohjelmisto on suunniteltu olemaan jatkuvassa suorituksessa, siihen liittyy palveluita, jotka ovat niin sanotussa ðdaemonò-tilassa eli jatkuvassa suorituksessa järjestelmän taustalla. Koska nämä palvelut toteuttavat muun muassa järjestelmässä kulkevan sähköpostin käsittelyä, niiden suoritus täytyy keskeyttää, jotta järjestelmä ei yritä suorittaa toimintoja, jotka suorittava ohjelmakoodi ei ole sillä hetkellä saatavilla. Päivitysten yhteydessä saatetaan myös tarvita muutoksia järjestelmän muihin komponentteihin, kuten MySQL-tietokantarakenteisiin tai järjestelmän asetustiedostoihin. Järjestelmän toiminnan takaamiseksi vaaditaan myös tiettyjä tarkistuksia ja varmennuksia, kuten tiettyjen järjestelmäkäyttäjänimien olemassaolon. Sillä RPM-pakettien käyttö tulee käyttöön uutena päivitystapana jo olemassa oleviin asiakaspalve-

limiin, spec-tiedoston pitää myös pystyä tunnistamaan ja käsittelemään jo olemassa olevat asennukset, joita ei ole toteutettu RPM-pakettia käyttäen. Näistä syistä spec-tiedoston sisältö pitää muokata sisältämään tarvittavat toimenpiteet oikeissa suorituksen vaiheissa.

```
root@secure:~/tkanto/rpm
Summary: Deltagon D3
Vendor: Deltagon Group Oy
Name: D3
# change this accordingly
Version: 2.7.6
Release: 1
Source: %{name}-%{version}.tar.gz
License: Deltagon
Group: Applications
# dependencies are manually resolved
Autoreq: 0

# this defines build chroot, not all sections use this
BuildRoot: /usr/src/redhat/BUILD/%{name}-%{version}/

###### CONFLICTS
# All software that is subject to NOT run on the same server
# should be listed here
# do checks for this in kovetus
#Conflicts: sendmail, sendmail-cf
#Conflicts: squid
######
```

Kuva 1. Deltagon D3 -ohjelmiston spec-tiedoston yleistiedot

Spec-tiedoston sisältö alkaa paketin sisällön perustiedoilla, kuten kehittäjän ja julkaisijan nimillä. Kuvassa 1 esitetty spec-tiedoston alkuosa sisältää tarvittavat tiedot paketin sisällöstä. Kuvassa esitetyssä tiedostossa on esitelty myös spec-tiedoston kommenttirivituki ò#ò-alkuisille riveille. Näin tiedostoon voidaan lisätä vapaasti rivejä, joille voidaan kirjoittaa muistiinpanoja ja huomautuksia, jotta tiedoston tulkinta on helpompaa kehittäjälle, joka ei tunne sen rakennetta ja sisältöä. Tiedostossa esiintyviä òTietue: Arvoò-pareja kutsutaan tageiksi, ja tiedoston alkuosa koostuu joukosta näitä tageja. Source-tagi määrittää pakatun lähdekoodin sisältävän tiedoston, jonka rpmbuild purkaa kokoamishakemistorakenteeseensa ja josta tulee löytyä ainoastaan tiedostot, jotka on määritelty spec-tiedoston %files-osiossa. Autoreq-tagilla kontrolloidaan ohjelman tarvitsemien muiden ohjelmistojen eli riippuvuuksien automaattista latausta ja asennusta. Arvolla nolla puuttuvista ohjelmistoista varoitetaan kun pakettia yritetään asentaa. BuildRoot-tagi on kuvan 1 esimerkissä asetettu rpmbuildin oletusarvoksi, ja se määrittelee

mihin hakemistoon Source-tagin määrittelemä paketti puretaan ja missä ohjelman pakkaus.rpm-paketiksi suoritetaan.

```
root@secure:~/tkanto/rpm
###### DEPENDENCIES

# used in some legacy scripts
Requires: expect >= 5.43

# needed by Crypt::GCrypt
Requires: libgcrypt >= 1.2.4, libgcrypt-devel >= 1.2.4

# HTTPS UI
Requires: httpd >= 2.2.3, mod_ssl >= 2.2.3
#, mod_security >= 2.5.9

# self-explanatory
Requires: postfix >= 2.3.3
Requires: mysql >= 5.0.45, mysql-server >= 5.0.45
Requires: perl >= 5.8.8
Requires: postmantis >= 0.45

# quarantinecopy
Requires: wget

# htmlprocess
Requires: lynx

# eth0:x interfaces
Requires: fake
######
```

Kuva 2. Riippuvuuksien määrittely Deltagon D3 -ohjelmiston spec-tiedostossa

Ohjelmiston riippuvuudet määritellään Requires-tagilla. Kuvassa 2 on esitetty Deltagon D3 -ohjelmiston riippuvuuksien määrittely. Kuten kuvassa 2 on esitelty, Requires-tagi voidaan jakaa useaan osaan jolloin vaaditut riippuvuudet voidaan jaotella selkeyden vuoksi omiin joukkoihinsa. Notaatiolla ònysql >= 5.0.45ò tarkoitetaan, että spec-tiedoston ohjelmisto vaatii ohjelman ònysqlò-version 5.0.45 tai uudemman olevan asennettuna, jotta ohjelmisto toimisi oikein. Riippuvuudet voidaan määritellä myös ilman versionumeroa, jolloin riittää, että vaaditusta ohjelmasta on asennettuna mikä tahansa versio. Pakettia asennettaessa tarkistetaan, onko järjestelmään asennettu tarvittavat ohjelmat. Asennus keskeytetään, mikäli jokin vaadittu ohjelma puuttuu.

```
root@secure:~/tkanto/rpm
%description
Deltagon D3 is easy to use and scalable solution for confide
ntial e-mailing.
System works without installations on sender or receiver wor
kstations.
Solution utilises the e-mail system in use and is also
suitable for adhoc messaging. D3 solution enables protected
two-way e-mail traffic.

# couple of rpm macros
%prep
%setup
%install
# can use $RPM_BUILD_ROOT here

# clean build root
%clean
rm -rf $RPM_BUILD_ROOT
```

Kuva 3. Kuvaus ja makrojen suoritus

Kuvassa 3 esitetään ohjelmiston kuvauksen määrittely sekä komennot, jotka suoritetaan, kun paketti kootaan lähdetiedostoista. Kuvaus eli %description upotetaan rpm-paketin otsikkotietueisiin, josta paketteja käsittelevät ohjelmistot voivat lukea ja esittää sen. Kuvaus voi olla hyvinkin kattava tekstimuotoinen esitys ohjelmiston toiminnallisuudesta tai yksinkertainen ja mainospuhemallinen ohjelmiston kuvaus. Deltagon D3 -päivityspaketin kuvauksena toimii osa ohjelmiston tuotekuvauksen johdantoa.

Paketin kokoaminen toteutetaan oletusmakroja käyttäen. Kokoamiseen tarvitaan siis vain lähdekoodin sisältävä .tar.gz-paketti, jonka tulee sijaita spec-tiedostossa määritellyssä hakemistopolussa tai kokoamispalvelimen käyttöjärjestelmän oletushakemistopolussa.

```
root@secure:~/tkanto/rpm
%files
# folders/files that are taken into rpm file
# can has NOT use $RPM_BUILD_ROOT here

## codebase
/server/base/bin
/server/base/lib
/server/base/sendmodules
/server/base/www
/server/base/error
/server/base/adminlib
/server/base/admin_tools
/server/base/help
/server/base/mimetypes
/server/base/packets
/server/base/profile

# these are empty dirs in RPM, data inside is preserved over
 updates
/server/base/temp
/server/base/log
/server/base/dashboard
```

Kuva 4. Pakettiin sisällytettävät tiedostot

Pakettiin sisällytettävät tiedostot määritellään spec-tiedoston %files-osiossa. Deltagon D3 -ohjelman pakettiin sisällytetään kaikki kokoamishetkellä käytössä olevat ohjelmakooditiedostot ja mahdolliset kokonaan uudet asetustiedostot.

6.3 Testaus

Ennen kuin ohjelmiston uusi versio voidaan lähettää asiakaspalvelimille päivitettäväksi, sen toiminnallisuus ja yhteensopivuus vanhojen versioiden kanssa täytyy testata ja todeta toimivaksi. Tätä varten tarvitaan testausprosessi, jolla varmistutaan päivityspaketin toimivuudesta. Testausprosessiin täytyy kuulua tarvittava määrä testipalvelimia, joista osalle tulee olla asennettuna ohjelmiston edeltäviä versioita. Tämä on tarpeellista, sillä ohjelmiston toiminnallisuus on hyvin riippuvainen kolmansien osapuolien tuottamista ohjelmistoista ja vaatii niiden olevan asetuksiltaan kykeneviä käsittelemään ohjelmiston tuottamaa dataa. Esimerkiksi ohjelmiston käyttämän tietokannan tulee sisältää kaikki tietueet, jotka siihen on lisätty muiden päivitysten yhteydessä, jotta ohjelmiston oikeanlainen toiminnallisuus voidaan taata.

Testausprosessiin kuuluu koodikannan asennus- ja päivitystoimenpiteet sekä toiminnallisuuden testaustoimenpiteiden jälkeen. Jokainen päivityspaketti testataan samalla testausprosessipohjalla, mutta testattavaksi merkitään myös paketin tuomat uudet toiminnallisuudet. Uudet toiminnallisuudet, joiden käyttötarpeen ja -asteen oletetaan olevan korkea, sisällytetään olemassa olevaan testausprosessipohjaan, jolloin ne testataan myös jokaista tulevaa päivitystä testatessa. Korkean käyttöasteen toiminnallisuudella tarkoitetaan tässä yhteydessä toiminnallisuutta, joka suoritetaan osana ohjelmiston perus-toiminnallisuutta, kuten sähköpostiviestien vaihtoehtoinen käsittely- tai luomistapa tai on osa ohjelmiston käyttöliittymää.

Päivitetyn ohjelmiston testausta varten on varattava ainakin kaksi palvelinkonetta: yksi palvelinkone, jolle on asennettu ohjelmiston sillä hetkellä tuotannossa oleva ohjelmisto sekä toinen uusia asennuksia varten. Lisäksi tarvittaessa voidaan varata palvelinkoneita joille on asennettu vanhempia versioita ohjelmistosta, mikäli tahdotaan testata päivitystä vanhemmasta versiosta uusimpaan versioon. Koska testipalvelinkoneiden ei tarvitse kyetä samaan käsittelytehoon kuin tuotannossa oleva asiakaspalvelin, niiden ei tarvitse olla täysin asialleen omistettuja kokonaisia palvelinratkaisuja, vaan voidaan toimia virtualisoidussa palvelinympäristössä.

Kun päivityspaketti on luotu, testattu ja todettu täysin toimivaksi, se voidaan siirtää jaeltavaksi päivityspalvelimelle.

7 Päivityspalvelin

7.1 Yleistä

Päivitysten jakelun kannalta järjestelmän oleellinen komponentti on päivityksiä jakeleva palvelin. Palvelimen kiintolevylle on tallennettu kaikki ohjelmistopaketit ja niiden päivityspaketit jotka on julkaistu levitykseen. Asiakaskoneet suorittavat päivityspalvelimelle versionumerokyselyitä, joiden perusteella asiakaskone päättää, tarvitseeko se päivityspalvelimelle tallennettua päivityspakettia ollakseen ajan tasalla. Mikäli päivityspalvelimelta löytyy versio kysellystä ohjelmasta, joka on uudempi kuin asiakaskoneelle asennettu ohjelmaversio, asiakaskone pyytää päivityspalvelinta lähettämään uuden version päivityspaketin omalle kiintolevylleen.

Päivityspalvelimen päivityspakettien jakaminen toteutetaan yum-paketinjakeluohjelmaa hyödyntäen. Päivityspalvelimelle asennetaan HTTPS-protokollalla yhteyksiä hyväksyvä yum-repositorio, johon asiakaspalvelimet yhdistävät yum-ohjelman asiakasohjelmistolla. Repositorio asennetaan päivityspalvelimelle vapaan lähdekoodin HTTP-palvelinohjelmisto Apachen avulla SSL suojatuksi HTTPS-palvelimeksi.

7.2 Päivityspalvelimen suojaus

Koska päivityspalvelin on erittäin tärkeä osa päivitysjärjestelmää, se on syytä suojata huolella ylimääräiseltä liikenteeltä, joka voisi häiritä sen toimintaa. Lisäksi palvelimelle pääsy muilta kuin luotetuilta lähteiltä on syytä estää, sillä palvelimelle päässyt tunkeutuja voi saada käsiinsä palvelimelta löytyvää arkaluontoista tietoa tai pahimmassa tapauksessa vaikuttaa palvelimen tarjoamien palveluiden saatavuuteen ja toimivuuteen.

Päivityspalvelimen ja verkon, johon se on yhdistetty, välisen suhteen toimintamallina käytetään mallia, jossa asiakaskoneet tai muut ulkoiset käyttäjät eivät koskaan lähetä päivityspalvelimelle pysyvää dataa - päivityspalvelimelle lähetetään kyselyitä, joihin palvelin vastaa lähettämällä kyselyssä pyydettyä dataa mikäli kyselijällä todetaan olevan oikeus vastaanottaa kyseistä dataa. Kun tätä toimintamallia noudatetaan, vältytään erilaisilta haavoittuvuuksilta, joilla ulkopuoliset tekijät voisivat esimerkiksi muokata järjestelmän toiminnan kannalta tärkeitä asetustiedostoja tai syöttää päivityspalvelimen jakelemien pakettien joukkoon väärennettyjä paketteja, jotka altistaisivat kaikki päivityspalvelinta käyttävät asiakaskoneet hyökkäykselle. Ainoa taho, jolla on oikeus lähettää päivityspalvelimelle levylle tallennettavia tiedostoja, on päivityspalvelimen tarjoaman yrityksen palvelin, jolla julkaistavat päivityspaketit testataan ja varmistetaan toimiviksi. Päivityspalvelinta ei myöskään koskaan käytetä päivityspaketin kokoamispalvelimena jotta ulkopuoliset tahot eivät pysty syöttämään lähdekoodiin omia muutoksiaan, vaikka päivityspalvelimelle päästäisiin jotenkin käsiksi.

Poikkeuksena siihen, että päivityspalvelimelle ei lähetetä pysyvää dataa, on päivitysraporttien lähetyspalvelu. Tämä palvelu toteutetaan siten, että syötettyä dataa ei käytetä koskaan suoraan määrittelemään tiedostojen nimiä, vaan nimi luodaan istunnon tietojen perusteella. Syötettyä tietoa ei koskaan käytetä muuhun kuin luodun tiedoston sisälle syöttämiseen turvalliseksi todetuilla perl-toiminnoilla.

Kaikki palvelimelle otettavat yhteydet varmennetaan ennen kuin mitään toimintoja suoritetaan. HTTPS-kyselyissä varmennukseen käytetään SSL-sertifikaattia ja palvelimella ylläpidettävää sallittujen kyselijöiden IP-listaa.

Ennen päivityspaketin lähetyksen pyytämistä asiakaskone suorittaa päivityspalvelimelle HTTPS-kyselyn, jossa pyydetään myöntämään kyseessä olevalle asiakaskoneelle oikeus yhdistää päivityspakettien repositorioon ja suorittaa sen käyttöön liittyviä toimintoja. Näin pääsy palvelimelle rajataan vain hallituille IP-osoitteille ja yhteyksien luomistapahtumia voidaan valvoa erittäin tarkasti.

7.3 Tekninen toteutus

7.3.1 Palvelimen käyttöjärjestelmän asennus

Päivityspaketteja jakelevan palvelinkoneen ohjelmistojen asennus ja konfigurointi aloitetaan alustamalla palvelimen kiintolevyt ja asentamalla käytettävä käyttöjärjestelmä. Deltagon Group Oy:n käyttöön valittu käyttöjärjestelmä on RedHat-pohjainen CentOS Linux-käyttöjärjestelmä. Käyttöjärjestelmän asennus suoritetaan kickstart-menetelmällä, jossa asennusohjelmalle annetaan syötteeksi tiedosto, jossa kerrotaan käyttöjärjestelmän asetukset sekä kaikki asennuslevyltä asennettavat paketit. Kickstart-tiedostoa käytetään Deltagon Group Oy:n tarvitsemien uusien palvelimien alustamiseen. Kaikki palvelimet alustetaan tämän tiedoston ohjeiden mukaan ja palvelimen toiminnasta ja tarpeista riippuen tarvittavat lisäpaketit asennetaan käyttöjärjestelmän asennuksen jälkeen.

Päivityspalvelimen tarvitsemat oleelliset ohjelmistot ovat *mysql*, *mysql-server* ja *iptables*. Näillä ohjelmistoilla pidetään yllä palvelimen tallettamia asiakas-tietoja ja palvelimelle sallittuja yhteyksiä. Käyttöjärjestelmän asennuksen jälkeen tarvitaan vielä joitakin paketteja jotta palvelimella on kaikki järjestelmän tarvitsemat ohjelmistot. Tarvittavat lisäohjelmat ja moduulit ovat *apache*, *mod_ssl* ja *createrepo*, joilla ylläpidetään palvelimen HTTPS-rajapintaa ja luodaan tarvittavat metatiedot palvelimella sijaitsevien RPM-pakettien jakeluun

7.3.2 Yum-repositorion, apachen ja palomuurin konfigurointi

Päivitysjärjestelmän toiminnan kannalta tarvittavat ohjelmistot ovat *createrepo* ja *apache*, jotka yhdessä tarjoavat kaiken tarpeellisen toiminnallisuuden pakettien jakelua varten HTTPS-protokollan avulla. Ensin valitaan hakemisto, joka toimii päivityspalvelimen repositoriohakemistona. Esimerkin vuoksi valitaan repositoriohakemistoksi */var/deltagon.repo*. Tälle hakemistolle suoritetaan *createrepo /var/deltagon.repo* -komento, jolloin *createrepo* luo kaikista tästä hakemistosta löytyneistä RPM-paketeista XML-muotoisen *repomd* repositorion. *Repomd* sisältää tiedot kolmesta pakatusta XML-muotoisesta tiedostosta, joihin on sisällytetty repositorion tarjoamien RPM-pakettien otsikkotietueet. Pakatut XML-muotoiset tiedostot ovat nimeltään *filelists.xml.gz*, *primary.xml.gz* ja *other.xml.gz*. *Filelists.xml*-tiedosto sisältää tekstimuodossa jokaisen paketin sisältämien tiedostojen tiedostopolun sekä pakettien versiotiedot. Repositorion hakemistohierarkiaksi syntyy */var/deltagon.repo/repodata*, jossa RPM-tiedostot tallennetaan */var/deltagon.repo*-hakemistoon ja repositorion XML-muotoiset tiedot *repodata*-hakemistoon.

Jotta luotuun repositorioon voidaan yhdistää ja suorittaa kyselyitä, tarvitaan rajapinta yhteyksien luomiselle. Deltagon repositorion rajapinnaksi valittiin HTTPS-rajapinta jonka toiminnallisuus toteutetaan apache-ohjelmalla. Apachen oletusasetuksissa on asetuksia, joita ei tarvita päivityspalvelimen toiminnallisuuden kannalta. Näihin liittyy muun muassa oletushakemistoja ja oletussivuja sekä hakemistojen oikeusasetuksia. Nämä asetukset poistetaan käytöstä jotta järjestelmään ei jää turhaa toiminnallisuutta, johon ulkopuoliset käyttäjät voisivat mahdollisesti päästä käsiksi.

Verkkosivustojen asetukset määritellään apachen asetustiedostoihin ns. VirtualHosteiksi. Koska päivityspalvelimen toiminnallisuus tahdotaan suojata ja hallinnoida SSL-yhteyksillä, asetustiedostona käytetään apachen *conf.d*-hakemistossa sijaitsevaa *ssl.conf*-asetustiedostoa. VirtualHostille määritellään IP-osoite ja portti, josta sivustoon päästään yhdistämään. Sivustolle määritellään myös hakemistopolku ServerRoot, josta sivustolla näytettävät tiedostot ja sivut haetaan.

```
root@centos-ks-s1:/etc/httpd/conf.d
<VirtualHost 192.168.1.122:443>
ServerName update.deltasson.com
DocumentRoot /var/deltagon.repo/
<Directory "/var/deltagon.repo">
        Order Deny,Allow
        Deny from all
        Allow from 192.168.1.81
</Directory>
SSLEngine on
SSLCertificateFile /etc/httpd/conf/ssl.crt/deltagon.repo.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/deltagon.repo.
key

ErrorLog /var/log/httpd/deltagon.repo/ssl_error_log
CustomLog /var/log/httpd/deltagon.repo/access_log common
CustomLog /var/log/httpd/deltagon.repo/agent_log agent
CustomLog /var/log/httpd/deltagon.repo/referer_log referer
TransferLog /var/log/httpd/deltagon.repo/ssl_transfer_log

</VirtualHost>
```

Kuva 5. VirtualHostin määrittely apachessa

Kuvassa 5 on esimerkki apachen *ssl.conf* -asetustiedostoon määritellystä VirtualHostista. ServerName-muuttuja määrittelee sivun FQDN-osoitteen (Fully Qualified Domain Name), jonka nimipalvelimet tulkitsevat päivityspalvelimen IP-osoitteeksi, eli VirtualHostin määrittelyssä esitetyksi osoitteeksi. DocumentRoot määrittelee palvelimen polun, josta sivun sisältö haetaan. Yum-repositorion tapauksessa riittää, että polku osoittaa yum-repositorion paketit sisältävään hakemistoon. Yum-asiakasohjelma etsii *repodata*-hakemiston ja sen sisällön tästä juuresta. Directory-lohkossa määritellään *mod_ssl*-moduulin tarjoaman yhteyksienhallintatyökalun asetukset. Määrittely tapahtuu Directory-lohkossa, sillä yhteyksienhallintatyökalu toimii hakemistokohtaisesti, eli saman sivuston eri hakemistoille voidaan määritellä eri oikeuksia. Order-rivillä määritellään, missä järjestyksessä määriteltyjä asetuksia tulkitaan. Deny,Allow-järjestyksellä ensin muistiin luetaan kaikki ÒDeny fromÒ-rivit, ja mikäli yhteyden muodostamista yrittävä taho löytyy joltakin näistä riveistä, pyyntö hylätään paitsi, jos yhdistäjä löytyy myös joltakin ÒAllow fromÒ-riviltä. Allow,Deny-järjestyksessä tutkitaan ensin kaikki ÒAllow fromÒ-rivit, ja mikäli yhdistäjää vastaavaa sääntöä ei löydetä, pyyntö hylätään. Tämän jälkeen tutkitaan vielä kaikki ÒDeny fromÒ-rivit, ja mikäli vastaava sääntö löydetään, pyyntö hylätään. Deltagon D3 -päivitysjärjestelmän yhteyksiä tahdotaan rajata toimimaan siten, että kaikkien paitsi sallittujen yhdistäjäosoitteiden pyynnöt hylätään. Tämä voidaan toteuttaa kummallakin järjestyksellä. Kuvan 5 esimerkissä on käytetty De-

ny,Allow-järjestystä, jolloin tarvitaan ÒDeny from allÒ-säännön määrittely. Mikäli esimerkissä käytettäisiin Allow,Deny-järjestystä, tämä rivi voitaisiin jättää pois vaikuttamatta lopputulokseen.

SSL protokollan käyttäminen vaatii, että VirtualHostille on määritelty SSL-sertifikaatti sekä yksityinen avain, jota käytetään datan kryptauksessa. Sertifikaatti määritellään SSLCertificateFile-muuttujalla ja yksityinen avain SSLCertificateKeyFile-muuttujalla. Näitä tiedostoja käytetään SSL-yhteyksien suojaamiseen.

Saapuvia yhteyksiä voidaan myös rajata vaatimalla, että kyselijätaho esittää oman julkisen sertifikaattinsa yhteyttä muodostaessa. Tarjottu sertifikaatti pitää löytyä palvelimen luotettujen sertifikaattien tai sertifikaattimyöntäjien joukosta. Tämän asetuksen nimi on SSLVerifyClient, arvolla ÒrequireÒ. Tämän kylkiäisasetusarvo on SSLVerifyDepth, jolla määritellään, kuinka monta askelta sertifikaatin myöntäjäketjua verifioidaan.

7.3.3 Päivitysraporttien vastaanottopalvelu

Päivityspalvelimelle lähetettävät päivitysraportit vastaanotetaan perl CGI (Common Gateway Interface)-skriptillä toteutetulla HTTPS-palvelulla, joka asennetaan *apache*en päivityspakettirepositorion rinnalle. Skriptitiedosto vastaanottaa POST-kyselyjä sallituista osoitteista ja käsittelee ennalta määritellyt parametrit, joiden perusteella se tallentaa lähetetyn raportin levylle ja tiedot tallennetusta tiedostosta järjestelmän tietokantaan. POST kyselyjä käsiteltäessä saadut syötteet tarkistetaan erittäin tarkkaan ja pyyntö hylätään, mikäli pyynnön tekijän kaikki tarjoamat tiedot eivät vastaa odotettuja arvoja.

7.3.4 Järjestelmän tietokanta

Päivitysjärjestelmän täytyy ylläpitää päivityksiä kyselevien ja tarvitsevien asiakaspalvelinten yhteystietoja sekä käyttöoikeuksia. Nämä tiedot tallennetaan MySQL-relaatiotietokantaan. Tietokannassa ylläpidetään myös tapahtumalokia päivityspalvelimelle lähetetyistä asiakaspalvelinten päivityslokeista ja mahdollisista virhetilanteista.

Tietokantana käytetään päivitysjärjestelmälle omistettua INNODB-tyyppistä relaatiokantaa nimeltään updatesystem. Kanta koostuu kolmesta taulusta, joista kaksi on kiinteää tietoa sisältäviä tauluja ja kolmas näitä tauluja hyväksi käyttävä relaatiotaulu.

Taulukko 2. *clients*-taulun kuvaus

Sarakkeen nimi	**Tiedoston tyyppi**	**Kuvaus**
id	Kokonaisluku	Pääavain, juokseva numero. Käytetään viittaamaa haluttuun tietueeseen taulussa
name	Merkkijono	Palvelimen omistaman asiakasyrityksen nimi
ip	Merkkijono	Asiakaspalvelimen IP osoite
permissions	Merkkijono	Palvelimen omaamat päivitysoikeudet
version	Merkkijono	Asiakaspalvelimelle asennetun D3 ohjelmiston versio
contact_email	Merkkijono	Asiakasyrityksen yhteyshenkilön sähköpostiosoite

Taulukossa 2 kuvattu clients-taulu sisältää järjestelmän tuntemien asiakaspalvelimien yhteystiedot. Näitä tietoja käytetään järjestelmän web-hallinnointityökalussa, web-rajapintaa käyttävien yhteyksien oikeuksien tarkistuksissa sekä julkaisuilmoituksissa.

Taulukko 3. *issues*-taulun kuvaus

Sarakkeen nimi	**Tiedoston tyyppi**	**Kuvaus**
id	Kokonaisluku	Pääavain, juokseva kokonaisluku. Käytetään viittaamaan haluttuun tietueeseen taulussa
definition	Merkkijono	Tapahtumaa kuvaava nimi
severity	Kokonaisluku	Numeerinen esitys tapahtuman vakavuudesta

Issues-taulu, joka on kuvattu taulukossa 3, sisältää järjestelmän tuntemat ilmoitustyypit, joita asiakaskoneet lähettävät päivityspalvelimelle. Definition-kenttä on ihmisille luettavissa oleva kuvaus tapahtumasta. Kuvaus näytetään ylläpitäjille järjestelmän web-

hallintatyökalussa, jotta asiakaspalvelinten lähettämien raporttien sisällöistä voidaan nähdä mitä raportti koskee ilman, että raportin sisältöä on tarpeellista lukea tai edes avata. Severity-kentällä voidaan lisätä eri raporttien näkyvyyttä hallinnointityökalun käyttöliittymässä värjäämällä tietyn vakavuusasteen ylittävät tapahtumat huomiota herättävällä värillä. Tietyn vakavuusasteen ylittävistä tapahtumista voidaan myös ilmoittaa ylläpitäjille suoraan sähköpostitse tai muuten käytössä olevin keinoin ilman, että ylläpitäjän tarvitsee tarkkailla jatkuvasti palvelinten tapahtumia hallinnointityökalun avulla.

Taulukko 4. *logs*-taulun kuvaus

Sarakkeen nimi	**Tiedoston tyyppi**	**Kuvaus**
id	Kokonaisluku	Pääavain, juokseva kokonaisluku
client	Kokonaisluku	Viiteavain *clients*-taulun *id*-kenttään
issue_id	Kokonaisluku	Viiteavain *issues*-taulun *id*-kenttään
date	Aika	Tapahtuman ilmoitusaika. Haetaan automaattisesti käyttöjärjestelmältä kun tapahtumaa syötetään kantaan
logfile	Merkkijono	Palvelimelle tallennetun lokitiedoston nimi

Taulukossa 4 kuvattu *logs*-taulu on asiakaspalvelimien tuottamien päivitysraporttien ja ilmoitusten tallennuspaikka. Raportteja tallennettaessa tarkistetaan raportin syöttäjän IP-osoite, joka yhdistetään asiakkaan IP-osoitteeseen. Näin lähetetty raportti voidaan tallentaa siten, että se voidaan yhdistää sen lähettäneeseen asiakaspalvelimeen. *issue_id*-kenttää käytetään kuin *client*-kenttää, mutta raporttia lähettävä palvelin ilmoittaa raporttia lähettäessään siihen liitettävän tapahtuman tyypin, joka voidaan ilmoittaa kannan id numeron tai kuvaavan tekstin perusteella, jolloin *id*-numero voidaan päätellä tekstin sisällöstä.

logfile-kenttään tallennetaan levylle tallennetun raporttitiedoston nimi, jonka tallennushakemiston ohjelmakoodi tietää suorituksessaan. Pelkän tiedostonimen tallennus sallii erilaisia ratkaisuja tallennushakemistolle. Oletuksena lokitiedostot tallennetaan kaikki samaan hakemistoon *(/var/log/updatesystem*) mutta järjestelmä voidaan myös muuttaa käyttämään asiakaskohtaisia lokihakemistoja, jotka voidaan tallentaa esimerkiksi *clients*-tauluun. *logs*-taulun tarkoitus on kuitenkin tallentaa yksittäisiä raportteja ja tapahtumia, joten tiedoston nimi on molemmissa tapauksissa ainoa tarpeellinen tieto tiedostosta.

7.3.5 Järjestelmän asetusten ylläpito

Järjestelmän tarvitsemia palomuurisääntöjä sekä muita tietoturvan kannalta oleellisia asetuksia valvotaan ja varmennetaan tasaisin väliajoin käyttöjärjestelmän ajastinta (*crontab*) hyödyntäen. Tarkastukset suorittava perl-skriptitiedosto asetetaan suoritettavaksi tunneittain tai useammin.

Skriptitiedosto suorittaa MySQL-kyselyn tietokannan clients-taululle, jolla se noutaa kunkin järjestelmän tunteman asiakaspalvelimen IP-osoitteen ja sille kuuluvat oikeudet. Tämän jälkeen skripti lukee järjestelmän palomuurin asetustiedoston ja etsii sieltä löydettyjen IP-osoitteiden sääntörivit ja varmistaa niiden eheyden. Mikäli tiedostosta löytyy rivejä, joissa asiakaspalvelimelle on määritelty sääntö, joka ei vastaa sille myönnettyjä oikeuksia, rivit muutetaan vastaamaan kannassa määriteltyjä oikeuksia. Palomuurin asetustiedostosta löydetyt rivit, jotka eivät vastaa mitään tunnettua asiakas-palvelinta tai tunnettua oletussääntöä poistetaan tiedostosta. Vaihtoehtoisesti rivien muuttamisen ja poistamisen sijaan voidaan nostaa virhetilalippu, jonka olemassaolon perusteella voidaan reagoida ja ilmoittaa palvelimen ylläpitäjälle virhetilanteesta.

Palomuurinsääntöjen eheystarkistuksen jälkeen vastaavat tarkistukset suoritetaan *apache*-ohjelman *ssl.conf*-tiedostolle, johon on määritelty yum-repositorion web-rajapinnan asetukset. Asetuksista löytyvät IP-osoitteiden rajaukset varmennetaan saman kaavan mukaan kuin palomuurisäännötkin. *Ssl.conf*-tiedoston sisältö on formaatiltaan paljon yksinkertaisempaa kuin palomuurisääntötiedoston, joten sen muokkaus ja tarkistus on vastaavasti yksinkertaisempaa. Ainoa monimutkaisuutta tuova muuttuja on käytössä oleva sääntöjen esitys- ja tarkistusjärjestys. Koska järjestyksiä on olemassa kiinteä määrä, niille voidaan tehdä erilliset käsittelytapansa

Tämä skriptitiedosto toimii web-hallinnointityökalun toimintojen varmentajana. Oletus on, että web-hallinnointityökalulla lisätyt tai muokatut oikeudet kullekin asiakasosoitteelle on muokattu myös palomuurisääntöihin sekä *apache*-ohjelman asetustiedostoihin. Skripti varmistaa, että tietokannan tiedot vastaavat aina järjestelmän oikeita asetuksia ja korjaa mahdollisia käsin tehtyjä muutoksia.

7.3.6 Järjestelmän web-hallinnointityökalu

Web-hallinnointityökaluun kuuluu kaikki oleellinen asiakaspalvelinten tilojen valvomiseen ja saatavilla olevien päivityspakettien ylläpito. Hallinnointityökalussa on useita näkymiä, joilla valvotaan palvelinten tilannetta. Päänäkymässä on listattuna kaikki palvelimen tuntemat asiakaspalvelimet ja niiden yleistiedot ja linkki palvelimen tapahtumalokiin. Tällä sivulla ei ole muuta oikeaa toiminnallisuutta kuin tilanteen yleisnäkymän tarjoaminen.

Tapahtumalokinäkymään päästään päänäkymältä palvelinkohtaisesta tapahtumalokilinkistä. Tapahtumaloki populoidaan palvelimen id:n perusteella palvelimen *logs*-tietokantataulusta. Sivulla näytetään lista raportoiduista tapahtumista niiden tapahtumisjärjestyksessä ja siten, että tapahtumaan liittyvä *issue_id* on näkyvillä luettavassa muodossa. Sivulla näytetään myös lokitiedoston ensimmäinen rivi, jota painamalla avautuu lokinäkymä.

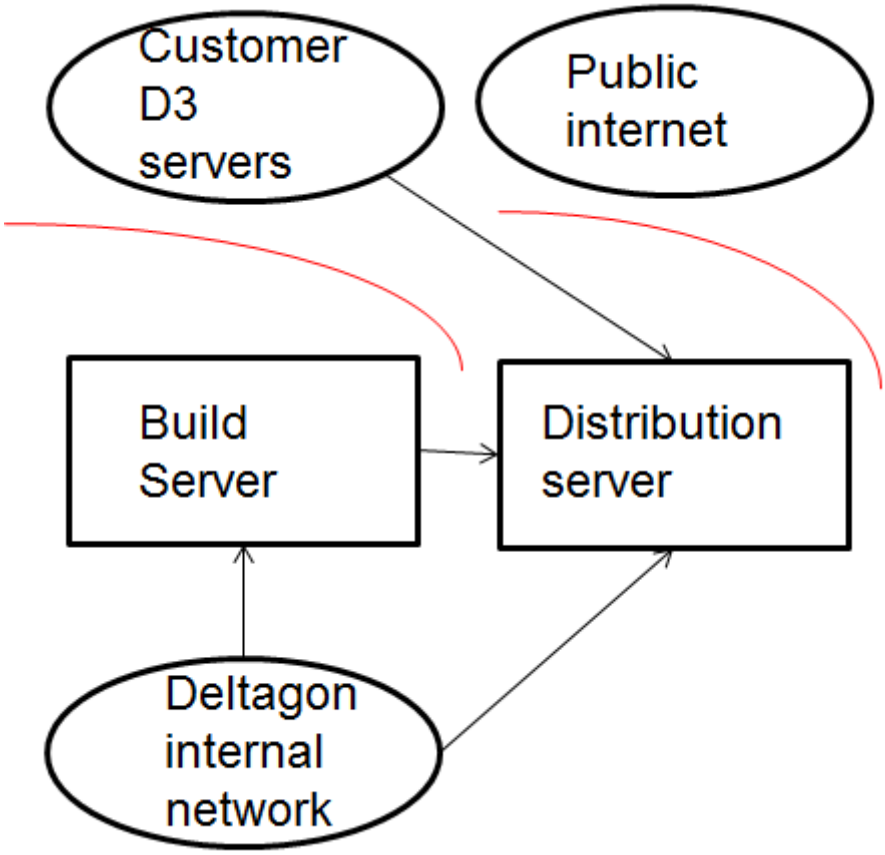
Lokinäkymässä näytetään koko sivulla tapahtumaan liittyvä loki, joka raportin yhteydessä on jätetty. Tämä on tulostettu suoraan lokitiedostosta näkyville erikoismerkkienkoodauksen kautta.

Lokeista ja palvelimista erillisenä näkymänä on uusien pakettien lisääminen palvelimen repositorioon. Sivulla näytetään paketointikoneella saatavilla olevat paketit, joista voidaan valita jokin tuotavaksi julkaisuun. Paketeilla saattaa olla erikoistoimintoa vaativia lisäominaisuuksia kuten alusta, joka pyritään käsittelemään automaattisesti paketin metatietojen perusteella. Paketti tuodaan kasauspalvelimelta joko ssh-komennolla tai tätä käyttötarkoitusta varten tehdyllä CGI-skriptillä, joka sallii yhteyden julkaisupalvelimelta ja kuljettaa paketin tiedot HTTPS:llä palvelimelta toiselle.

Web-hallinnointityökalu toteutetaan Perl-skriptillä, jotka on jaoteltu toiminnallisuuksien ja näkymien mukaan moduuleiksi. Kukin moduuli toteuttaa oman nimiavaruutensa itsenäisesti, joita hallitseva CGI-skripti kutsuu tarpeen mukaan. Moduuleja kutsutaan käyttäjän valitseman näkymän perusteella. Kukin moduuli toteuttaa sisällöntuotantofunktion, jolle välitetään käyttäjän valintaan liittyvät muuttujat kuten asiakkaan tai lokin tunniste. Sisällöntuotantofunktio palauttaa muuttujapseudo-olion sisällä moduulille oleelliset tiedot, jotka piirretään html-raameihin hallitsevassa CGI-skriptissä.

7.3.7 Päivityspalvelimen ja kokoamispalvelimen paikka Deltagon Group Oy:n verkossa

Päivityspalvelin sijoitetaan Deltagon Group Oy:n verkon DMZ (Demilitarized Zone) alueeseen, johon päästään myös yrityksen sisäverkon ulkopuolelta. Alue ei kuitenkaan ole täysin julkinen vaan ulkopuolisten sisäänpääsy on rajattu IP-osoitteiden ja tunnustautumismenetelmien perusteella.



Kuva 6. Hahmotelma päivitysjärjestelmään liittyvistä verkkokomponenteista

Kuvassa 6 on esitetty pelkistetty hahmotelma päivityspalvelimeen liittyvistä verkkokomponenteista. Kuvion nuolet ilmaisevat sallittujen yhteyksien luomisia. Kuten kuviosta käy ilmi, kokoamispalvelin (build server) on eristetty kaikista ulkoverkon yhteyksistä ja on täten osa Deltagon Group Oy:n sisäverkkoa (deltagon internal network) johon kuuluu esimerkiksi toimistotilojen työpöytäkoneet sekä kaikki ohjelmiston testauspalvelimet. Päivitys- tai jakelupalvelin (distribution server) sallii yhteyksiä ulkoverkosta, mutta vain tunnetuista Deltagon Group Oy:n asiakaspalvelimista (customer D3 servers). Päivityspalvelimelta ei myöskään ole tarvetta yhteyksien muodostamiselle takaisin sisäverkkoon, sillä kaikki palvelimelta tarvittava tieto saadaan web-hallinnointityökalun tai

suoran SSH shell -yhteyden avulla. Näin vältytään myös mahdollisilta tietoturvariskeiltä joissa päivityspalvelimen altistuttua myös sisäverkon turvallisuus olisi vaarassa. Päivitysjärjestelmän lisäkomponenteissa voidaan tarvita ylimääräisiä yhteydenluontimahdollisuuksia tietyille tunnetuille palvelimille, kuten päivitysilmoitusta lähetettäessä Deltagon Group Oy:n omalle sähköpostipalvelimelle.

8 Asiakaskone

8.1 Yleistä

Järjestelmää käyttävät asiakaspalvelinkoneet ovat pääasiassa Deltagon Group Oy:n asiakkaiden omistamia palvelimia, joille on asennettu Deltagon D3 -sähköpostin salausjärjestelmä. Asiakaskoneet pitävät Deltagon D3 -järjestelmänsä sekä sen vaatimat ohjelmistot ajan tasalla siihen asennetuilla paketinhallintaohjelmistoilla ja Deltagonin oman päivitysten hallintajärjestelmän avulla.

Asiakaspalvelinkoneiden tulee pystyä suoriutumaan eriasteisista toiminnoista päivitysjärjestelmän eri toteutusvaiheissa. Perustoiminnallisuus vaatii päivityspakettien asentamisen rpm-paketista ja pakettien noutamisesta Deltagon-päivityspalvelimelta. Tarvittaessa päivityspaketit tulee pystyä tuomaan palvelimen kiintolevylle muinkin keinoin, kuten käsin siirtämällä. Asiakaspalvelimen tulee myös pitää yllä lokia päivitykseen liittyvistä toiminnoista ja päivitysyrityksistä. Myöhemmissä vaiheissa asiakaspalvelimen tulee pystyä lähettämään edellä mainittuja lokeja päivityspalvelimelle raporttien muodossa käyttäen päivityspalvelimen tarjoamaa raportointirajapintaa.

8.2 Yum-ohjelman asetukset

Deltagon D3 -ohjelmiston päivittäminen toteutetaan Yum-ohjelmalla Deltagon Group Oy:n omasta repositoriosta, joten ohjelma täytyy asettaa käyttämään oikeaa repositoriota. Lisäksi ohjelma asetetaan käyttämään oletuksesta poikkeavia asetuksia joihinkin toimintoihin liittyen.

Yum-ohjelman asetustiedosto löytyy oletuksena hakemistopolusta */etc/yum.conf*. Ohjelman käyttämät pakettirepositoriot voidaan luetella tässä tiedostossa tai erillisissä

tiedostoissa */etc/yum.repos.d/*-hakemistossa. Repositorio ÒDeltagonÒ asetetaan tiedostoon Deltagon.repo ja sen sisällöksi asetetaan seuraavat rivit:

```
[Deltagon]
name=Deltagon RPM Repository
baseurl=https://192.168.1.122/
```

Repositorioon viitataan ohjelmaa käytettäessä hakasulkujen sisään kirjoitetulla tunnisteella. Baseurl-rivi kertoo ohjelmalle, mistä repositorion metatiedot ja itse paketit noudetaan. Tämä asetetaan siihen osoitteeseen ja porttiin, johon päivityspalvelimen VirtualHost määriteltiin apache-ohjelman konfiguraatioissa (kuva 5). Yum-ohjelman omaan konfiguraatiotiedostoon lisätään myös seuraavat oletuksesta poikkeavat muuttujat arvoineen:

```
gpgcheck=0
plugins=1
showdupesfromrepos=1
tsflags=repackage
```

Oletuksena yksi Yum-ohjelman käyttämistä tunnistamistavoista on GPG-avain (GNU Privacy Guard), jota Deltagon D3 -palvelimissa ei käytetä. Yum-ohjelman haut epäonnistuvat aina GPG-tarkistukseen, mikäli toiminnallisuutta ei käytetä, joten tarkistus kytketään pois päältä asettamalla asetusarvo gpgcheck nollaan. Plugins-optiolla sallitaan erikseen asennettavien lisätoimintojen käyttö. Deltagon D3 -palvelimelle asennetaan joitakin lisätoimintoja yum-ohjelmalle, jotta lataus- ja asennustoimintoja, joten tämä optio täytyy asettaa päälle. *Showdupesfromrepos*-optiolla repositorion metatiedoista luetellaan kaikki sieltä löytyvät paketit eikä vain asentamattomia ja kaikkein uusimpia versioita. Tämä on tärkeää sillä Deltagon D3 -ohjelmiston päivitysten yhteydessä saatetaan tehdä suuriakin muutoksia esimerkiksi MySQL-tietokantaan. Nämä muutokset tehdään aina vain sen version päivityspaketissa, joten mikäli päivitys tahdotaan tehdä vanhemmasta versiosta, on tarpeellista asentaa jokainen välipäivitys sillä hetkellä asennettuna olevan version ja uusimman tuotantoversion välillä. *Tsflags* optiolla välitetään parametreja paketin asentavalle RPM-ohjelmalle. *Repackage*-parametri käskee RPM:n pakata päivityksen yhteydessä sillä hetkellä asennettuna oleva versio varmuuskopioksi talteen. Tällöin voidaan suorittaa niin sanottu rollback- tai palautustoimenpide, jolla perutaan kaikki RPM-toimenpiteet annettuun ajankohtaan asti ja palautetaan senhetkinen asennettujen pakettien tilanne. Tälle toiminnallisuudelle voi olla tarvetta hätätapauksissa, mikäli uudessa versiossa huomataan kriittinen virhe, joka estää koko järjestelmää toimimasta oikein.

yum-ohjelmalle välitetään kutsun yhteydessä aina myös ò-disablerepo=* --enablerepo=òDeltagonò. Näin toimitaan aina vain Deltagon.repossa määritellyllä päivityspalvelimella, eikä esimerkiksi list komennon yhteydessä tarvitse karsia pois muiden julkisten repositorioiden tarjoamia tiedostoja. Koodissa òDeltagonò repositorion nimi luetaan palvelimelle tallennetusta päivitysasetustiedostosta.

Asennuksen yhteydessä ei tarvita repositoriotietoja, vaan asennus tehdään yumin ðocalinstallò komennolla. Oletuksena yum lataa ja asentaa paketit samalla kertaa, mutta Deltagon D3 -palvelimille asennettu *downloadonly* yum liitännäinen keskeyttää tavallisen toiminnon latauksen jälkeen ja tallentaa RPM-paketin määriteltyyn hakemistoon. Asennuksessa käytettävä ðocalinstallò-komento lukee asennettavan paketin levyltä parametrina annetusta polusta. Päivityskirjaston asetustiedostossa käytettyä pakettipolkua käytetään *downloadonly*-liitännäisen tallennushakemistona ja asennuskomennon polkuna.

8.3 Päivitysjärjestelmän asetustiedosto

Asiakaspalvelimen päivitysjärjestelmän käyttämät hakemistopolut, reposito-rion nimi ja automaattipäivityksen asetukset ovat muutettavissa D3-järjestelmän päivitysasetustiedostossa. Tämä tiedosto on pääasiassa tarkoitettu muokattavaksi asennuksen yhteydessä, mikäli oletusasetuksista halutaan poiketa. Tiedoston sisältö on yleisesti käytössä oleva òavain=arvoò muodossa. Asetuksiin voidaan muuttaa polku, mihin haetut päivitystiedostot tallennetaan. Sillä D3-palvelimet ovat tehtävälleen omistettuja palvelimia, niissä on harvoin tarvetta muuttaa oletuspolkuja. Joskus palvelinten levyosiointi voi olla toteutettu normaalista poiketen, jolloin paketit ja lokitiedostot voidaan haluta tallentaa eri polkuun.

Asetustiedostossa on myös mahdollista kytkeä päälle automaattinen asennus päivityspaketteja ladatessa. Oletuksena paketti vain haetaan ja täytyy asentaa erillisellä komennolla tai napin painalluksella. Oletustoiminta sallii esimerkiksi pakettien lataamisen palvelimelle, josta ollaan päivityksen vuoksi katkaisemassa internetyhteys.

D3-järjestelmän automaattinen ajan tasalla pito on myös mahdollista, mutta harvoin suositeltavaa. Sillä D3 on korkean saatavuuden järjestelmä, on tärkeää, että palvelu ei

yllättäen katkea, mikäli päivityksessä tapahtuu virhetilanne. Kun päivitykset tehdään valvonnan alla, mahdollisiin virheisiin voidaan reagoida heti.

8.4 Päivitysjärjestelmän kirjasto

Päivitysjärjestelmän toiminnallisuudet tehdään perl-moduulina, joka toteuttaa tarvittavat järjestelmäkutsut ja muutokset silloin, kun sen funktioita kutsutaan. Kirjasto toteutetaan siten, että sen palveluja voidaan käyttää sekä web-puolen kutsuissa että automatisoiduissa päivitysskriptoissa. Käytännössä tämä tarkoittaa sitä, että funktiot eivät ikinä palauta suoria tulosteita vaan tiedot, joita kutsuva osapuoli tarvitsee jatkaakseen.

8.4.1 Kirjaston perustoiminnot

Kirjasto lukee alustuksessa oman asetustiedostonsa ja asettaa sieltä löytyneet arvot sisäiseen hajautustaulureferenssiin. Moduuli tarjoaa *new*-funktion, joka palauttaa *bless*-ydinfunktiolla alustetun hajautustaulureferenssin. *New*-funktiossa kutsutaan asetustiedoston lukufunktiota.

```
package D3::Update;
sub new {
    my ($class, %opts) = @_;
    my $self = bless {}, $class;
    $self->load_configuration($opts{configuration_file}) or return;
    return $self;
}
sub load_configuration {
    my ($self, $file) = @_;
    $file = ñ/etc/d3/update.cfgò unless ($file && -f $file);
    open(my $handle, ñ<ò, $file) or return;
    while( <$handle> ) {
        my ($key, $value) = $_ =~ /^([^=]+)=(.*)$/;
        $self->{configuration}->{$key} = $value;
    }
    close($handle);
    return 1;
}
```

Avainarvoparien luvussa käytetty säännöllinen lause lukee avaimeksi kaikki merkit rivin alusta, jotka eivät ole merkki ñ=ò Arvoksi luetaan kaikki ò=ò merkkiä seuraavat merkit, välilyönnit mukaan lukien.

Kirjasto tarjoaa funktiot *list, download* ja *install.* Ne toteuttavat IPC::Run moduulilla yum-ohjelman kutsut, käsittelevät sen tulosteet ja palauttavat kerätyt tiedot jäsennellyissä hajautustaulureferensseissä. IPC::Runin toimintaa selvennetään kohdassa 8.4.2. Tämän osion esimerkeissä muuttujat $out ja $error ovat IPC::Runin asettamia muuttujia.

.*List* funktiossa *yum*ia kutsutaan ò*list D3*ò parametreilla, ja sen tulosteen kukin rivi jaotellaan hajautustaulureferenssiin *package, version* ja *arch* alkioihin. Hajautustaulureferenssi syötetään sitten taulureferenssimuuttujaan loppuun. Taulureferenssi tallennetaan hajautustaulureferenssiin, johon tallennetaan myös kutsun paluuarvo.

```
sub list {
   my ($self, $package) = @_;
   IPC::Run::run()é
   my $return_href = { óreturn_valueô=> $?, entries => [], error => $error };
   foreach my $line (split(/\r?\n/, $out)) {
      my $entry = {};
      ($entry->{package}, $entry->{version}, $entry->{arch}) = split(/\s+/, $line);
      push(@{ $return_href->{entries} }, $entry);
   }
   $self->log($?, $out, $err};
   return $return_href;
}
```

Perlin *split*-komento jakaa ensimmäisenä annetun säännöllisen lauseen tai merkkijonoparametrin perusteella erotellun toisen merkkijonoparametrin, ja palauttaa ne järjestyksessä listamuuttujassa.

Download-funktio palauttaa vain paluuarvonsa, sillä kaikki tiedostojen käsittely tapahtuu *yum*-ohjelmalla. Se välittää tiedostojen lataushakemiston *yum*-komennon yhteydessä, ja on kutsujan vastuulla käsitellä lataushakemiston sisältöä halutulla tavalla. Sille välitetään halutun paketin nimi, versio ja alustan versio.

```
sub download {
   my ($self, $package, $version, $arch) = @_;
   IPC::Run::run()é .;
   $self->log($?, $out, $err);
   return { return_value => $?, error => $error };
}
```

Asennusfunktio *install* toimii kuin latausfunktio, mutta se kutsuu myös asennuksen onnistuttua kaikkien solujen päivitysfunktiota *update_nodes*, joka on tarkemmin kuvattu kohdassa 8.4.4.

8.4.2 IPC::Run-moduuli

IPC::Run on moduuli, jolla voidaan kutsua haluttua käyttöjärjestelmän tuntemaa ohjelmaa ja välittää annetut parametrit suoraan halutulle ohjelmalle. IPC (Inter-process Communication) on yleinen kirjastopohja työkaluille, joilla ohjelmaprosessit voivat keskustella keskenään. Perinteisesti Perlissä vastaavat kutsut tehdään ``` eli backtick-merkkien sisään kirjoitetuilla konsolikomennoilla. Vastaavasti voitaisiin myös kutsua ydinfunktioita system() tai exec(), jotka myös suorittavat konsolikomennon joka sille välitetään. Backtick-kutsuja pidetään erittäin vaarallisina varsinkin, jos kutsun mukana kulkee käyttäjältä saatua syötettä. Tällöin käyttäjäsyöte voi hajottaa kutsun tarkoitetun muodon ja syöttää haluamaansa syötettä suoraan konsoliin. Tämä voidaan estää joillakin eri tavoilla, kuten niin sanotulla ```safe pipe```llä, joka suorittaa kutsutun ohjelman käyttämättä konsolia. IPC::Run tekee tästä prosessista suoraviivaisempaa ja selkeämpää ja saa talteen kutsun tulosteet, virheet ja paluuarvon ilman, että niitä tarvitsee erikseen määritellä talteen otettaviksi. IPC::Run-kirjastoa voidaan käyttää ohjelman kanssa keskusteluun, mutta tässä työssä tarvitaan vain yksittäisiä komentoja, joiden tulosteet otetaan talteen. Esimerkiksi *list*-komento, jolla etsitään kaikkia D3-paketteja suoritetaan seuraavasti:

```
IPC::Run::run( [ ```/usr/bin/yum```, ```list```, ```D3```], \$in, \$out, \$err);
```

Muuttujat $in, $out ja $err julistetaan ennen kutsua, ja pitää välittää kutsulle oikeassa järjestyksessä muuttujaviitteinä. Kutsun paluuarvo asetetaan ympäristömuuttujaan $? kuten muissakin perlin järjestelmäkutsuissa. Kutsutun ohjelman tuloste asetetaan $out muuttujaan sellaisenaan, ja virheviestit $err-muuttujaan. Kutsuttava ohjelma ja sen parametrit välitetään taulukkoreferenssissä, kuten yllä olevassa esimerkissä. Esimerkiksi valitun paketin asennuskomento tapahtuu kokonaisuudessaan seuraavasti:

```
my $file = $self->{configuration}->{package_directory} . ```/```. $package;
IPC::Run::run([ ```/usr/bin/yum```, ```localinstall```, ```--disablerepo=*```, $file] ,
                \$in, \$out, \$err);
```

Vastaavasti tietyn paketin lataus tapahtuu latausfunktiossa saaduilla parametreilla:

```
IPC::Run::run([ò/usr/bin/yumò,
               òdownloadò,
               òð downloadonlyò,
               òð downloaddir=$self->{configuration}->{package_directory}ò,
               ñ-disablerepo=*ò,
               ñ-enablerepo=$self->{configuration}->{repository}ò,
               ñ$package-$version-$archò], \$in, \$out, \$err);
```

8.4.3 Kirjaston käyttö

Kirjasto ladataan kutsuvaan koodiin kutsumalla sen *new*-funktiota, joka palauttaa perl-hajautustaulun, jonka referenssiarvo on kirjaston nimi. Näin hajautustaulun kautta voidaan kutsua kirjaston tarjoamia funktioita. Kirjaston funktioita käytetään ja sen virheisiin reagoidaan seuraavan esimerkin mukaisesti:

```
use D3::Update;
my $update = D3::Update->new(configuration_file => ñ/etc/d3/update.cfgð);
my $result = $update->list;
if ($result->{return_value} != 0) { #react }
my @entries = @{$result->{entries}};
#print entries however
```

Esimerkissä $result-muuttuja on perlin hajautustaulukko, johon on tallennettu kutsun tuottamat tulokset niitä vastaaviin avainkenttiin. Lataus- ja asennusfunktioita käytetään saman kaavan mukaan.

8.4.4 Muiden solujen päivitys

Kirjasto tarjoaa myös palvelimen sisarusten päivittämistoiminnallisuuden. Asennuskomentoa ajettaessa palvelimen asetuksista luetaan järjestelmään kuuluvien palvelimien määrä ja osoitteet. Näiden tietojen perusteella asennettava paketti siirretään kaikille palvelimille ja kutsutaan kullakin palvelimella paikallista skriptitiedostoa, joka lataa päivityskirjaston ja kutsuu asennuskomentoa paikallisesti samoin tiedoin kuin aktiivipalvelimella. Rpm-paketit on tehty siten, että palvelimiin tehtävät muutokset, jotka tulee tehdä vain tietyillä soluilla, tehdään D3-järjestelmän omien tilatietojen perusteella. Koodikanta päivitetään kuitenkin jokaisella palvelimella. Kutsumalla päivityskirjastoa paikallisesti saadaan päivitysraportit tehtyä paikallisen palvelimen asetusten perusteella. Suoritettava funktio on toteutettu seuraavasti:

```
use D3::Config;
sub update_nodes {
   my ($self, $package) = @_; #package is full path
   my $d3config = D3::Config->new;
   my $node_returns = {};
   my $remoteuser = $self->{configuration}->{remote_user};
   foreach my $node (@{$d3config->{nodes}}) {
      next if ($d3config->{this_node} eq $node);
      IPC::Run::run([ñ/usr/bin/scpò, $package, ñ$remoteuser\@$node:$packageò]);
      my ($in, $out, $err);
      IPC::Run::run([ñ/usr/bin/sshò, òñ$remoteuser\@$nodeò,
ñ/usr/bin/d3_remote_updateò, $package], \$in, \$out, \$err);
      $node_returns->{$node} = $out;
   }
}
```

Kaikki palvelimet on asetettu tunnistamaan ja sallimaan yhteydet toisiltaan käyttöjärjestelmän omilla menetelmillä, jolloin käytetyt *scp*- ja *ssh*-yhteydet ovat käytettävissä paketin siirtoon ja asennusfunktion kutsuun.

8.5 Päivitysraportit

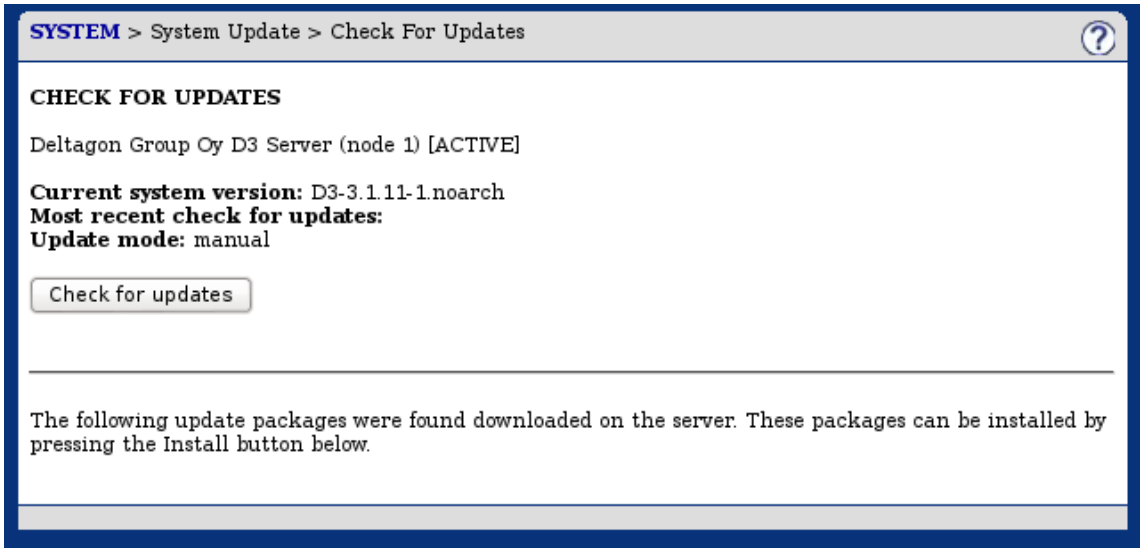
Päivitystapahtumista on tärkeää jäädä jälki, jota voidaan virhetilanteissa tutkia. Kaikista kirjastokutsuista luodaan lokitiedosto asetustiedostossa määriteltyyn polkuun. Lokitiedostoon tallennetaan päivämäärä, kaikki suoritetut komennot, niiden raa'at tulosteet ja virheilmoitukset. Tiedosto nimetään päivämäärän, kellonajan (kuluneina sekunteina 1.1.1970 00:00:00 lähtien, tunnetaan myös *unixtime*nä), tapahtuman tyypin ja nousevan tunnisteen perusteella. Näiden tietojen perusteella lokeja voidaan järjestellä niitä lukevissa komponenteissa.

Päivitysraportit yritetään myös lähettää päivityspalvelimelle sen tarjoaman raportinlähetysrajapintaan. Lähetys toteutetaan perlin LWP::UserAgent moduulin http-kyselytoiminnoilla. Raporttia lähetettäessä lähetetään tapahtuman lokitiedoston sisältö, tapahtuman tyyppi ja palvelimen tunnistavat tiedot kuten sen sertifikaatti.

8.6 Paketin asennuskäyttöliittymä

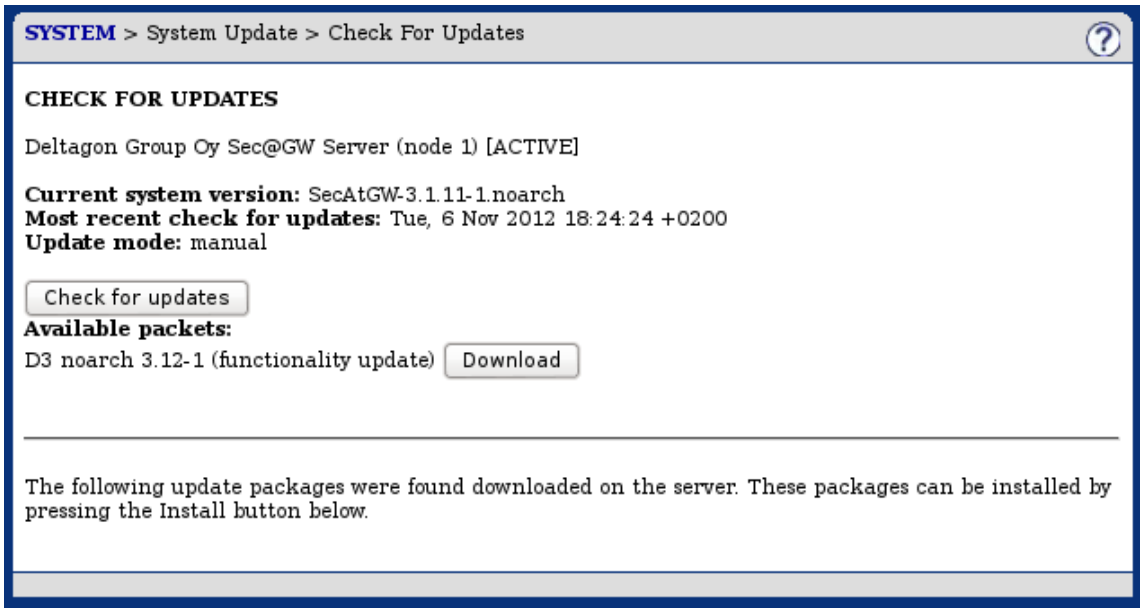
Päivityspaketti asennetaan palvelimelle Deltagon Group Oy:n D3-ratkaisun oman hallintapaneelin kautta. Hallintapaneelissa palvelimen ylläpitäjätason käyttäjillä on pääsy päivitystyökaluun. Päivitystyökaluun kuuluu päänäkymä, paketin käsintuonti järjestel-

mään ja päivitysloki. Päänäkymä koostuu kahdesta osiosta: saatavilla olevien pakettien lista ja asennettavissa olevien pakettien lista. Asiakasympäristöt voivat olla niinkin tiukasti suljettuja, ettei päivityspalvelimelle päästä HTTPS-yhteydellä. Tällöin ylläpitäjä voi tuoda palvelimelle asennettavissa olevien pakettien listaan paketteja jotka on lähetetty esimerkiksi suojattuna sähköpostina suoraan ylläpitäjälle. Tavallisessa käytössä kuitenkin saatavilla olevien pakettien listan voi päivittää nappia painamalla. Tällöin kutsutaan päivitysjärjestelmän kirjaston funktiota, joka lataa päivityspalvelimelta kaikki saatavilla olevat paketit ja palauttaa ne listana käyttöliittymälle.



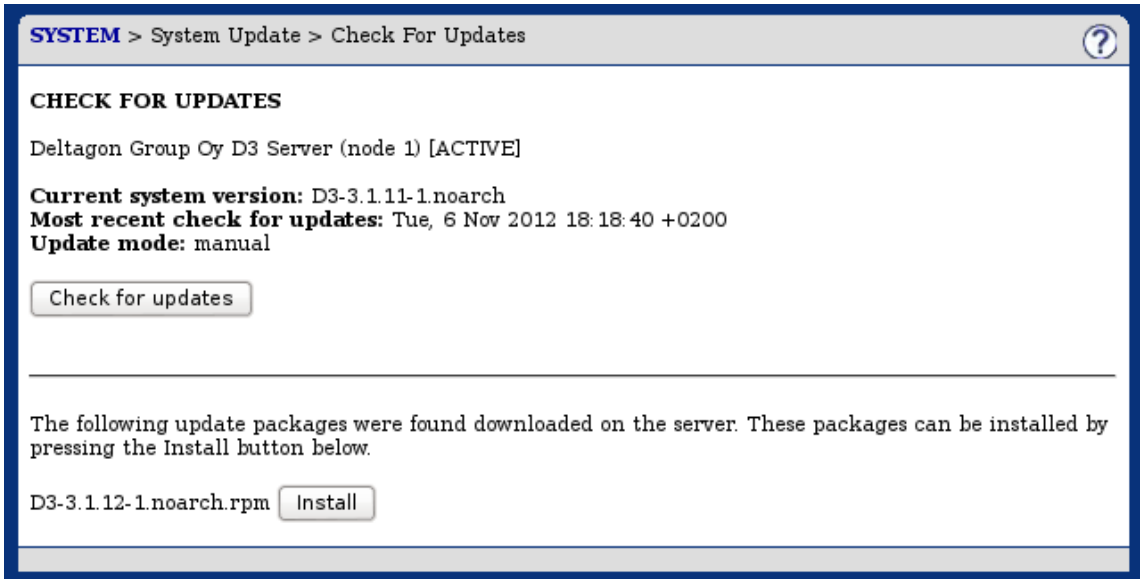
Kuva 7. Päivitystyökalun pääsivu

Listan päivityksen jälkeen kunkin paketin voi ladata asennettavaksi listalla olevasta pakettia vastaavasta latausnapista. Tämä kutsuu päivitysjärjestelmän kirjaston funktiota joka lataa tietyn nimisen ja versioisen paketin palvelimelle asennettavissa olevien pakettien hakemistoon. Kun paketti on haettu, sen voi asentaa painamalla asennusnappia asennettavissa olevien pakettien listalta.



Kuva 8. Päivitystyökalu kun lista on päivitetty

Asennettavien pakettien listalta voidaan asentaa jokin tietty paketti painamalla sitä vastaavaa asennusnappia. Tämä kutsuu päivitysjärjestelmän kirjaston funktiota asentaa paketti sekä aktiivisella pääpalvelimella että kaikilla mahdollisilla sisarsoluilla.



Kuva 9. Päivitystyökalu kun paketti on päivitettävissä

Kaikissa toiminnoissa on mahdollista, että syntyy virhetilanteita. Palomuuri voi olla asetettu niin, ettei latauslistaa voida päivittää, jolloin käyttöliittymässä näytetään aikakatkaisun virheilmoitus. Latauksessa itsessään voi tapahtua virhe, ja esimerkiksi katketa

yhteys repositorioon. Koska lataukset tehdään yum-ohjelmaa käyttäen, voidaan sen palauttamilla virhekoodeilla ja tulosteilla koota raportteja. Päivityskirjaston tuottamat raportit on luettavissa päivityslokinäkymässä. Näillä raporteilla populoidaan myös päivityspalvelimen palvelinkohtainen tapahtumaloki. Tiukasti suojatuista asiakaskoneista näitä ei tietenkään saada muualta kuin ylläpitoliittymästä. Virhetilanteisiin reagoidaan käyttöliittymässä palautetun yum-paluuarvon perusteella.

8.7 Automaattinen päivitys

D3-järjestelmä käyttää käyttöjärjestelmän cron-ohjelmistoa suorittamaan ajastettuja tehtäviä. Ajastettuihin tehtäviin lisätään skripti, joka lataa päivitysjärjestelmäkirjaston ja tarkistaa asetuksistaan, onko automaattipäivitys päällä. Mikäli päivitys on kytketty päälle, skripti kyselee päivityspalvelimelta saatavilla olevia D3-järjestelmäversioita ja uuden version löytyessä suorittaa sen asennuksen, kuin hallintakäyttöliittymästä olisi painettu lataus- ja asennusnappeja. Raportit luodaan ja lähetetään kuten hallintakäyttöliittymässäkin.

9 Lopputulokset

Työ tehtiin suunnitelmamuotoisena ja toteutettiin Deltagon Group Oy:n työkaluihin paketoinnin ja asiakaspalvelinten osalta täysin. Jakelujärjestelmän yum-repositorio toteutettiin asiakaspalvelinten toiminnallisuuden yhteydessä ja osana yhtiön sisäverkon uudelleenjärjestelyä.

Päivitysjärjestelmä on toteuttanut alkuperäisen tehtävänsä ja lyhentänyt yksittäisen asiakaspalvelimen komplikaatiottoman päivitysajan useasta tunnista muutamiin minuutteihin. Virhetilanteita aiheuttavat päivitykset saadaan valtaosassa tapauksista korjattua jo testausvaiheessa. Pienien korjauspäivitysten julkaisu ja jakelu on nopeutunut huomattavasti, sillä niiden lataus ja asennus onnistuu napin painalluksella.

Päivityspalvelinta voidaan käyttää myös muiden Deltagon Group Oy:n tarjoamien palveluiden jakeluun, kun kohdepalvelimet ovat samoja D3-järjestelmiä. Näin on jo ryhdytty toimimaan joidenkin D3-lisäpalveluiden kanssa.

Asiakaspalvelimen asetusten kustomoitavuudesta johtuen myös eri alustojen päivityspaketit on pystytty toteuttamaan jakamalla päivityspalvelimen repositorio niitä vastaavin hakemistopolkuihinsa, joissa oikean alustan paketit ovat haettavissa ja asiakaspalvelimen repositoriotiedot asetettavissa osoittamaan oikeaan repositoriopolkuun.

Ydintoiminnallisuuksien kirjastototeutusmalli sallii myös sen, että sitä toteuttavia käyttöliittymiä voidaan implementoida muihinkin päivitystehtäviin. Samaa kirjastoa voidaan pienillä muutoksilla käyttää täysin D3-järjestelmästä irrallisten tuoteiden päivittämiseen, täysin eri repositoriosta. Mikään ei periaatteessa estä järjestelmäpäivitysten tekemistä kirjastoa käyttäen, mutta käytännön syistä käsin ajettu yum-päivitys on todennäköisesti parempi vaihtoehto. Kirjasto kuitenkin pystyisi siihen.

Toteutus ei myöskään ole täysin riippuvainen yum ohjelmistosta. Sen korvaaminen jollain toisella paketinjakelujärjestelmällä ei tuota asiakasohjelmistoon ylitsepääsemättömiä käytännön esteitä, sillä kaikki komennot voidaan korvata tarvittavilla uusilla komennoilla. Järjestelmä kokonaisuutena on siis hyvin taipuvainen uusiin, parempiin toteutustapoihin.

Lähteet

1 Online Perl Documentation [verkkodokumentti] [viitattu 6.11.2012] Saatavissa: http://www.perl.org.

2 Market Share [verkkodokumentti] [viitattu 6.11.2012] Saatavissa: http://www.mysql.com/why-mysql/marketshare/.

3 Linux servers keep growing, Windows & Unix keep shrinking [verkkodokumentti] 15.3.2012 [viitattu 7.11.2012] Saatavissa: http://www.zdnet.com/blog/open-source/linux-servers-keep-growing-windows-and-unix-keep-shrinking/10616.

4 Macros: Helpful Shorthand for Package Builders [verkkodokumentti] [viitattu 6.11.2012] Saatavissa: http://www.rpm.org/max-rpm/s1-rpm-inside-macros.html.

5 Conway, Damian. 2005. Perl Best Practices, Standards and Styles for Developing Maintainable Code. O'Reilly Media.