Opas turvalliseen Wordpress-kehittämiseen

GNU/Linux Debian 12 -pohjaisen palvelimen turvallisuusohjelmistot ja -konfiguraatiot

Kalle Vesanterä

AIMlearning-hanke (Centria-ammattikorkeakoulu)

Euroopan unionin osarahoittama

Elinkeino-, liikenne- ja ympäristökeskus

AIM learning

centria ammattikorkeakoulu

kpedu

Kokkola Karleby

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

root's password:
```

Tietoa hankkeesta

Tämä työ on toteutettu Euroopan Unionin osarahoittamassa AIMlearning-hankkeessa, joka toteutetaan 1.2.2024 - 30.4.2027 välisenä aikana. Hanketta koordinoi Centria-ammattikorkeakoulu ja se toteutetaan yhteistyössä Keski-Pohjanmaan ammattiopisto Kpedun kanssa.

Lisätietoa AIMlearning-hankkeesta sekä muita osallistujaprojekteja löydät osoitteessa https://aimlearning.fi/.

Vastuuvapauslauseke

Lisenssi / License

Euroopan unionin osarahoittama

Sisällysluettelo

Johdanto

Idea tähän oppaaseen syntyi osana tieto- ja viestintätekniikan insinööriopintojen harjoittelua. Tarkoituksena oli luoda jonkinlainen ohjeistus AIMlearning-hankkeen osallistujille, jonka avulla web-kehittämistä voitaisiin tuoda (ainakin jonkin verran) turvallisempaan suuntaan. Allekirjoittaneelta itseltään löytyy melko kattavaa osaamista web- ja Wordpress-kehityksestä, sekä Linuxin parissa työskentelystä, joten työn aloittaminen oli varsin luonnollista. Lopulta Wordpress-maailman ohjeistus alkoi kasvaa valtavasti, jolloin koko työn tavoite keskitettiin Wordpress + Linux -web-palvelinohjeistukseen.

Opasta alettiin tekemään osissa. Ensimmäisen vaiheen aikana jokaisen osion peruskonfiguraatiot ja testit ajettiin tavallisessa Linux-käytössä. Kaikki tämä dokumentoitiin. Toisessa vaiheessa Wordpress-käyttöön tehtiin spesifit konfiguraatiot, jotka jälleen dokumentoitiin ja testattiin. Näistä Wordpress-konfiguraatioiden dokumentaatioista luotiin viimein osiot tähän oppaaseen.

Kaikki konfiguraatiot ja asennukset on siis tehty oikeassa elämässä. Niitä on testattu, rikottu ja korjattu. Tiedot on pyritty hakemaan täysin julkisista ja mahdollisimman luotettavista lähteistä. Tekoälyn käyttö on pidetty täydellisen minimissä; oppaan jokaisen tekstirivin on tuottanut ihminen. Muutamaan hyvin yksityiskohtaiseen ongelmaan on kysytty apua tekoälyltä, kunnes on todettu tämän hallusinoivan aivan mitä sattuu, jolloin tiedot on etsitty ja varmistettu oikeista lähteistä.

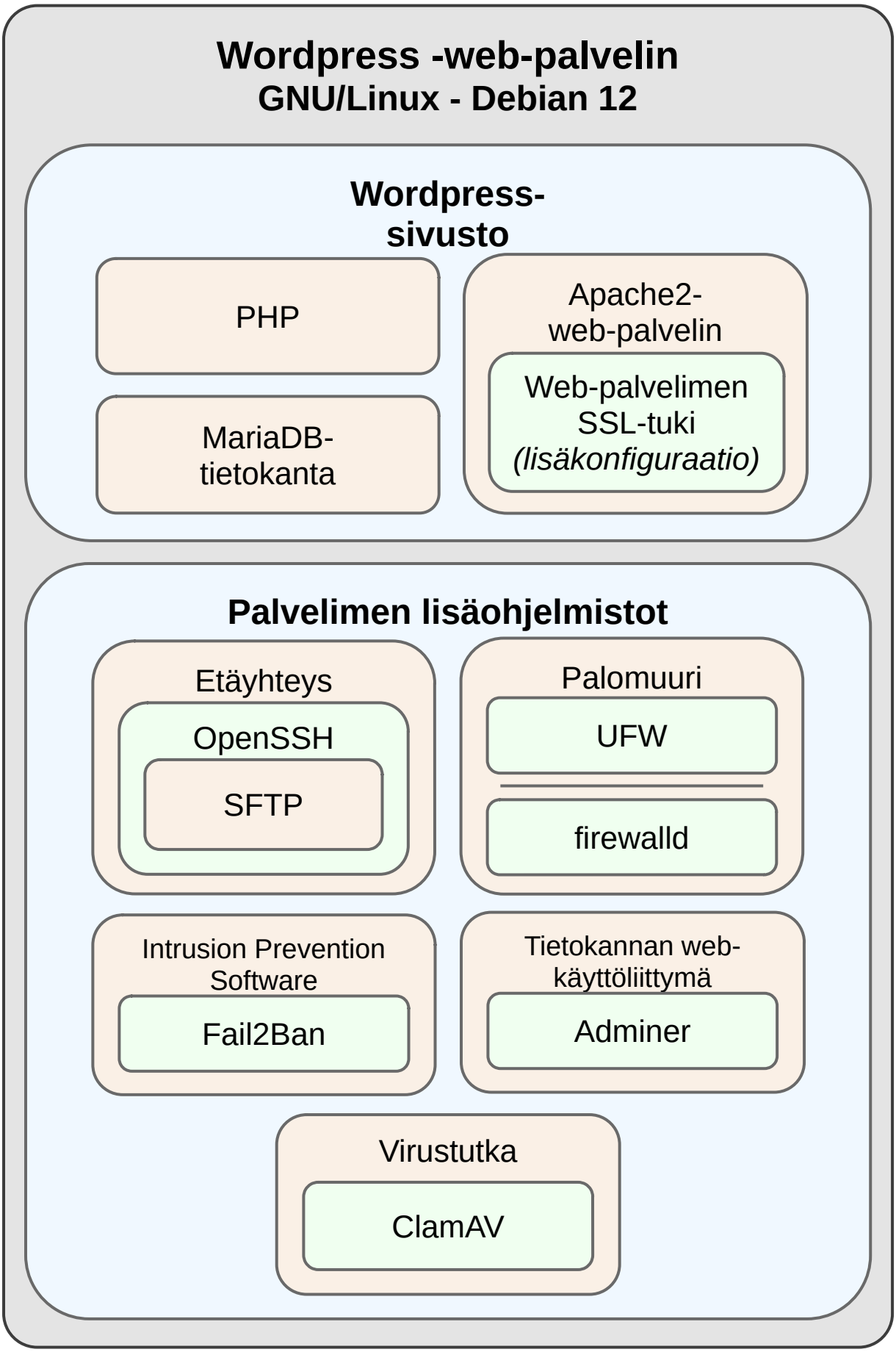
Oheinen kuvaaja näyttää mukavasti rakennetun palvelimen sisällön. Jokaisesta osioista on tehty työohjeistukset, joiden muodostama kokonaisuus luo hyvän pohjan turvalliselle Wordpress-palvelimelle. Ohjeistusta voi kuitenkin hyödyntää ohjenuorana mille tahansa palvelimelle; miltei jokainen yksittäinen osio voidaan ottaa käyttöön erikseen näin halutessaan.

Koska opas on tehty täysin opiskelijatyönä, uskon sen sisältävän elementtejä joita opiskelijat itse arvostavat omassa oppimateriaalissaan. Lisäksi koko julkaisu on täysin julkinen, joten kynnys sen hyödyntämiseen missä tahansa yhteydessä on mahdollisimman matala. CC-BY-SA -lisenssi myös sallii sisällön muokkaamisen, jolloin kuka tahansa saa vaikkapa laajentaa sisältöä tai korjata selkeät virheet.

Tämän oppaan kirjoittaminen on ollut melkoinen matka. Kolme kuukautta intensiivistä työskentelyä, jonka lopputuloksena karvan alle 150 sivua dokumentaatiota ja osittainen järjenmenetys. Vaikka aihepiiri on varsin tuttu, on tämän aikana oppinut uutta melko tavalla. Teknisen osaamisen lisäksi projektin- ja ajanhallinta sekä dokumentaatiotaidot ovat kehittyneet valtavasti. Onko tekeminen ollut aina mukavaa? Ei. Ryhtyisinkö tähän uudelleen? Ehdottomasti.

Kalle Vesanterä
31.7.2025

Euroopan unionin osarahoittama



Kuvaaja 1: Wordpress-palvelimen moduulit

Näin luet tätä opasta

Tämä opas on alun alkaen suunniteltu yksi moduuli kerrallaan. Tämä tarkoittaa, että jokainen konfiguraatiokategoria on testattu toiminnaltaan itsenäisesti ennen kuin se on siirretty Wordpress-palvelimelle. Punainen lanka oppaassa on Wordpress-kehityspalvelimen konfiguroimisessa alusta loppuun, mutta tarvittaessa jokainen erillinen ohjelmisto konfiguraatioineen voidaan soveltaa mihin tahansa palvelimeen / työasemaan.

Tämä opas on tehty sillä oletuksella, että lukija hallitsee perustiedot ja -taidot Linuxin ja komentorivin käytöstä. Varsinkin oikeita palvelimia ja työasemia konfiguroidessa tulee olla erittäin tarkkana jokaisen komennon ja konfiguraatiomuutoksen yhteydessä. Lopullinen vastuu palvelimen, työaseman ja käyttöjärjestelmän toiminnasta ja turvallisuudesta on aina komennon suorittajalla!

sudo:n käyttö

Aivan ensimmäiseksi maininta `sudo` -komennon käytöstä oppaassa. **Yhteenkään esimerkkikomentoon ei ole sisällytetty sudo:a.** Tämä ei kuitenkaan tarkoita etteikö sitä tarvittaisi palvelimen konfiguroinnin aikana.

Valinta on tehty tietoisesti. Yleensä suoritettava komento ilmoittaa, tavalla tai toisella, että kyseisellä käyttäjällä ei ole oikeuksia suorittaa kyseistä komentoa. Tällöin tiedetään, että oikeuksia tulee hetkellisesti nostaa, jotta vaadittava muutos saadaan tehtyä. `sudo` on komentona erittäin voimakas, eikä sitä tule käyttää turhaan.

Komentorivi

Opas on kirjoitettu työohjemaiseen tyyliin, joten se myös toimii parhaiten kun työtä tehdään reaaliajassa. Erilaisia komentoja on pyritty avaamaan, jotta lukija ymmärtää niiden toiminnan mahdollisimman hyvin. Komentorivilaatikoihin on tarvittaessa sisällytetty komennon tulosteet. Itse komennon erottaa `$` -merkistä ja harmaasta taustasta.

```
$ mariadb-secure-installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
...
```

SFTP:n tapauksessa logiikka on samanlainen, mutta taustaväri ja etumerkintä erottaa komennot tavallisista konsolikomennoista.

```
sftp> pwd
Remote working directory: /
...
```


Euroopan unionin osarahoittama

Jos puolestaan kyseessä on muokattava tiedosto, näytetään sen komentorivi rivinumeroiden kera. Rivinumerot vastaavat esimerkkipalvelimella tehtyä konfigurointitiedostoa. Muokattavan tiedoston polku on sisällytetty vastaavaan tekstisisältöön.

```
1 # MariaDB 11.4 repository list - created 2025-05-07 12:34 UTC
2 # https://mariadb.org/download/
3 X-Repolib-Name: MariaDB
4 Types: deb
...
```

Informaatiolaatikot

Ohjeistuksen sekaan on asetettu relevantteihin kohtiin erillisiä informaatiolaatikoita. Näiden informaatiolaatikoiden tarkoituksena on avata tiettyjä komentoja ja konfiguraatioita lisää. Nämä laatikot saattavat myös sisältää tärkeitä varoituksia mahdollisista vaaranpaikoista konfiguroinneissa. Informaatiolaatikot tunnistaa helposti seuraavista symboleista:

Lisätietoa komennosta tai konfiguraatiosta.

Pysähdy, ole tarkkana!

Mahdollisesti vaarallinen konfiguraatio!

Lähdeluettelot

Koska jokainen osio on rakennettu alunperin itsenäiseksi moduuliksi, on näillä myös omat itsenäiset lähteensä jotka sijaitsevat osion lopussa. Lähdeluettelot sisältävät internetlähteitä, sekä käytössä olevan Debian 12 -jakelun `man` -komennon alaisia manuaaleja.

Lähdeluetteloita kannattaa hyödyntää tiedonhaussa, jos tarkoituksena on tehdä kustomoituja konfiguraatioita tai ongelmanratkaisua. Työohje antaa hyvän yleiskuvan ohjelmistojen ja konfiguraatioiden toiminnallisuudesta, mutta on kuitenkin mahdotonta tehdä sataprosenttisen kattava ohjeistus kaikkiin tilanteisiin.

Esimerkkipalvelin

Oppaan esimerkeissä käytetty palvelin on Debian 12 -jakelulla varustettu virtuaalikone. Koko ohjeistus on siis suunnattu Debian 12 -jakelulle, mutta yleisesti ottaen perusperiaatteet eri ohjelmistoilla ja konfiguraatioilla ovat kautta jakeluiden hyvin samankaltaiset. Jos konfigurointia tehdään jollekin toiselle jakelulle, erityisesti jos tämä ei ole Debian-pohjainen, tulee mahdollisten eroavaisuuksien kanssa olla tarkkana!

Wordpressin käyttöönotto palvelimella

Wordpress on yksi suurimmista ja suosituimmista Content Management System -alustoista, mitä avoimen lähdekoodin markkinoilla on tarjolla. Sitä on käytetään hyvin monenlaisilla palvelimilla ja alustoilla, mikä tarkoittaa että samasta ohjelmistokokonaisuudesta löytyy monia erilaisia versiota. Onneksi, ainakin elämänlaadun kannalta, asentaminen on tehty tavan käyttäjille ja kehittäjille hyvin helpoksi. Linux-palvelimella tai -työasemalla työskennellessä pääsääntöisesti kaikki Wordpressiin liittyvät ohjelmistot on saatavilla jakelun omasta repositoriosta. Esimerkiksi Debian 12 -jakelulla Wordpressin ja tämän riippuvuudet saa asennettua yhdellä komennolla:

```
$ apt install wordpress curl apache2 mariadb-server
```

Tämä ei kuitenkaan tarkoita että Wordpress on nyt täysin käyttövalmis, vaan ylimääräistä konfiguraatiota tulee tehdä jotta turvallinen kehittäminen voidaan aloittaa. Tässä luvussa käymme läpi Wordpressin laitteistovaatimuksia, ohjelmistoriippuvuuksia sekä konfiguraatioita joita näille tulee tehdä. Asennamme jokaisen riippuvuuden erikseen, jolloin pääsemme tutustumaan tarkemmin näiden ohjelmistojen tarkoitukseen ja toimintaan Wordpressin suhteen.

Lähteet:

https://wiki.debian.org/WordPress

Wordpressin ominaisuuksia

Wordpress on kypsynyt ajan kanssa hyvin monipuoliseksi CMS-alustaksi. Ominaisuuksia löytyy melkein jokaiseen käyttökohteeseen ja tarvittaessa toiminnallisuutta voidaan laajentaa plugin-lisäosilla.

Web-käyttöliittymä

Käytännössä Wordpressin hallinta tapahtuu täysin Wordpress Dashboardin kautta. Tämän hallintapaneelin ylitse pystyy muokkaamana sivuston sisältöä, mediatiedostoja, plugineja, teemoja; käytännössä kaikkea mitä sivustolta löytyy. Kunhan palvelin on muuten kunnossa, ei Wordpressiin itsessään tarvitse koskea muutoin kuin Dashboardin kautta.

Pluginit

Wordpress itsessään on nykypäivän standardeilla melko ”raaka” ohjelmisto. Saatavilla on kuitenkin satoja plugin lisäosia.

Euroopan unionin osarahoittama

Näillä lisäosilla voidaan kustomoida sivuston toiminnallisuutta; jopa turvallisuutta pystytään lisäämään plugineilla.

Hyvin suuri osa plugineista on ilmaisia, tai ainakin niistä on saatavilla ilmaisversio. Wordpressin pluginien ympärille on kuitenkin syntynyt täysin oma liiketoimintakulttuuri. Maksulliset pluginit yleensä ovat huomattavasti laajempia ja yleensä näitä päivitetään tiheämmin. Ilmaispluginit ovatkin monesti abandonwarea, joten niiden asentamisen kanssa kannattaa olla tarkkana.

Teemat

Wordpress-sivustojen ulkoasu rakennetaan yleensä teemojen avulla. Teemoja on saatavilla lukemattomia, niitäkin ilmaisia ja maksullisia. Teeman voi myös tehdä itse, jos kykyjä graafiseen suunnitteluun löytyy.

Muokkaustyökalut

Wordpressin Dashboard sisältää jo valmiiksi melko monipuolisia muokkaustyökaluja, joilla voidaan rakentaa verkkosivuston sisältösivuja. Valmiiden muokkaustyökalujen lisäksi on saatavilla kolmannen osapuolen tekemiä plugineita, joilla muokkausominaisuuksia voidaan laajentaa.

Sivuston teema yleensä määrittää sivuston visuaalisen näkymän, jonka sisälle itse tietosisältö voidaan määrittää. Muokkaustyökaluilla voidaan myös lisätä sivustolle kustomoitua koodia, jolla voidaan rakentaa yksityiskohtaista toiminnallisuutta sivuston sisälle.

Lähteet:

https://wordpress.org/about/features/

Laitteistovaatimukset

Wordpress itsessään on melko kevyt ohjelmistopaketti, ainakin moderneille tietokoneille. Suorituskykyvaatimukset toki elävät sen mukaan, millainen sivusto rakennetaan. Jos kyseessä on suuri verkkokauppa isolla tietokannalla, useammalla pluginlla ja valtavalla liikennemäärällä, tulee laitteiston jo olla melko suorituskykyistä.

Toisaalta, tavallinen blogisivusto ei juurikaan vaadi tehoja tietokoneelta tai palvelimelta. Käytännössä voimme ajatella ettei Wordpress-sivuston kehitysvaiheessa laitteelta vaadita suurta suorituskykyä. Tärkeintä on, että laitteisto on sillä tavalla nykyaikaista, että kehitysympäristö voidaan pystyttää vakaalle alustalle. Tämän oppaan kaikki esimerkit on tehty virtuaalikoneella sijaitsevassa palvelimessa, eikä ongelmia ole ollut lainkaan.

Vähimmäislaitteistovaatimukset (Kinsta.com)	
Prosessori	Vähintään 1,0GHz
RAM	Vähintään 512MB
Tallennustila	Vähintään 1GB

Taulukko 1: Kinsta.com vähimmäislaitteistovaatimukset Wordpressille

Prosessori

Web-sivustojen kannalta prosessori on keskeisessä roolissa. Kaikki tapahtumat joita web-palvelimella tapahtuu, lisäävät kuormaa tämän prosessorille. Moderni rauta on jo pitkään ollut hyvin suorituskykyistä, eikä esimerkiksi prosessorin kellotaajuus tule olemaan ongelma web-palvelimen kannalta.

Käytännössä suurempi merkitys on prosessorin ytimien määrällä, sillä niiden avulla web-palvelin pystyy moniajamaan suuria tehtävämääriä yhtä aikaa. Mitä enemmän sivustolla on samanaikaisia käyttäjiä, sitä enemmän ytimien määrästä hyötyy.

Tavalliselle verkkosivustolle, jossa samanaikaisia vierailijoita on maksimissaan muutama sata, riittää todennäköisesti 8-ytiminen prosessori vallan mainiosti. Tähän toki vaikuttaa moni muuttuja, kuten verkkosivun sisältö ja Wordpressin tapauksessa pluginien määrä.

Sivuston kehityksen aikana suurta kuormaa ei kuitenkaan ole, jolloin tavallisella työasemaraudalla pärjää varsin hyvin. Tuotantoon vietäessä sivusto todennäköisesti pyörii jonkinlaisen hostauspalvelun tai palvelimen päällä, jolloin suorituskyvyn riittävyys ja skaalautuvuus voidaan tarjota palveluntarjoajan puolelta automaattisesti.

RAM (keskusmuisti)

Keskusmuistiin voidaan melkeinpä soveltaa samaa logiikkaa kuin prosessoriin suorituskykyvaatimusten osalta. Mitä enemmän web-palvelimella on sisältöä, sen enemmän keskusmuistilta vaaditaan kokonaiskapasiteettia. Myös keskusmuistin nopeudella on väliä, varsinkin jos samanaikaisia asiakkaita on paljon.

Määrällisesti keskusmuistia ei vaadita kovinkaan paljoa. Wordpressin tapauksessa hyvin tavallisesti 1GB riittää erittäin hyvin. Jos työasemalta tai palvelimelta löytyy siis oletuksena 8GB keskusmuistia, ei tuo tule loppumaan mitenkään kesken.

Tallennustila

Verkkosivujen tapauksessa tallennustilan vaadittu määrä riippuu hyvin pitkälti siitä, millaista sisältöä verkkosivuilla tarjoillaan. Jos sivusto sisältää suuren määrän

Euroopan unionin osarahoittama

mediatiedostoja, kuten videoita ja kuvia, tulee tämä tietenkin vaatimaan paljon tallennustilaa.

Wordpressin kanssa käytetään usein erilaisia teemoja, jotka saattavat olla kooltaan suhteellisen suuria. Lisäksi tietokanta, johon miltei kaikki sisältö tallennetaan, kasvaa ajan kuluessa. Tallennustilan nopeus voi myös vaikuttaa oleellisesti verkkosivuston toimintaan. Onneksi nykyään SSD-asemat ovat hyvin yleisiä ja varmatoimisia, joten sellaisen valinta sivustolle on helppoa. Halutessaan tallennustilan kestävyyden kanssa voidaan pelata varman päälle; kaksi asemaa voidaan peilata keskenään, jolloin yhden hajotessa dataa ei kuitenkaan menetetä.

Kehityksen aikana tallennustilavaatimukset tiedetään usein melko tarkasti, eikä työasemalta tai palvelimelta tämä tule loppumaan kesken. Tuotantoon vietäessä kannattaakin kiinnittää huomiota sivuston käyttötarkoitukseen, ja kaavoittaa tallennustila sen mukaisesti.

Lähteet:

https://kinsta.com/blog/wordpress-server-requirements/

https://godeinteractive.com/magazine/wordpress-server-requirements/

https://documentation.commvault.com/11.20/hardware_specifications_for_web_server.html

Ohjelmistovaatimukset

Lähtökohtaisesti Wordpressin kanssa, kuten minkä tahansa muun ohjelmiston, voidaan ajatella uusimman ohjelmistoversion olevan kaikista paras. Tähän löytyy toki aina poikkeuksia. Koska Wordpressiä ajetaan niin monella eri alustalla, on niillä erilaisia vaatimuksia versioille. Esimerkiksi Debianin repositorioista jakelun versioille 10-13 löytyy neljä eri versiota Wordpressistä. Erilaisten ohjelmistoversioiden risteämisen mahdollisia haittoja voivat olla bugit, epästabiilius tai yhteensopimattomuus muiden ohjelmistojen kanssa.

Wordpressin kehittäjät itse kehottavat asentamaan aina kaikista uusimman version. Uusin versio on se, jonka turvallisuus on kehittäjien puolesta taattu. Tätä versiota myös ylläpidetään aktiivisesti.

Loppujen lopuksi, suositulle Linux-jakelulle asennettaessa repositorion sisältämä versio riippuvuuksineen on yleensä turvallinen ja stabiili vaihtoehto. Halutessaan oman jakelun käyttämät versiot voi etsiä jakelun paketinhallintaohjelman kautta, tai vaihtoehtoisesti jakelun repositorion verkkohakemistosta. Jos kuitenkin halutaan varmistaa että kaikki on varmasti kunnossa, voidaan asentaa kaikista uusin versio vaatimuksineen suoraan Wordpressiltä.

Wordpress 6.8.1 ohjelmistovaatimukset (julkaistu 30.4.2025)			
PHP	7.4+		
Tietokanta	MySQL 8.0+	*tai*	MariaDB 10.5+
Web-palvelin	apache2 *(mod_rewrite moduulilla)*	*tai*	Nginx *(ei suoraan täyttä tukea Wordpressille)*
HTTPS-tuki	*Kyllä*		

Taulukko 2: Wordpressin suosittelemat vähimmäisohjelmistovaatimukset

PHP

PHP, eli PHP: Hypertext Preprocessor, on avoimen lähdekoodin skriptauskieli. PHP:tä voi periaatteessa käyttää missä tahansa yhteydessä, mutta erityisesti se on vakiintunut web-kehittämiseen.

Web-puolella PHP:tä voidaan periaatteessa käyttää sekä backendissä että frontendissä; jälkimmäisen kanssa PHP:tä voidaan muun muassa upottaa HTML-koodin sekaan. PHP kuitenkin suoritetaan täysin palvelimen päässä, toisin kuin esimerkiksi frontendin JavaScript.

Alustana Wordpress on hyvin pitkälti toteutettu PHP:llä. Kirjoitushetkellä Github-repositorion mukaan n. 66% koodista on PHP:tä; JavaScript-koodia puolestaan on alle 20% kokonaisuudesta.

MySQL / MariaDB

MySQL on erittäin suosittu avoimen lähdekoodin relaatiotietokanta, joka käyttää pohjanaan SQL-kieltä. Tietokantana MySQL on hyvin pitkälle kypsynyt ja sitä pidetään yleisesti luotettavana ja hyvin skaalautuvana. Lisäksi sen etuna on peruskäytön helppo omaksuminen.

MariaDB puolestaan on MySQL:n haara, joka on tämän entisten kehittäjien luoma. Hyvin pitkälti nämä ovat toiminnaltaan samanlaisia sekä keskenään yhteensopivia. Pellin alla on joitakin enemmän tai vähemmän relevantteja eroavaisuuksia, kuten hakunopeus ja skaalautuvuus.

Wordpress-käytössä molemmat ovat toimivia ratkaisuja. Ensikädeltä MySQL:n voidaan ajatella olevan vakiintuneempi yrityskäytössä. Vaikka molemmat ovat avoimen lähdekoodin tietokantoja, MySQL tarjoaa sekä avoimen että yrityslisenssin. MariaDB on puolestaan täysin avoimen lisenssin tietokanta.

Euroopan unionin osarahoittama

Apache2 ja Nginx

Apache2 ja Nginx ovat molemmat hyvin suosittuja web-palvelimia. Nginx on hieman tuoreempi ratkaisu, jonka alkuperäinen tarkoitus oli ratkaista suuriin määriin samanaikaisia yhteyksiä liittyvät ongelmat.

Apache2 on tietyllä tavalla kokonaisvaltaisempi palvelinohjelmisto. Sen etuna on esimerkiksi dynaamisen sisällön ja PHP-kutsujen natiivi tuki. Apache2 on kuitenkin raskaampi, sillä sen arkkitehtuuri toimii olennaisesti hitaammin kuin Nginx:n.

Nginx onkin siis hyvin suosittu verkkoalustoilla joissa on tarve nopeudelle ja tehokkaalle skaalautuvuudelle. Se ei kuitenkaan sisällä suoraan kaikkia ominaisuuksia joita Wordpress saattaa tarvita.

Yleinen tapa Wordpress-sivustoilla on käyttää Nginxiä eräänlaisena etuliittymänä Apache2:lle, joka toimii varsinaisena web-palvelimena. Nämä molemmat ovat kuitenkin virallisesti tuettuja web-palvelimia Wordpress-käytössä.

Lähteet:

https://wiki.debian.org/WordPress

https://make.wordpress.org/hosting/handbook/

https://wordpress.org/about/requirements/

https://wordpress.org/download/releases/

https://make.wordpress.org/hosting/2025/04/16/wordpress-6-8-server-compatibility/

https://www.php.net/manual/en/introduction.php

https://github.com/WordPress/WordPress

https://www.mysql.com/

https://developer.wordpress.org/advanced-administration/server/web-server/nginx/

https://mariadb.org/

https://www.oracle.com/mysql/what-is-mysql/

https://aws.amazon.com/compare/the-difference-between-mariadb-vs-mysql/

https://nginx.org/en/

https://httpd.apache.org/

https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

https://www.cloudways.com/blog/nginx-vs-apache/

Wordpressin asentaminen

Kuten aiemmin mainittiin, Wordpressin ja tämän riippuvuudet saa yleensä asennettua yhdellä komennolla. Osa ohjelmistoista vaatii kuitenkin erityisiä konfiguraatioita, jotta Wordpress saadaan toimimaan työasemalla tai palvelimella. Tässä osassa esimerkkiasennus on tehty Debian 12 -jakelulla.

Pääpiirteittäin asennus tapahtuu kaikilla jakeluilla samalla tavalla. Joitakin eroavaisuuksia todennäköisesti on, kuten jakelun tukemat ohjelmistoversiot ja mahdolliset tiedostopolut. Nämä kannattaa varmistaa etukäteen, jotta mahdolliset ongelmat voidaan välttää.

ASENNUSVAIHEESSA TULEE ASETTAA MONIA ERI SALASANOJA. NÄIDEN TULEE OLLA VAHVOJA, NE EIVÄT SAA OLLA KAIKKI KESKENÄÄN SAMOJA JA NE KANNATTAA TALLENTAA TALTEEN SALATTUUN SALASANAPANKKIIN.

Ohjelmistoriippuvuuksien asentaminen

PHP

Debian 12:sta pakettirepositoriosta on saatavilla PHP:n versio on 8.2.x. Tämän, sekä sen apupaketit, saadaan asennettua seuraavalla komennolla:

```
$ apt install php php-cli php-common
```

Varmistetaan asennuksen onnistuminen tarkistamalla versionumero:

```
$ php --version
```

Lähteet:

https://wiki.debian.org/PHP#PHP_on_Debian

Tietokanta - MariaDB

Esimerkkiasennuksessa tietokantapalvelimeksi valittiin MariaDB, jonka etuna on täysin avoin ohjelmistolisenssi. Wordpress-käytössä erot MariaDB:n ja MySQL:n välillä ovat kuitenkin marginaaliset, joten mitään estettä MySQL:n käytölle ei ole. Asennusohjeistus on kuitenkin tehty ainoastaan MariaDB:lle.

Versioksi valittiin mahdollisimman tuore ja stabiili versio (11.4). Tämä vaatii uuden repositorion lisäämistä apt:in repositoriohakemistoon. Oikean repositorion tiedot saa haettua MariaDB:n sivustolta: https://mariadb.org/download/?t=repo-config.

Varmistetaan ensin että repositorion lisäämiseen tarvittavat paketit on asennettuna. Tämän jälkeen MariaDB:n repositorion avain lisätään Debianin paketinhallinnan tietoihin:

```
$ apt install apt-transport-https curl
$ mkdir -p /etc/apt/keyrings
$ curl -o /etc/apt/keyrings/mariadb-keyring
'https://mariadb.org/mariadb_release_signing_key.pgp'
```


Euroopan unionin osarahoittama

Repositorion konfiguraatiotiedot tulee lisätä omaan `/etc/apt/sources.list.d/mariadb.sources` -tiedostoonsa. Tarvittaessa tämä voidaan luoda komennolla:

```
$ touch /etc/apt/sources.list.d/mariadb.sources
```

Tiedostoon tulee lisätä seuraavankaltainen sisältö. Tämä voidaan tehdä komentorivipohjaisella tekstieditorilla, kuten `nano`:lla tai `vim`:llä. Ajankohtainen informaatio tulee tarkastaa MariaDB:n sivustolta.

```
 1 # MariaDB 11.4 repository list - created 2025-05-07 12:34 UTC
 2 # https://mariadb.org/download/
 3 X-Repolib-Name: MariaDB
 4 Types: deb
 5 # deb.mariadb.org is a dynamic mirror if your preferred mirror goes offline.
See https://mariadb.org/mirrorbits/ for details.
 6 # URIs: https://deb.mariadb.org/11.4/debian
 7 URIs: https://mirrors.xtom.ee/mariadb/repo/11.4/debian
 8 Suites: bookworm
 9 Components: main
10 Signed-By: /etc/apt/keyrings/mariadb-keyring.pgp
```

Kun avaimet on lisätty paketinhallintaan ja konfiguraatiotiedosto on valmiina, voidaan paketinhallintatyökalu päivittää. Apt hakee MariaDB:n repositorion tiedot, jonka jälkeen täältä voidaan asentaa mariadb-server -paketti.

```
$ apt update
$ apt install mariadb-server
```

Onnistunut asennus voidaan varmistaa tarkastamalla MariaDB:n versionumero:

```
$ mariadb --version
```

Kun asennus on tehty onnistuneesti, tulee seuraavaksi ajaa MariaDB:n turvallisuuskonfiguraatioskripti. Tällä skriptillä saadaan helposti asetettua tietokannalle turvallisia asetuksia ennen kuin sitä lähdetään laajentamaan. Varsinaisesti kehitys-vaiheessa tätä ei ole välttämätöntä tehdä, mutta Wordpressin tapauksessa siitä ei ole mitään haittaa.

TÄSSÄ VAIHEESSA ON ERITTÄIN TÄRKEÄÄ LUKEA KAIKKI MITÄ SKRIPTI KERTOO!

TIETOKANNAN ROOT-KÄYTTÄJÄN SALASANA ON OLETUKSENA TYHJÄ. ÄLÄ HÄMÄÄNNY SIITÄ, ETTÄ KONFIGURAATIOSKRIPTI KERTOO SALASANAN OLEVAN OLEMASSA, VAAN ASETA TURVALLINEN SALASANA!

Lähtökohtaisesti seuraavat vastaukset ovat turvallisia, mutta ne ovat täysin omassa harkinnassa tilanteesta riippuen. **Varsinkin production-palvelimella nämä voivat olla elintärkeitä.**

```
$ mariadb-secure-installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB SERVERS IN
PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.
...
```

Asennus on luonut tietokannalle automaattisesti root-käyttäjän. Tälle käyttäjälle tulee asettaa turvallinen salasana, sillä sen oikeudet ovat rajattomat koko tietokantajärjestelmässä. Vielä tässä vaiheessa salasana on kuitenkin tyhjä, joten vastauksen tulee olla tyhjä.

```
...
Enter current password for root (enter for none): [paina enter]
OK, successfully used password, moving on...
...
```

`Unix_socket authentication`:in käyttöönotto on mielenkiintoinen asetus. **Vaikka esimerkissä sitä ei ole otettu käyttöön, kannattaa sen käyttöönottoa harkita.** Käytännössä tämä tarkoittaa, että varsinaista salasana-autentikointia tietokantaan ei tarvita, vaan autentikointi hoidetaan käyttöjärjestelmän käyttäjätietojen perusteella. Tämä saattaa lisätä turvallisuutta; esimerkiksi mahdollisesti murretun web-palvelimen www-data -käyttäjän kautta ei ole mahdollisuutta päästä tietokantaan root-käyttäjänä. Lisätietoja: https://mariadb.com/kb/en/authentication-plugin-unix-socket/

Euroopan unionin osarahoittama

Jätetään tietokannan salasana-autentikointi käyttöön:

```
...
Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.
...
```

Nyt tulee olla erittäin tarkkana! Skripti olettaa, että olemme kirjautuneet alussa root-käyttäjälle sisään, käyttäen salasanaa. Tämä ei kuitenkaan pidä paikkaansa! Salasanaa ei ole todellisuudessa olemassa, vaan se on tyhjä. Skripti ohjeistaa, että voimme jatkaa vaihtamatta root-käyttäjän salasanaa; tämä ei pidä paikkaansa! Salasana tulee vaihtaa tyhjästä turvalliseksi, oikeaksi salasanaksi.

```
...
You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password: [aseta salasana]
Re-enter new password: [varmista salasana]
Password updated successfully!
Reloading privilege tables..
 ... Success!
...
```

Poistetaan testikäyttäjä(t) tietokannasta.

```
...
By default, a MariaDB installation has an anonymous user, allowing anyone to log
into MariaDB without having to have a user account created for them.  This is
intended only for testing, and to make the installation go a bit smoother.  You
should remove them before moving into a production environment.


Remove anonymous users? [Y/n] y
 ... Success!
...
```

Estetään root-käyttäjäksi kirjautuminen kaikkialta muualta, paitsi paikalliselta palvelimelta. Käytännössä SSH-yhteyden yli pystyy kuitenkin kirjautumaan root-käyttäjäksi.

```
...
Normally, root should only be allowed to connect from 'localhost'.  This ensures
that someone cannot guess at the root password from the network.


Disallow root login remotely? [Y/n] y
 ... Success!
...
```

Poistetaan testitietokanta; emme tarvitse sitä mihinkään.

```
...
By default, MariaDB comes with a database named 'test' that anyone can access.
This is also intended only for testing, and should be removed before moving into
a production environment.


Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
...
```

Otetaan välittömästi käyttöön skriptissä asetetut konfiguraatiot.

```
...
Reloading the privilege tables will ensure that all changes made so far will
take effect immediately.


Reload privilege tables now? [Y/n] y
 ... Success!


Cleaning up...


All done!  If you've completed all of the above steps, your MariaDB installation
should now be secure.


Thanks for using MariaDB!
```


Euroopan unionin osarahoittama

Lopuksi käynnistämme MariaDB:n palvelun uudelleen, jolloin tehdyt konfiguraatiot tulevat varmasti käyttöön.

```
$ systemctl restart mariadb
```

Lähteet:

https://mariadb.com/kb/en/authentication-plugin-unix-socket/

Web-palvelin - Apache2

Joissakin jakeluissa Apache2 web-palvelin on valmiiksi asennettuna. Tämä voidaan varmistaa tarkastamalla palvelun status:

```
$ systemctl status apache2
```

Jos Apache2 ei ole asennettuna, saadaan seuraavankaltainen virheilmoitus:

```
Unit apache2.service could not be found.
```

Tämä voidaan korjata asentamalla Apache2 jakelun pakettirepositoriosta:

```
$ apt install apache2
```

Jotta Wordpress toimii täyden spesifikaation mukaisesti, tulee Apache2:ssa ottaa käyttöön `mod_rewrite` -moduuli. Tämä moduuli mahdollistaa muunmuassa osoitepolkujen dynaamisen uudelleenkirjoittamisen järjestelmän sisällä.

```
$ a2enmod rewrite
$ systemctl restart apache2
```

Oletettavasti Apache2 ei ole automaattisesti aktiivisessa tilassa. Tämä voidaan asettaa aktiiviseksi komennolla:

```
$ systemctl start apache2
```

Vastaavasti, jos haluamme asettaa Apache2:n aktiiviseksi aina koneen käynnistyessä:

```
$ systemctl enable apache2
```

Wordpress

Asentaminen pakettirepositoriosta

Esimerkkikäytössä olevan Debian 12 -jakelun repositorion tuettu Wordpress-versio on 6.1.6. Pakettirepositoriosta asennettuna Wordpressin käyttöönotto on melko suoraviivaista, mutta vaatii hieman konfiguraatioita. Seuraavaa esimerkkiä voi käyttää ohjenuorana, toki omaan käyttöön soveltaen.

Asennetaan Wordpress pakettirepositoriosta:

```
$ apt install wordpress
```

Apache2 toimii Virtual Host -konfiguraatiotiedostojen pohjalta. Käytännössä nämä ovat sivustokohtaisia konfiguraatioita, jotka yhdistetään tiettyyn domainiin, alidomainiin, IP-osoitteeseen ja porttiin tai näiden kaikkien yhdistelmiin. Virtual Host -tiedostot sijaitsevat `/etc/apache2/sites-available/` -hakemistossa, josta Apache2 osaa ottaa ne tarvittaessa käyttöön. Wordpress-sivustoa varten voidaan luoda uusi konfiguraatiotiedosto:

```
$ touch /etc/apache2/sites-available/wp.conf
```

Luotuun tiedostoon voidaan asettaa seuraavankaltainen sisältö, esimerkiksi `nano` -tekstieditoria hyödyntäen:

```
<VirtualHost *:80>
    ServerName localhost
    ServerAdmin webmaster@localhost
    DocumentRoot /usr/share/wordpress
    Alias /wp-content /var/lib/wordpress/wp-content
    <Directory /usr/share/wordpress>
          Options FollowSymLinks
          AllowOverride Limit Options FileInfo
          DirectoryIndex index.php
          Require all granted
    </Directory>
    <Directory /var/lib/wordpress/wp-content>
          Options FollowSymLinks
          Require all granted
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
          CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```


Euroopan unionin osarahoittama

Käytetyssä esimerkissä ServerName ja ServerAdmin -riveillä domain (`example.com`) on korvattu `localhost`:lla. Tämä on riittävää, jos Wordpressiä käytetään ainoastaan paikalliseen kehittämiseen. Muissa tapauksissa konfiguraatiota tulee muuttaa asiaankuuluvasti. Rivit voi siis domainia käytettäessä konfiguroida vastaavanlaisesti:

```
1 <VirtualHost *:80>
2     ServerName blog.example.com
3     ServerAdmin webmaster@example.com
...
```

Apache2:ssa oletus- Virtual Host on `000-default`. Tämä tulee vaihtaa edellä luodun konfiguraation mukaiseksi:

```
$ a2dissite 000-default
$ a2ensite wp
```

Uusi konfiguraatio saadaan aktivoitua lataamalla Apache2-palvelu uudelleen:

```
$ systemctl reload apache2
```

Toimiakseen Wordpress vaatii vielä kirjautumistiedot aiemmin luotuun MariaDB-tietokantaan. Nämä tulee konfiguroida `/etc/wordpress/config-localhost.php` -tiedostoon, josta Wordpress noutaa ne.

Tiedostoon luodaan seuraava sisältö, jossa **`password`** vaihdetaan **uniikkiin ja turvalliseen salasanaan**:

```
1 <?php
2     define('DB_NAME', 'wordpress');
3     define('DB_USER', 'wordpress');
4     define('DB_PASSWORD', 'password');
5     define('DB_HOST', 'localhost');
6     define('WP_CONTENT_DIR', '/var/lib/wordpress/wp-content');
7 ?>
```

Tietenkin, myös itse MariaDB-tietokantaan tulee luoda edellistä konfiguraatiota vastaava käyttäjä. Tämä saadaan helposti aikaan luomalla SQL-tiedosto, jonka avulla MariaDB luo käyttäjän ja tietokannan Wordpressin käyttöön. Käytännössä tiedosto kannattaa tallentaa väliaikaisesti esimerkiksi omaan kotihakemistoon, josta sen voi myöhemmin poistaa:

```
$ touch ~/wp.sql
```

Luotuun `wp.sql` -tiedostoon asetetaan muutama SQL-komento. Kyseisessä komennossa **`password`** vaihdetaan samaan, mikä määriteltiin `/etc/wordpress/config-localhost.php` -tiedostoon.

```
1 CREATE USER wordpress@localhost IDENTIFIED BY 'password';
2 CREATE DATABASE wordpress;
3 GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
4 ON wordpress.*
5 TO wordpress@localhost
6 IDENTIFIED BY 'password';
7 FLUSH PRIVILEGES;
```

Tässä vaiheessa `wordpress@localhost` jätetään sellaisekseen, domainista huolimatta. Tietokanta, siihen kirjautuminen, muokkaaminen, ynnä muut toimet, tapahtuvat kaikki ’pellin alla’ ainoastaan palvelimen sisällä.

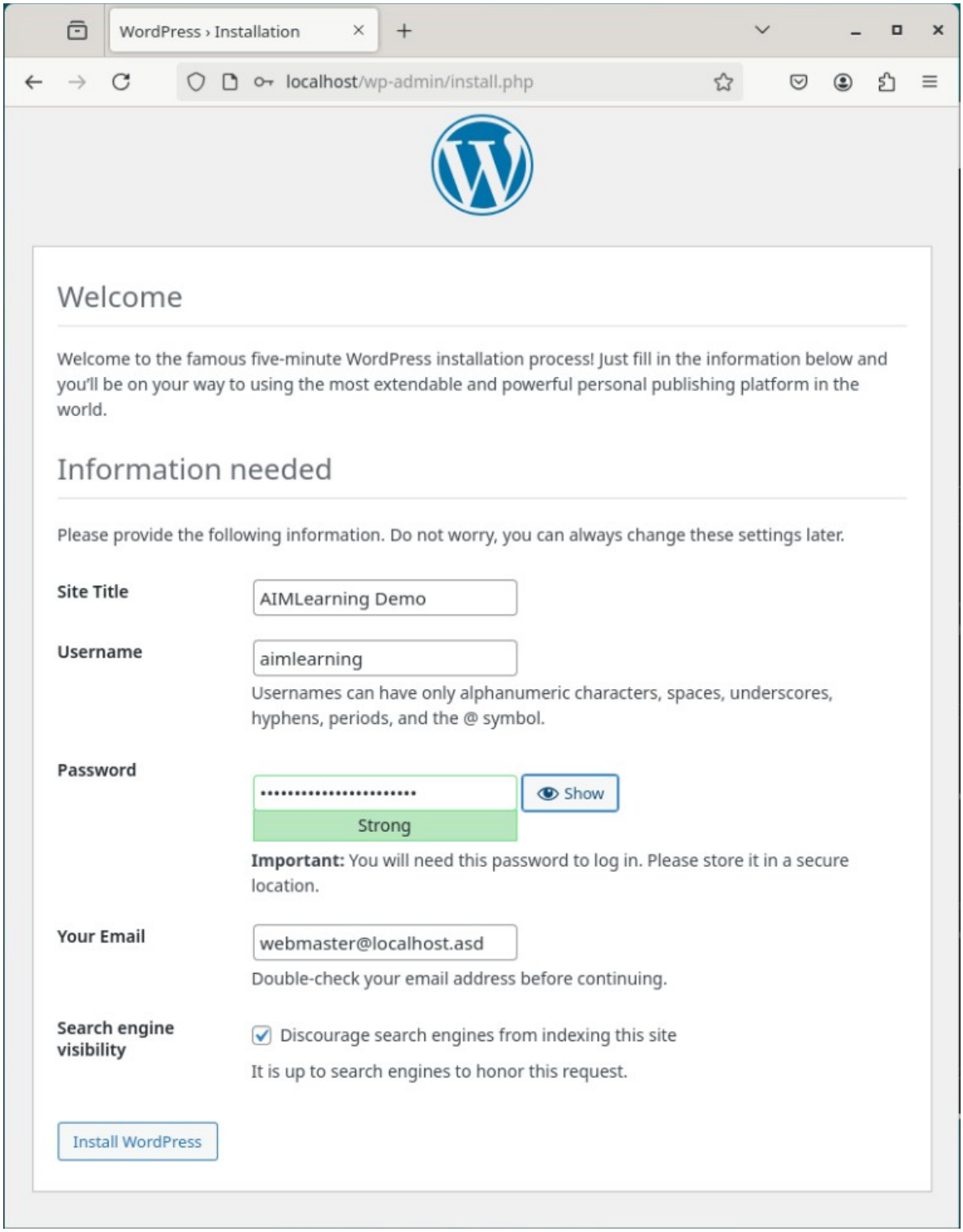
Wordpressin MariaDB-käyttäjä ja -tietokanta päästään luomaan suorittamalla luotu SQL-tiedosto komennolla:

```
$ cat wp.sql | mariadb --defaults-extra-file=/etc/mysql/mariadb.cnf -u root -p
```

Komennon syöttämisen jälkeen tulee ohjelmalle antaa **MariaDB:n root-käyttäjän salasana**, sillä komento suoritetaan tietokannassa kyseisenä käyttäjänä.

Euroopan unionin osarahoittama

Kun komentorivikonfigurointi on saatu valmiiksi, siirrytään selaimella viimeistelemään Wordpressin käyttöönotto. Tämä tulee toistaiseksi tehdä vielä palvelimen päässä, jonka selaimella siirrytään osoitteeseen: `http://localhost/wp-admin/install.php`.



Kuva 1: Wordpressin asennuksen informaatiolomake web-selaimessa

Lomake on hyvin yksinkertainen ja siinä asetetaan viimeiset käyttäjätunnukset. Nämä ovat ne tunnukset, joilla Wordpressin ohjausnäkymään päästään jatkossa kirjautumaan.

Lomake ehdottaa itse suhteellisen hyvää salasanaa, mutta tämä kannattaa vaihtaa parempaan. Lokaalissa ratkaisussa, kuten kuvassa, sähköpostikenttään voi laittaa jonkin toimimattoman sähköpostin. Asetuksia pääsee tarvittaessa muuttamaan myös jälkikäteen.

Wordpressin asennus on nyt viimeistelty. Wordpress Dashboard -ohjausnäkymään päästään kirjautumaan palvelimen selaimella, osoitteesta: `http://localhost/wp-admin/`

Toistaiseksi luodulla konfiguraatiolla sivustolle päästään ainoastaan siltä koneelta/palvelimelta, jossa Wordpress pyörii. Esimerkiksi virtuaalikoneen host-koneesta ei pääse sivustolle, vaan selaimeen tulee seuraavankaltainen ilmoitus:

```
Neither /etc/wordpress/config-192.168.122.4.php nor
/etc/wordpress/config-168.122.4.php could be found. Ensure one of them exists,
is readable by the webserver and contains the password/username.
```

Koska SSL-suojattua yhteyttä ei ole vielä otettu käyttöön Apache2:ssa, ei pääsyn sallimista verkkosivustolle palvelimen ulkopuolelta voi suositella. Tunnistautuessa Wordpressin Dashboardiin, on muun muassa admin-käyttäjän salasana luettavissa suoraan verkkoliikenteestä.

Lähteet:

https://wiki.debian.org/WordPress

https://mariadb.com/kb/en/configuring-mariadb-with-option-files/

Euroopan unionin osarahoittama

Palvelimen lisäohjelmistot ja konfiguraatiot

Vaikka Wordpress itsessään toimii melko moitteetta raa'an asennuksen jälkeen, ei mahdollista Linux-palvelinta tai -työasemaa voida pitää valmiina kehityskäyttöön. Tarpeen vaatiessa voi olla ehdotonta ottaa käyttöön erilaisia lisäohjelmistoja ja turvallisuuskonfiguraatioita.

Jos kehitystyötä tehdään ainoastaan yhdellä työasemalla, ei todennäköisesti ole tarvetta asentaa esimerkiksi etäyhteysohjelmistoja. On kuitenkin hyvin todennäköistä että sivuston parissa työskentelee useampi henkilö. Tällöin on järkevämpää valjastaa jokin oma laite kehitysaikaiseksi palvelimeksi, jolloin turvallisten etäyhteysohjelmistojen käyttöönotto tulee tarpeeseen.

Joka tapauksessa, riippumatta kehitysalustasta, tulee käyttöjärjestelmän ja laitteiston turvallisuuteen kiinnittää huomiota. Vanha ajatus siitä että Linux-pohjaiset käyttöjärjestelmät ovat itsenäisesti turvallisia on ehdottomasti vanhentunut. Suosion suuri kasvu tarkoittaa myös uhkaympäristön kasvua. Allekirjoittaneen omaan julkiseen palvelimeen kohdistui kymmeniä yhteysyrityksiä ensimmäisen tunnin aikana käyttöjärjestelmän asentamisesta. Linux-maailmassa on myös ymmärrettävä se, että käyttäjä on itse täydessä vastuussa työasemastaan tai palvelimestaan. Jakelusta riippuen, edes palomuurin hallinta ei välttämättä ole automaattisesti käytössä.

Tästä johtuen tässä isossa osiossa tullaan käymään läpi erilaisten hyöty- ja turvallisuusohjelmistojen asentaminen ja näiden oleellisimmat konfiguraatiot. Ohjelmien käyttötarkoitus pyritään avaamaan, jonka lisäksi työskentelyn aikana selostetaan suoritettavia komentoja ja konfiguraatiota.

Lisäohjelmistojen valinnassa ja konfiguraatioratkaisuissa punainen lanka on Wordpress-web-palvelimen turvaamisessa. Tästä huolimatta jokainen moduuli toimii erillisenä ohjeistuksenaan. Tarpeen vaatiessa, ohjeistusta mukailemalla on mahdollista konfiguroida samat palvelut vastaamaan omia tarpeitaan, kunhan muuttaa tehtäviä konfiguraatioita näin vaadittaessa.

Etäyhteys

Palvelinten kanssa työskennellessä etäyhteyden käyttäminen on lähes poikkeuksetta pakollista. Jos käytössä on vaikkapa Wordpress-sivustoa varten vuokrattu hostauspalvelu, saattaa itse palvelin olla täysin eri maassa kuin itse käyttäjät ja kehittäjät. Yleensä näissä tapauksissa palveluntarjoaja on luonut jonkinlaisen automaattisen etäyhteyspalvelimen ja -konfiguraatiot. Tarpeen vaatiessa näiden muokkaaminen omiin tarpeisiin saattaa olla hyödyllistä.

Vaikka kehitysaikainen palvelin olisi fyysisesti samassa tilassa kehittäjien kanssa, on etäyhteys silti kätevä työkalu tämän hallintaan. Olisi varsin ärsyttävää juosta oman

työpisteen ja palvelimen välillä, jotta palvelimella voi ajaa tarvittavia komentoja ja konfiguraatioita.

Tässä osiossa otamme käyttöön OpenSSH-palvelimen, käymme läpi tämän konfigurointia ja turvaamme yhteyden avainpari-autentikoinnilla. Vastaavasti tiedostonsiirtoa varten otetaan käyttöön OpenSSH:n oma SFTP-tiedostonsiirtomekanismi.

OpenSSH

OpenSSH on avoin työkalusetti Secure Shell (SSH) protokollan käyttöön, joka takaa salatun viestinnän kahden laitteen välillä. SSH:n yli voidaan turvallisesti käyttää mm. palvelimen konsolia etänä ja siirtää tiedostoja työaseman ja palvelimen välillä. Tämä on yksi oleellisimmista työkaluista kaikille web-kehittäjille, jotka työskentelevät erillisen palvelimen kanssa.

Tässä osiossa käydään läpi OpenSSH-Serverin asennus palvelimelle, sen turvalliset konfiguraatiot ja miten siihen voidaan ottaa yhteyttä.

Esimerkki on tehty Debian 12 -jakelulla varustetussa virtuaalikoneessa. OpenSSH on hyvin universaali ohjelmisto, mutta jotkin osat konfiguraatioissa saattavat erota eri jakeluiden välillä. Esimerkiksi tiedostopolut saattavat erota toisistaan.

Asentaminen

Aloitetaan asentamalla `openssh-server` -paketti riippuvuuksineen palvelimelle:

```
$ apt update && apt upgrade -y
$ apt install openssh-server
```

`sshd.service` saattaa olla automaattisesti päällä paketin asennuksen jälkeen. Koska turvallisia konfiguraatioita ei ole vielä tehty, laitetaan se pois päältä:

```
$ systemctl stop sshd
```

Konfigurointi

Seuraavaksi tehdään `sshd`:lle konfiguraatiot. Nämä löytyvät hakemistosta `/etc/ssh` . Hakemistosta löytyvät tiedostot `ssh_config` ja `sshd_config`. Ensin mainittu on työaseman asiakasohjelmiston konfiguraatiotiedosto, jälkimmäinen SSH-palvelimen konfiguraatiotiedosto. Tässä vaiheessa konfiguroimme palvelinta.

Euroopan unionin osarahoittama

Oletuskonfiguraatiotiedostosta kannattaa ottaa varmuuskopio, jotta mahdolliset virheet voidaan perua. Varmuuskopiotiedostosta poistetaan kaikilta käyttäjäryhmiltä kirjoitusoikeudet, jotta sitä ei ylikirjoiteta vahingossa.

```
$ cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
$ chmod a-w /etc/ssh/sshd_config.original
```

Konfiguraatiot tiedostoon voidaan tehdä komentoriviteksteditorilla, tässä tapauksessa nano:lla:

```
$ nano /etc/ssh/sshd_config
```

Konfiguraatiotiedostosta löytyy paljon asetuksia. Näistä ylivoimaisesti suurin osa on kommentoitu. Kommentoitu asetus on oletuksena päällä. Jos näitä asetuksia halutaan muuttaa, tulee kommentin aktivoiva # -merkki poistaa ja asettaa vaihtoehtoinen asetus. Esimerkkitapauksessa konfiguraatio muutettiin seuraavaksi, tärkeimmät muutokset **#selitettynä**:

```
 1 # This is the sshd server system-wide configuration file.  See
 2 # sshd_config(5) for more information.
 3
 4 # This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games
 5
 6 # The strategy used for options in the default sshd_config shipped with
 7 # OpenSSH is to specify options with their default value where
 8 # possible, but leave them commented.  Uncommented options override the
 9 # default value.
10
11 Include /etc/ssh/sshd_config.d/*.conf
12
13 Protocol 2    #SSH:n käyttämä protokollaversio. 2 on ainoa jota suositellaan
14                käytettävän nykypäivän järjestelmissä.
15 Port 22022    #Käytössä oleva portti
16 #AddressFamily any  #SSH tukee sekä IPv4 että Ipv6 -osoitteita.
17 #ListenAddress 0.0.0.0
18 #ListenAddress ::
19
20 #HostKey /etc/ssh/ssh_host_rsa_key
21 #HostKey /etc/ssh/ssh_host_ecdsa_key
22 #HostKey /etc/ssh/ssh_host_ed25519_key
23
24 # Ciphers and keying
25 #RekeyLimit default none
```

```
26
27 # Logging
28 SyslogFacility AUTH
29 LogLevel INFO
30
31 # Authentication:
32
33 #LoginGraceTime 2m
34 PermitRootLogin no  #Root-käyttäjänä kirjautumista ei sallita!
35 #StrictModes yes
36 MaxAuthTries 6      #Korkeintaan 6 kirjautumisyritystä.
37 MaxSessions 5       #Korkeintaan 5 samanaikaista sessiota.
38
39 PubkeyAuthentication yes  #Ottaa käyttöön avainparitunnistautumisen.
40
41 # Expect .ssh/authorized_keys2 to be disregarded by default in future.
42 #AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
43
44 #AuthorizedPrincipalsFile none
45
46 #AuthorizedKeysCommand none
47 #AuthorizedKeysCommandUser nobody
48
49 # For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
50 #HostbasedAuthentication no
51 # Change to yes if you don't trust ~/.ssh/known_hosts for
52 # HostbasedAuthentication
53 #IgnoreUserKnownHosts no
54 # Don't read the user's ~/.rhosts and ~/.shosts files
55 #IgnoreRhosts yes
56
57 # To disable tunneled clear text passwords, change to no here!
58 PasswordAuthentication yes     #Salasanatunnistautuminen vielä
59 PermitEmptyPasswords no         toistaiseksi käytössä...
60
61 # Change to yes to enable challenge-response passwords (beware issues with
62 # some PAM modules and threads)
63 KbdInteractiveAuthentication no
64
65 # Kerberos options
66 #KerberosAuthentication no
```

```
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
UsePAM no           #Käytössä ei ole tätä PAM:ia tukevia moduuleita.

AllowAgentForwarding no   #Pyritään rajoittamaan haitallista toimintaa.
AllowTcpForwarding no      Käytännössä näillä ei valtavaa merkitystä.
#GatewayPorts no
X11Forwarding no    #X11-näyttöpalvelimen jakamiselle ei tarvetta.
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no        #Message of the Day:lle ei nyt tarvetta
#PrintLastLog yes
TCPKeepAlive yes    #TCP-yhteys pyritään pitämään elossa
#PermitUserEnvironment no
#Compression delayed
ClientAliveInterval 15
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none
```

```
108
109 # no default banner path
110 #Banner none
111
112 # Allow client to pass locale environment variables
113 # AcceptEnv LANG LC_*
114 AcceptEnv no  #Emme salli ympäristömuuttujien välittämistä.
115
116 # override default of no subsystems
117 Subsystem     sftp  /usr/lib/openssh/sftp-server
118
119 # Example of overriding settings on a per-user basis
120 #Match User anoncvs
121 # X11Forwarding no
122 # AllowTcpForwarding no
123 # PermitTTY no
124 # ForceCommand cvs server
```

Avainpohjainen tunnistautuminen palvelimelle

Kun konfiguraatiotiedosto on saatu valmiiksi, siirrytään sen käyttäjän kotihakemistoon jolla SSH:ta tullaan käyttämään. Tässä tapauksessa käyttäjä on nimeltään ’`aim`’. Kotihakemistosta pitäisi löytyä `.ssh` -niminen piilotettu hakemisto:

```
$ ls -alh
total 116K
drwxr-xr-x 15 aim  aim  4,0K 15. 5. 11:39 .
drwxr-xr-x  3 root root 4,0K  7. 5. 14:38 ..
...
drwx------  2 aim  aim  4,0K 15. 5. 11:48 .ssh
...
```

Jos hakemistoa ei ole, se voidaan luoda komennolla:

```
$ mkdir -p ~/.ssh
```

`.ssh` -hakemistoon luodaan tiedosto, jota käytetään avainpari-pohjaiseen tunnistautumiseen. Tiedostoon tullaan tallentamaan avainparin **julkinen pari**.

```
$ touch ~/.ssh/authorized_keys
```


Euroopan unionin osarahoittama

Tässä vaiheessa palvelimen `sshd.service` tulee asettaa aktiiviseksi:

```
$ systemctl start sshd
```

Avainpari-pohjaista tunnistautumista varten tulee avainpari luoda **sillä laitteella, josta luodaan etäyhteys palvelimelle.**

Avainparin luonti Linuxilla

Asennetaan `openssh-client` -paketti jakelun ohjelmistorepositoriosta:

```
$ apt install openssh-client
```

Luodaan omaan kotihakemistoon `.ssh` -hakemisto:

```
$ mkdir -p ~/.ssh
```

Luodaan avainpari `ssh-keygen` -ohjelmalla, tässä tapauksessa `ed25519`-muodossa. Ohjelma pyytää asettamaan avaimelle salasanan, **jonka on syytä olla vahva**:

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/YourUser/.ssh/id_ed25519): /home/YourUser/.ssh/AIMlearning_Key
Enter passphrase for "/home/YourUser/.ssh/AIMlearning_Key" (empty for no passphrase): [aseta salasana]
Enter same passphrase again: [varmista salasana]
Your identification has been saved in /home/YourUser/.ssh/AIMlearning_Key
Your public key has been saved in /home/YourUser/.ssh/AIMlearning_Key.pub
```

Kun avainpari on luotu, tulee **JULKINEN** avain kopioida palvelimelle tehtyyn `~/.ssh/authorized_keys` -tiedostoon. Julkisen avaimen tunnistaa `.pub` -päätteestä. Tähän on kaksi tapaa:

Tapa 1

On mahdollista syöttää **JULKINEN** avain kohdepalvelimeen täysin SSH:n välityksellä. Tämä kuitenkin vaatii, että palvelimen `sshd_config` tiedostoon jätettiin seuraava konfiguraatio sallituksi:

```
...
58 PasswordAuthentication yes
...
```

Omalla koneella voidaan nyt ajaa seuraava komento, jossa palvelimen SSH-portti ja IP-osoite vastaa oikeaa:

```
$ ssh-copy-id -i .ssh/AIMlearning_Key.pub -p 22022 aim@192.168.122.197
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"AIMlearning_Key.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
aim@192.168.122.197's password: [syötä palvelimen käyttäjän salasana]

Number of key(s) added: 1

Now try logging into the machine, with: "ssh -i .ssh/AIMlearning_Key -p 22022
'aim@192.168.122.197'"
and check to make sure that only the key(s) you wanted were added.
```

Komennon selitykset:

`-p`: palvelimelle konfiguroitu ssh-portti

`-i`: paikallinen julkinen avain -tiedosto

`aim@192.168.122.197`:
palvelimella_käytettävä_käyttäjänimi@palvelimen_IP-osoite

Ohjelma kysyy **palvelimen käyttäjän salasanaa**. Tällä tunnistaudutaan palvelimelle SSH:n yli, jolloin ohjelma pääsee syöttämään julkisen avaimen automaattisesti.

Nyt palvelimelle pääsee kirjautumaan SSH:n välityksellä avainparia käyttäen. Tällä kertaa ohjelmalle annetaan **YKSITYINEN AVAIN** :

```
$ ssh -i .ssh/AIMlearning_Key -p 22022 aim@192.168.122.197
Enter passphrase for key '.ssh/AIMlearning_Key': [kirjoita avaimen salasana]
```

Komennon suorittamisen jälkeen ohjelma varmistaa yksityisen avaimen salasanan. Tämän hyväksyttyään SSH varmistaa avainten vastaavuuden, jonka toteuduttua palvelimelle kirjaudutaan sisään.

Aivan ensimmäisenä palvelimen `/etc/ssh/sshd_config` -tiedostossa kannattaa estää puhdas salasana-autentikointi:

```
...
58 PasswordAuthentication no
...
```


Euroopan unionin osarahoittama

Jonka jälkeen `sshd` -palvelu käynnistetään uudelleen:

```
$ systemctl restart sshd
```

Nyt SSH-yhteys on valmiina käytettäväksi ja palvelimen komentoriviä voi käyttää tavalliseen tapaan.

Tapa 2

JULKISEN avaimen voi kopioida palvelimelle manuaalisesti, muistitikun välityksellä tai vastaavalla tavalla.

Oman avaimen saa konsoliin näkyviin komennolla:

```
$ cat ~./ssh/avaimen_nimi.pub
```

Avain on suhteellisen pitkä, joten jos minkäänlainen suora kopiointi on mahdollista, on se suositeltavaa. Esimerkkitapauksessa palvelin oli virtuaalikone, johon tavallinen copy-paste toimi.

Avain kopioidaan kokonaisuudessaan palvelimen `~/.ssh/authorized_keys` -tiedoston ensimmäiselle riville. Jos avaimia luodaan jatkossa lisää, ne asetetaan kaikki omille riveilleen. Muuta dataa tiedostoon ei tule.

Lisäksi palvelimen `/etc/ssh/sshd_config` -tiedostossa kannattaa estää puhdas salasana-autentikointi:

```
...
58 PasswordAuthentication no
...
```

Jonka jälkeen `sshd` -palvelu käynnistetään uudelleen:

```
$ systemctl restart sshd
```

Nyt SSH-yhteys on valmiina käytettäväksi ja palvelimen komentoriviä voi käyttää tavalliseen tapaan.

Avainparin luonti Windowsilla - PuTTY

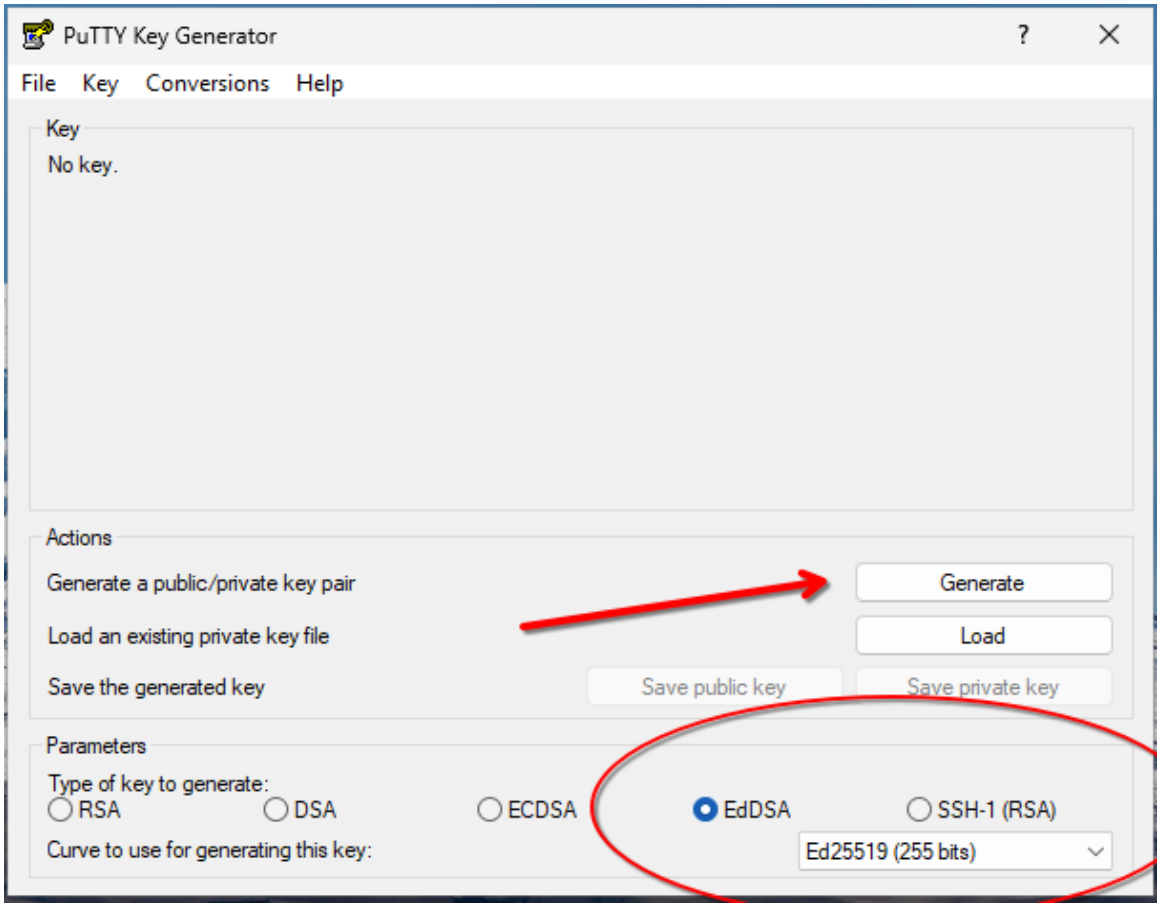
Windowsille on saatavilla useita erilaisia SSH-ohjelmia. Windowsista itsestään löytyy sisäänleivottu OpenSSH Client, johon pääsee käsiksi komentoriviltä. Toiminnaltaan tämä vastaa lähes täysin yleisesti BSD- ja Linux-käytössä olevaa versiota OpenSSH:sta.

Allekirjoittaneen mielestä Windowsia on kuitenkin mukavampi käyttää graafisesti. Seuraava ohje on siis tehty sitä silmällä pitäen.

Yksi yleisimmistä graafisen käyttöliittymän SSH-ohjelmistoista on PuTTY. Kyseisen ohjelmistopaketin saa ladattua osoitteesta: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

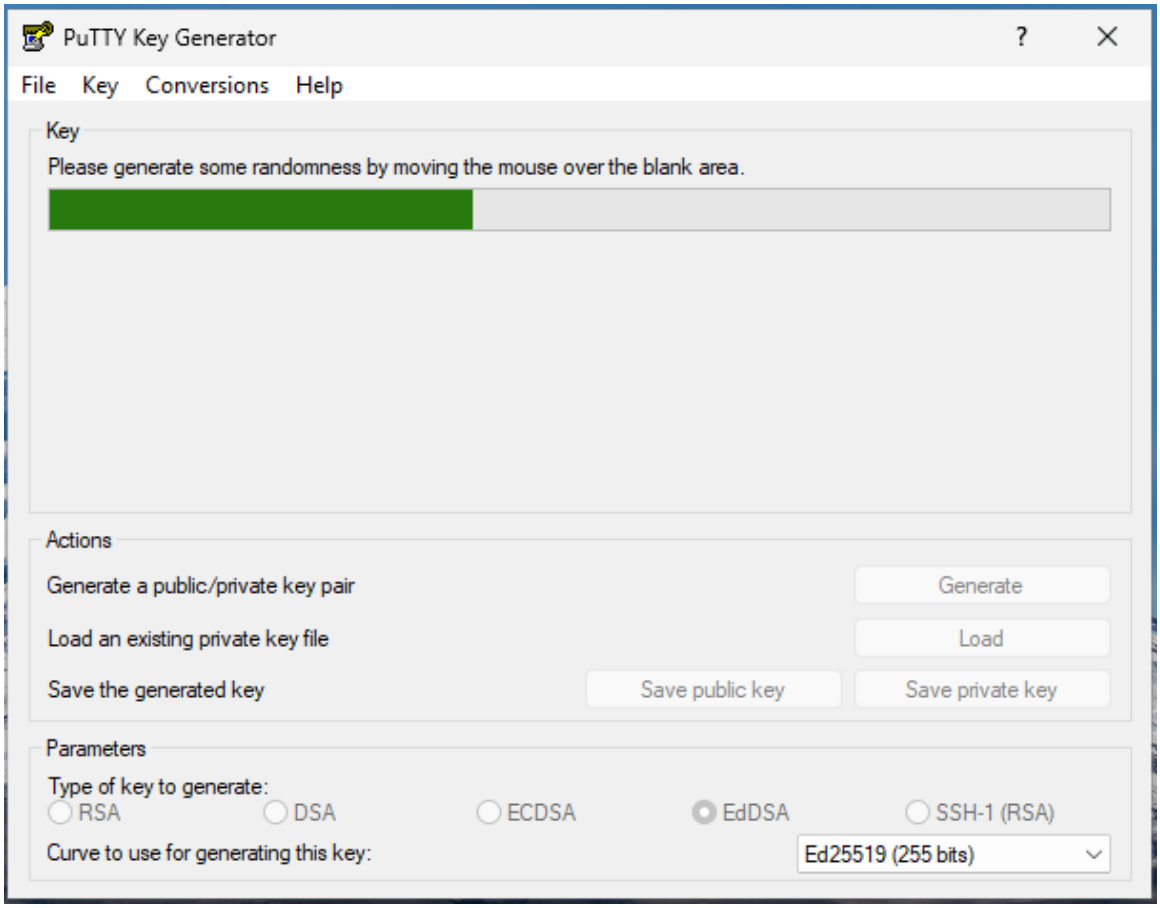
Koko ohjelmistopaketin ladatessa saa käyttöönsä sekä itse SSH-clientin, että PuTTY Key Generatorin jolla avainparit saadaan generoitua.

Generoidaan avainpari PuTTY Key Generaattorilla:



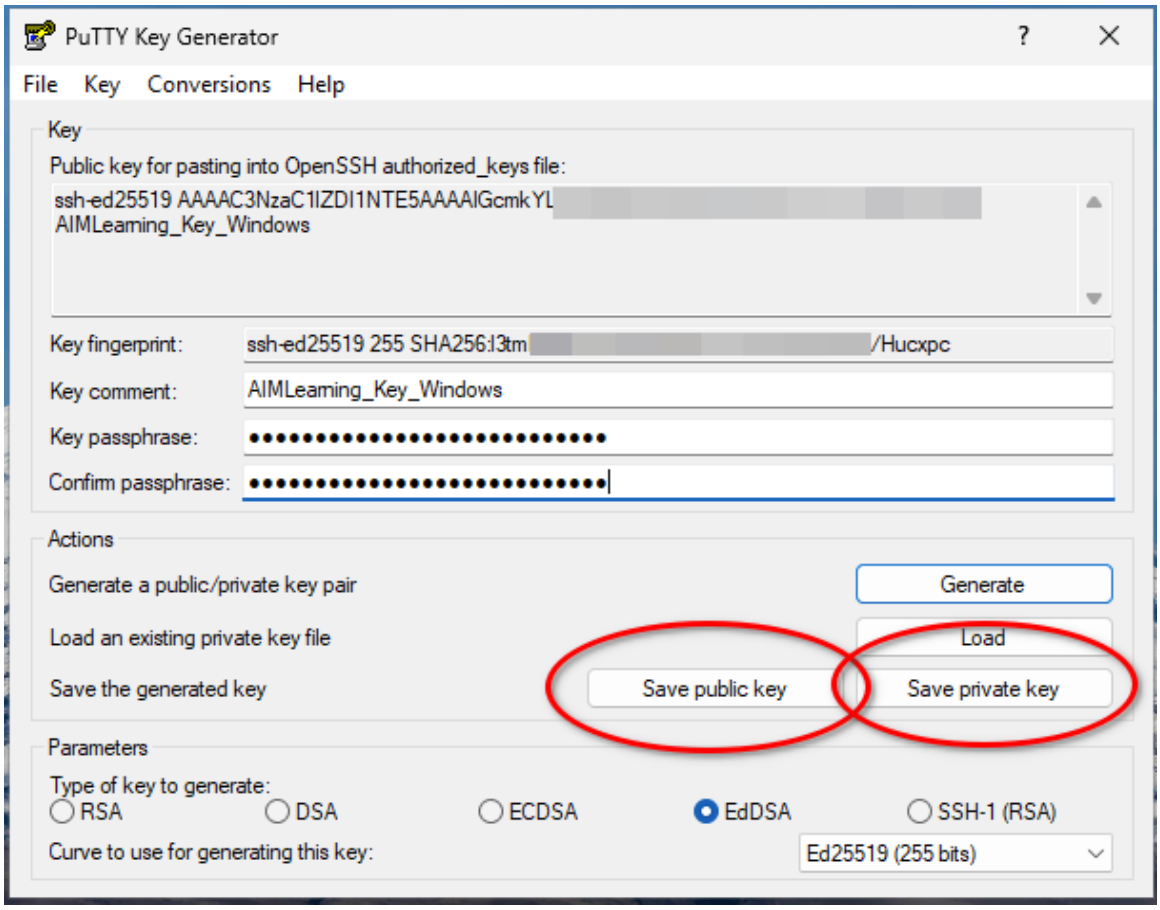
Kuva 2: PuTTY Key Generatorin avainten asetukset

Euroopan unionin osarahoittama



Kuva 3: PuTTY Key Generator luo avainta satunnaisuuden avulla

Kun avainpari on generoitu, asetetaan sille haluttu kommentti josta avain voidaan myöhemmin tunnistaa ja **vahva salasana**. Avainparin molemmat osat tallennetaan haluttuun sijaintiin:

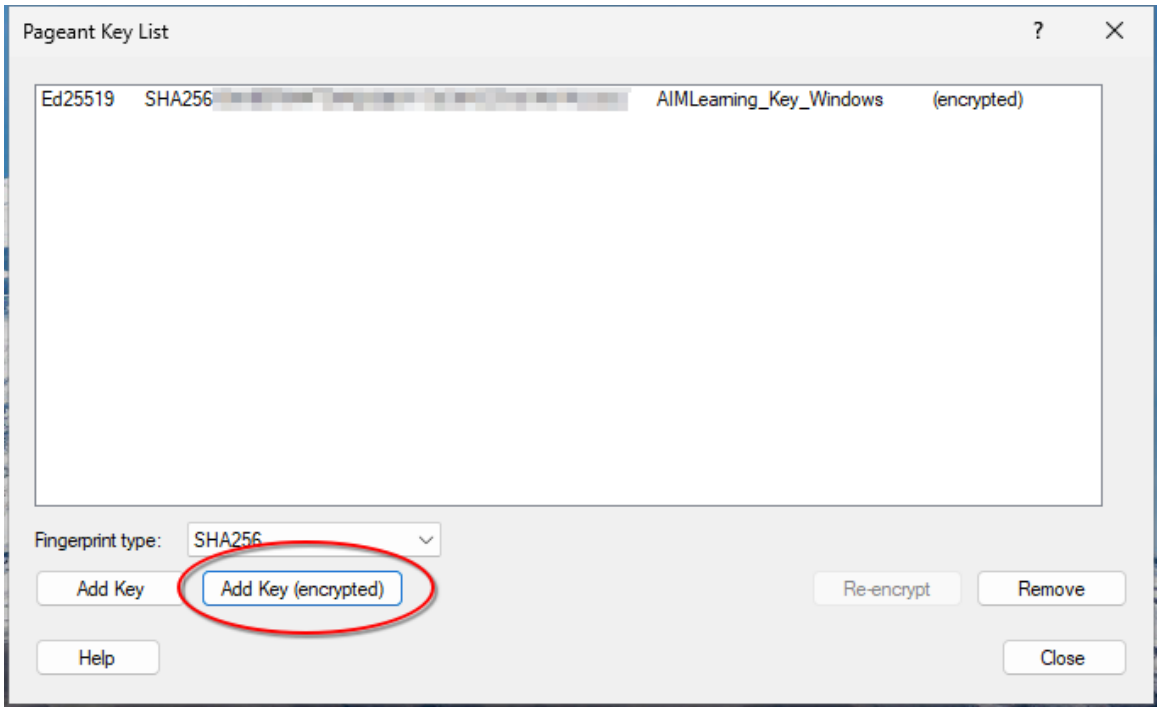


Kuva 4: PuTTY Key Generatorissa asetetaan salasana avaimelle; avainpari tallennetaan

Euroopan unionin osarahoittama

Avainparia voi käyttää suoraan PuTTY:n kanssa ilman kummempia konfiguraatioita. Jos kuitenkin ladattiin koko ohjelmistopaketti, tulee sen mukana myös Pageant, jolla voidaan hallita yksityisiä avaimia keskitetysti.

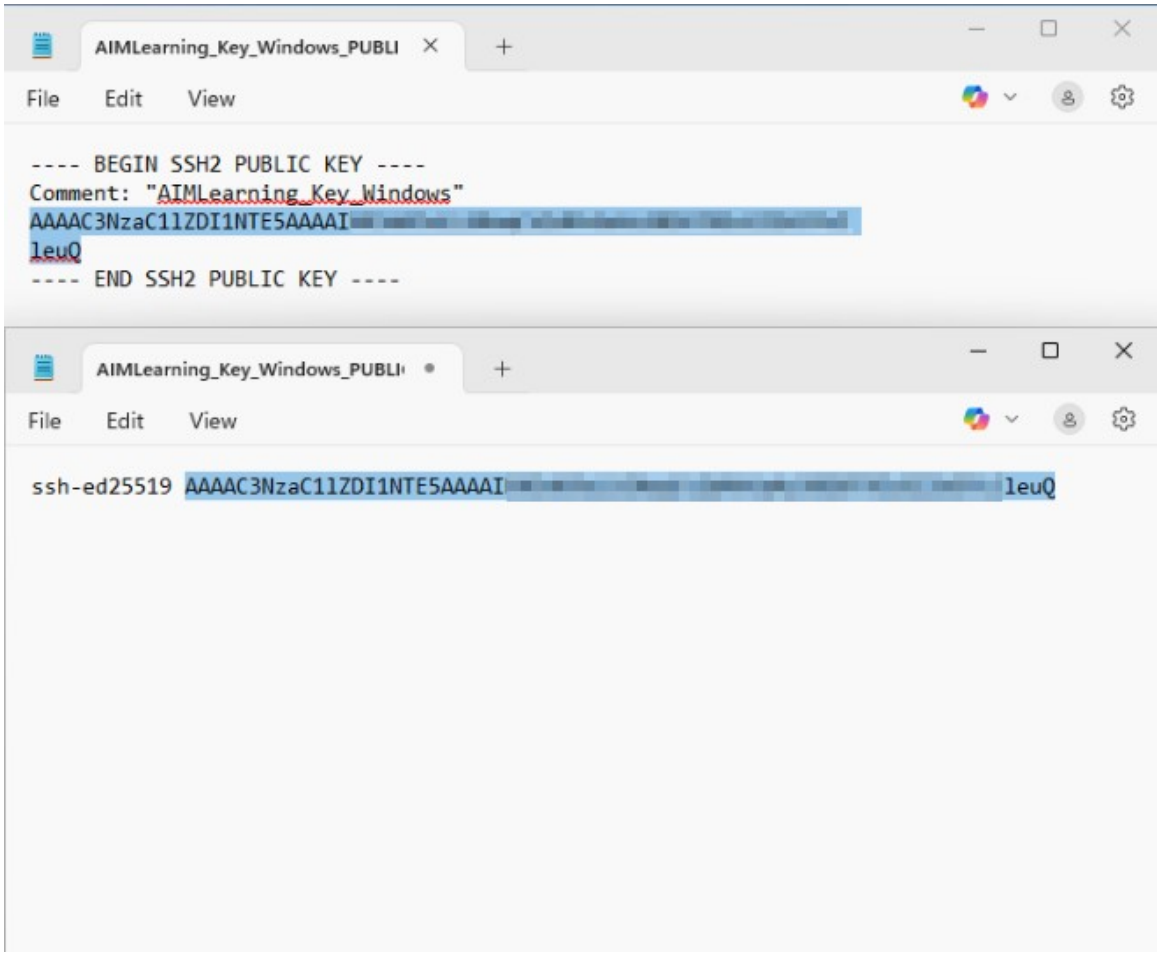
Lisätään yksityinen avain Pageantiin salatussa muodossa:



Kuva 5: Salattu yksityinen avain lisätään Pageantiin

Oman avaimen saa näkyviin avaamalla tiedoston vaikkapa Notepadissa. Tämä tulee myös muuttaa muotoon, jonka `openssh-server` osaa pureskella palvelimella:

Tehdään ensin kopio julkisen avaimen tiedostosta. Sitten avataan kopioitu tiedosto ja muutetaan se seuraavalla tavalla:



Kuva 6: PuTTY Key Generatorin luoma julkinen avain muutetaan OpenSSH:n käsittämään muotoon

Ylempi tiedosto on alkuperäinen. Alempi se, mihin muotoon haluamme sen muuttaa.

Kun avainpari on luotu, tulee **JULKISEN OPENSSH-YHTEENSOPIVAN** avaintiedoston sisältö kopioida palvelimelle tehtyyn `~/.ssh/authorized_keys` -tiedostoon. Tähän on kaksi tapaa:

Tapa 1

JULKISEN avaimen voi kopioida manuaalisesti, muistitikun välityksellä tai vastaavalla tavalla.

Avain on suhteellisen pitkä, joten jos minkäänlainen suora kopiointi palvelimelle on mahdollista, on se suositeltavaa.

Euroopan unionin osarahoittama

Esimerkkitapauksessa palvelin oli virtuaalikone, johon tavallinen copy-paste toimi.

Avain kopioidaan kokonaisuudessaan palvelimen `~/.ssh/authorized_keys` -tiedoston ensimmäiselle riville. Jos avaimia luodaan jatkossa lisää, ne asetetaan kaikki omille riveilleen. Muuta dataa tiedostoon ei tule.

Lisäksi palvelimen `/etc/ssh/sshd_config` -tiedostossa kannattaa estää puhdas salasana-autentikointi:

```
...
58 PasswordAuthentication no
...
```

Jonka jälkeen `sshd` -palvelu käynnistetään uudelleen:

```
$ systemctl restart sshd
```

Tapa 2:

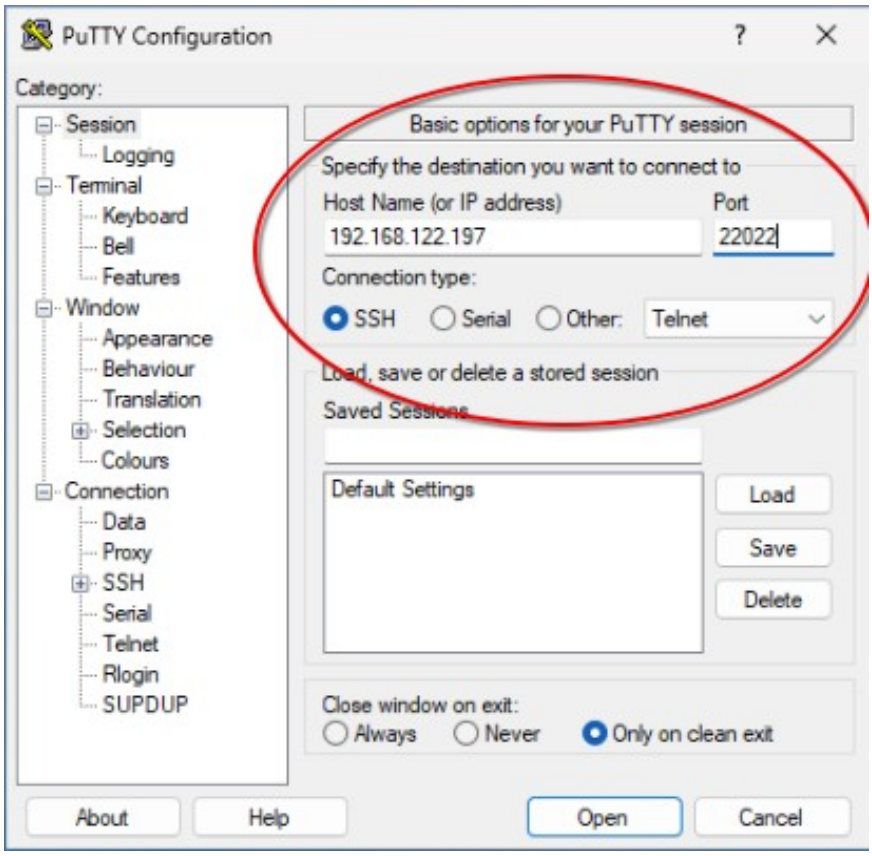
On mahdollista syöttää **JULKINEN** avain kohdepalvelimeen täysin SSH:n välityksellä. Tämä kuitenkin vaatii, että palvelimen `sshd_config` tiedostoon jätettiin seuraava konfiguraatio sallituksi:

```
...
58 PasswordAuthentication yes
...
```

Rivin muuttaminen ottaa käyttöön puhtaan salasana-autentikoinnin.

Tässä kohtaa voimme siirtyä ohjeistuksessa eteenpäin.

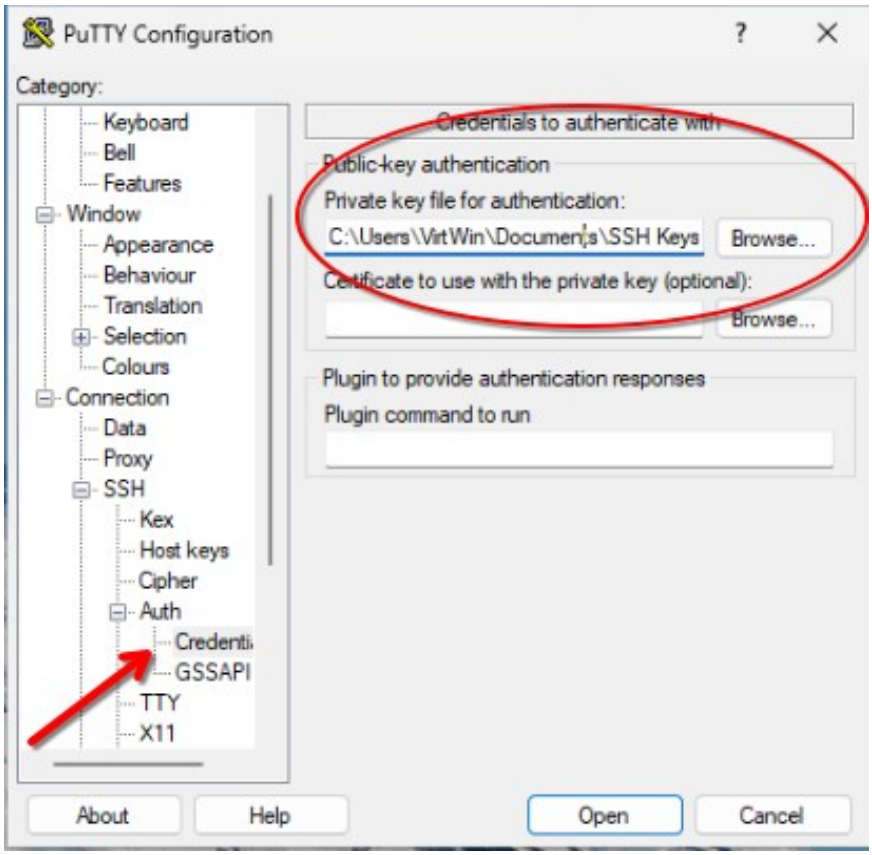
Palvelimeen voidaan nyt ottaa yhteyttä PuTTY:llä. Avataan PuTTY ja asetetaan siihen palvelimen IP-osoite, konfiguroitu portti ja varmistetaan että yhteystyyppi on SSH:



Kuva 7: PuTTY:llä asetetaan oikeat IP-osoite ja portti SSH-yhteyttä varten

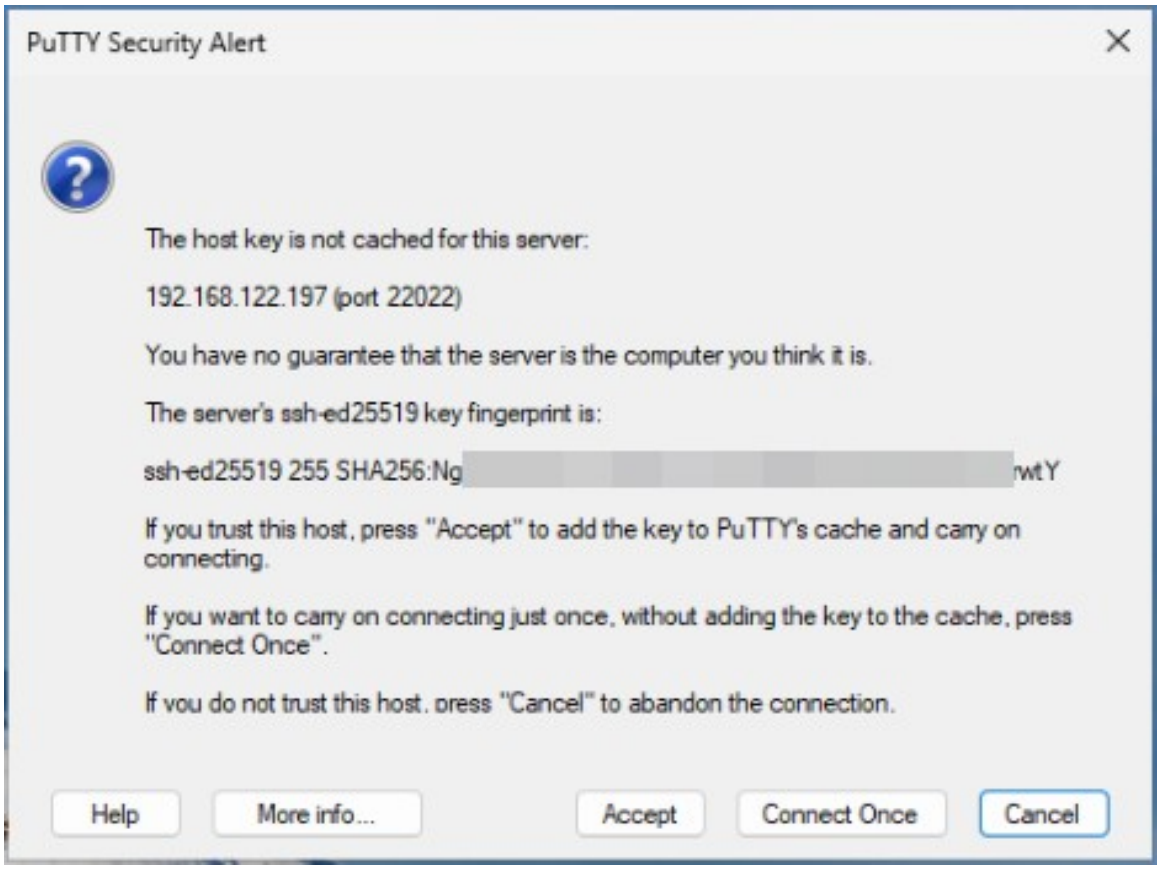
Euroopan unionin osarahoittama

Valitaan myös SSH-yhteydelle oikea **YKSITYINEN AVAIN** valikosta. Tämän voi tehdä vaikka julkista avainta ei ole vielä asetettu palvelimelle ja ollaan pelkän salasana-autentikoinnin varassa.



Kuva 8: PuTTY asetetaan käyttämään oikeaa yksityistä avainta

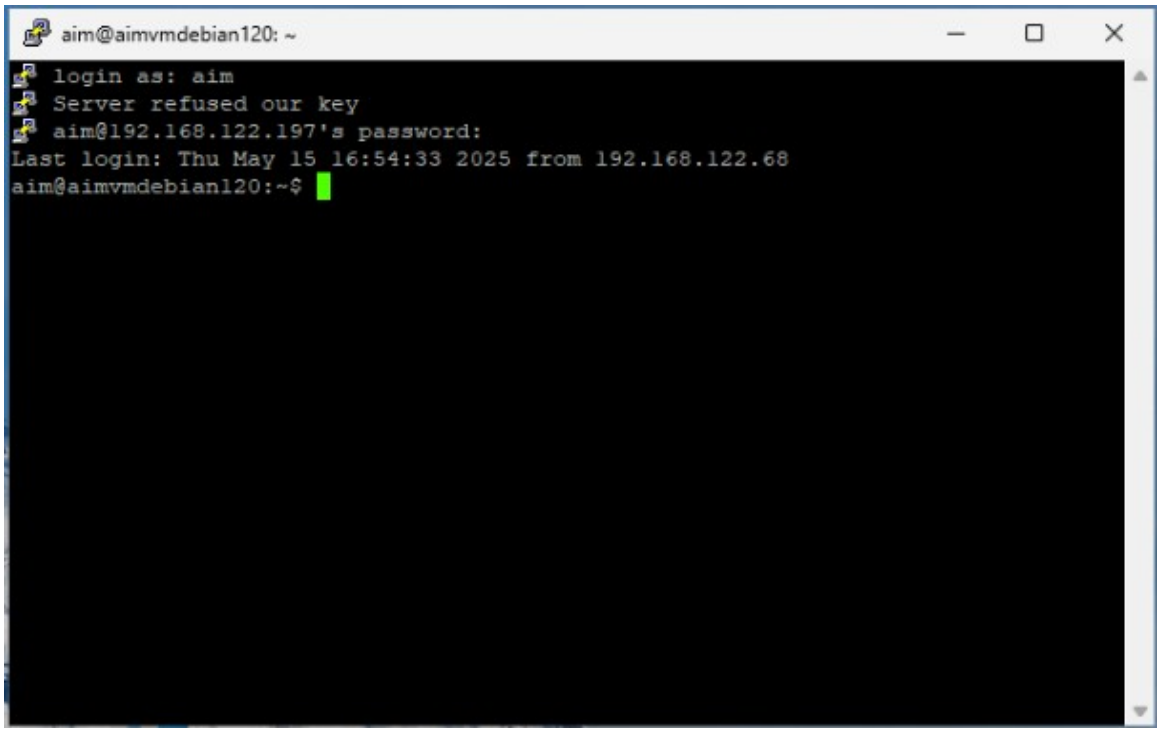
Painamalla Open-nappia avautuu yhteys palvelimelle. PuTTY antaa ilmoituksen siitä, ettei palvelimen tietoja ole vielä tallennettu, jolloin nämä ovat tuntemattomia. Koska luotamme palvelimeen, voimme hyväksyä avaimen lisäämisen välimuistiin.



Kuva 9: PuTTY varoittaa sille ennestään tuntemattomasta palvelimesta

Euroopan unionin osarahoittama

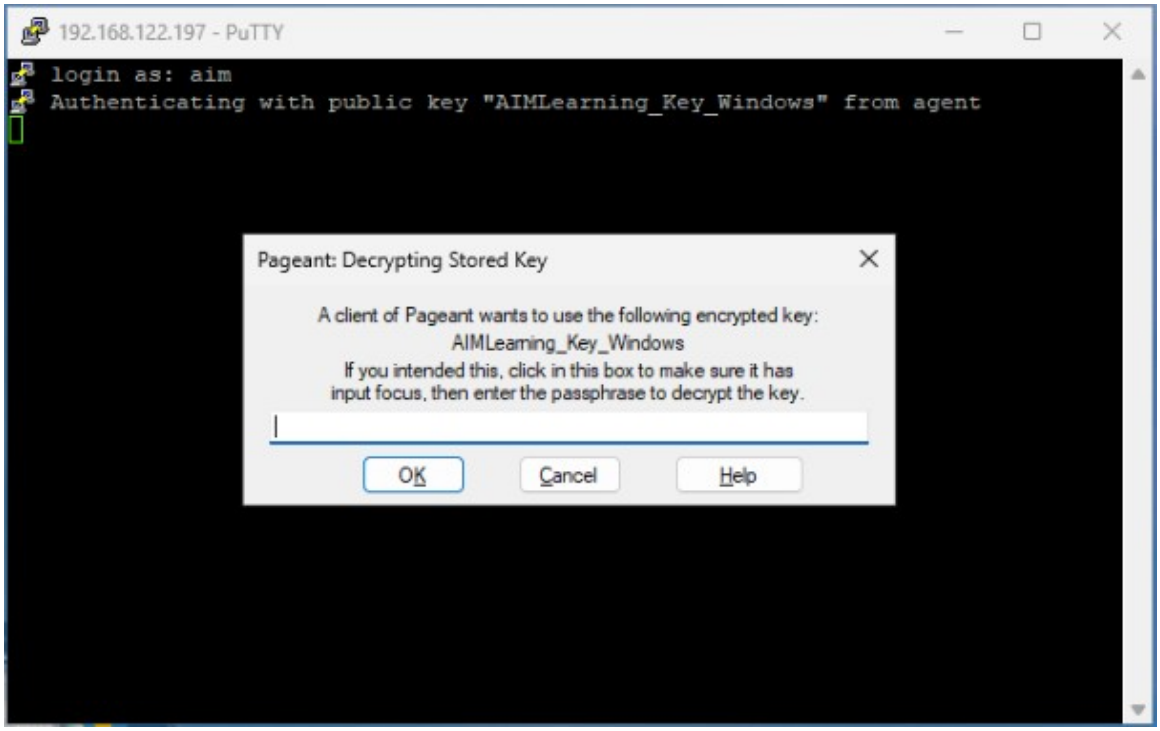
Jos ollaan pelkän salasana-autentikoinnin varassa, kirjautuminen näyttää seuraavalta:



Kuva 10: Salasanapohjainen autentikointi palvelimelle

Tässä vaiheessa on mahdollista käydä kopioimassa **OPENSSH-MUOTOINEN JULKINEN AVAIN** palvelimen `~/.ssh/authenticated_keys` -tiedostoon, jos sitä ei vielä ole tehty. PuTTY:llä tämä tapahtuu kopioimalla teksti Windowsin Notepadista, avaamalla SSH-yhteydellä tekstieditori palvelimella ja right-clickaamalla PuTTY:n terminaali-ikkunaa. Muistetaan myös poistaa salasana-autentikointi palvelimen `sshd_config` tiedostosta ja käynnistää `sshd.service` uudelleen!

Avainpohjainen kirjautuminen:



Kuva 11: Avainpohjaisen autentikoinnin yhteydessä tulee avata oma yksityinen avain

PuTTY kysyy ensin minä käyttäjänä palvelimelle kirjaudutaan. Tämän jälkeen alkaa autentikointi, jonka aikana Pageant pyytää **YKSITYISEN AVAIMEN** salasanaa. Kun tämä syötetään, pyrkii Pageant lukemaan avaimen tiedot avainparin varmistusta varten. Tämän onnistuttua autentikointi palvelimelle tapahtuu.

Lähteet:

https://documentation.ubuntu.com/server/how-to/security/openssh-server/index.html

https://linux.die.net/man/5/sshd_config
(man sshd_config)

https://www.howtogeek.com/devops/what-is-ssh-agent-forwarding-and-how-do-you-use-it/

https://www.ssh.com/academy/ssh/tunneling-example

https://manpages.ubuntu.com/manpages/questing/en/man5/sshd_config.5.html

https://servicenow.iu.edu/kb?id=kb_article_view&sysparm_article=KB0023919

Euroopan unionin osarahoittama

Tiedostonsiirto - SFTP

Vaatimukset: OpenSSH-Server asennettuna ja konfiguroituna

SFTP (SSH File Transfer Protocol) on SSH-protokollan yli toimiva tiedostonsiirtoprotokolla. Se tarjoaa SSH:n turvalliset ominaisuudet ja sitä voidaan käyttää ilman mitään muutoksia olemassa olevan SSH:n asennukseen. SFTP on käytännössä turvallinen vastine FTP:lle, jossa ei ole käytännössä mitään turvallisuusominaisuuksia.

SFTP:tä voidaan käyttää samoilla käyttäjätunnuksilla ja autentikaatiometodeilla kuin SSH:ta. On kuitenkin hyvä käytäntö luoda palvelimelle SFTP:tä varten oma käyttäjä, jonka oikeudet on rajoitettu ainoastaan tarpeellisiin toimiin.

Wordpressin tapauksessa voimme luoda käyttäjän, jolla on pääsyoikeudet ainoastaan Wordpressin sisältö-hakemistoon. Tällä käyttäjällä voimme tarvittaessa muokata tiedostoja suoraan, ilman suurempaa seikkailua SSH-yhteyden yli.

Tämä ohje olettaa että OpenSSH on toimintakuntoisena palvelimella ja siihen saadaan yhteys avainpari-autentikoinnin kautta.

Palvelimella SSH:n ja SFTP:n konfigurointi tapahtuu `/etc/ssh/sshd_config` -tiedostosta. Tiedosto näyttää seuraavankaltaiselta:

```
 1 # This is the sshd server system-wide configuration file.  See
 2 # sshd_config(5) for more information.
 3
 4 # This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games
 5
 6 # The strategy used for options in the default sshd_config shipped with
 7 # OpenSSH is to specify options with their default value where
 8 # possible, but leave them commented.  Uncommented options override the
 9 # default value.
10
11 Include /etc/ssh/sshd_config.d/*.conf
12
13 Protocol 2
14
15 Port 22022
16 #AddressFamily any
17 #ListenAddress 0.0.0.0
18 #ListenAddress ::
...
```

Tapauksessamme Wordpressin sisältökansio on hakemistossa `/var/lib/wordpress/wp-content`. Tämän omistaja on `www-data`, joka on monessa jakelussa web-palvelinten oletuskäyttäjä. Erillisen käyttäjän tarkoitus on mahdollistaa web-palvelimelle, kuten Apache2:lle, oikeus muuttaa tiedostoja tarpeen mukaan. Lisäksi erillisellä käyttäjällä voidaan eristää web-palvelimen tiedostot muusta järjestelmästä, joka on oleellinen asia järjestelmän kokonaisturvallisuuden kannalta. Jos tiedostot olisivat `root` -käyttäjän hallussa, mahdollinen hyökkäys web-palvelimelle avaisi helposti koko järjestelmän hyökkääjälle.

```
$ ls -alh /var/lib/wordpress/wp-content/
total 36K
drwxr-xr-x 6 www-data www-data 4.0K May 14 13:41 .
drwxr-xr-x 3 root     root     4.0K May  8 10:27 ..
-rw-r--r-- 1 www-data www-data   28 Jan  8  2012 index.php
drwxr-xr-x 5 www-data www-data  12K May 12 14:43 languages
drwxr-xr-x 2 www-data www-data 4.0K May 14 11:17 plugins
drwxr-xr-x 2 www-data www-data 4.0K May 14 11:17 themes
drwxr-xr-x 3 www-data www-data 4.0K May  8 12:08 uploads
```

Konfigurointi - SFTP-käyttäjä

Ennen SSH-konfiguraatioiden tekemistä luodaan uusi käyttäjä ja hakemisto, joihin toiminta SFTP:n yli rajoitetaan. Käyttäjän kotikansioon linkitetään Wordpressin sisältöhakemisto, eikä sille anneta pääsyä shell-ympäristöön. Uusi käyttäjä myös lisätään `www-data` -ryhmään, jotta tällä on oikeudet muokata Wordpress-hakemiston sisältöä. Tämä tehdään, jotta voimme rajoittaa mahdollista vahinkoa jota SFTP-käyttäjätunnusten ajautuminen vääriin käsiin aiheuttaa.

Käyttäjän hakemisto luodaan seuraavasti:

```
$ mkdir -p /var/sftp/files/aim-sftp
```

Luodaan itse käyttäjä:

```
$ useradd -d /var/sftp/files/aim-sftp -s /bin/false -G www-data aim-sftp
```

Komennon sisältö selitettynä:

`-d`: käyttäjän kotikansio

`-s`: käyttäjän shell, tässä tapauksessa /bin/false

`-G`: ryhmä, johon käyttäjä lisätään oman ryhmänsä lisäksi

Komennon lopussa uuden käyttäjän nimi.

Euroopan unionin osarahoittama

Uudelle käyttäjälle tulee myös luoda **VAHVA SALASANA**:

```
$ passwd aim-sftp
New password:
Retype new password:
passwd: password updated successfully
```

Voimme varmistaa onko ryhmät asetettu oikein:

```
$ id aim-sftp
uid=1001(aim-sftp) gid=1001(aim-sftp) groups=1001(aim-sftp),33(www-data)
```

Käytännöllisyyden vuoksi saattaa olla hyödyllistä lisätä palvelimen pääkäyttäjä samaan ryhmään SFTP-käyttäjän kanssa. ***Älä kuitenkaan unohda -a -flagia komennosta!*** *Ilman tätä kaikki muut ryhmät, mukaan lukien `sudo`, tulevat poistumaan käyttäjältä.*

```
$ usermod -aG aim-sftp aim
$ su aim
```

Konfigurointi - OpenSSH

SFTP-käyttäjän ollessa valmiina voimme siirtyä konfiguroimaan OpenSSH:ta. Konfigurointitiedosto on polussa `/etc/ssh/sshd_config`. Tiedostossa kannattaa ensimmäisenä varmistaa että seuraava (tai vastaava) rivi on kommentoitu:

```
...
117 #Subsystem       sftp    /usr/lib/openssh/sftp-server
...
```

Tämän rivin sijaan asetetaan toimintaan seuraava subsystem:

```
...
118 Subsystem       sftp    internal-sftp
...
```

`Subsystem / sftp-server` -konfiguraatio ottaa käyttöön OpenSSH-palvelimen täyden sisäisen SFTP-palvelimen. Alempana konfiguraatiossa määritellään käyttäjäkohtainen konfiguraatio, jossa pakotetaan käyttäjä käyttämään `Subsystem / internal-sftp`:tä. Internal-SFTP on täyden SFTP-palvelimen aliohjelmisto, jota voidaan käyttää itsenäisesti tai osana täyttä palvelinta.

Jos esimerkin kaltaisessa yhteydessä käytetään täyttä SFTP-palvelinta, ja Internal-SFTP:n käyttäminen jostain syystä epäonnistuu, ohjataan käyttäjä käyttämään täydellistä SFTP-palvelinta. Tällöin käyttäjä pääsee pakenemaan hakemistostaan, jolloin koko tiedostojärjestelmä on tämän saatavilla.

Jos SFTP-yhteys vaikuttaa epävakaalta, tai se ei toimi ollenkaan, voidaan kokeilla `Subsystem / sftp-server`:n sallimista.

Tarvitsemme myös SFTP-käyttäjien avainpohjaista autentikointia varten uuden avaintiedostopolun. Tämä voidaan lisätä oletuspolkujen kylkeen (kaikki samalle riville):

```
...
42 AuthorizedKeysFile   .ssh/authorized_keys .ssh/authorized_keys2
/etc/ssh/authorized_keys/%u
...
```

Lisäksi tiedoston lopusta löytyy esimerkki käyttäjäkohtaisten konfiguraatioiden tekemiseen:

```
...
119 # Example of overriding settings on a per-user basis
120 #Match User anoncvs
121 #       X11Forwarding no
122 #       AllowTcpForwarding no
123 #       PermitTTY no
124 #       ForceCommand cvs server
```

`Match User <nimimerkki>` -rivillä määritetään mille käyttäjälle seuraava konfiguraatio tehdään. Sitä seuraavat rivit ovat samoja konfiguraatioita kuin muussa tiedostossa, mutta nämä yliajavat yleiset konfiguraatiot. Tulemme hyödyntämään tätä SFTP-käyttäjän kanssa, rajoittaen tämän oikeuksia palvelimella.

Voimme ottaa SFTP-käyttäjällä käyttöön esimerkiksi seuraavan konfiguraation:

```
...
126 Match User aim-sftp
127         ForceCommand internal-sftp
128         ChrootDirectory /var/sftp/files/
129         X11Forwarding no
130         AllowTCPForwarding no
131         PermitTunnel no
132         AllowAgentForwarding no
```


Euroopan unionin osarahoittama

Konfiguraation selitykset:

`ForceCommand internal-sftp` asettaa käyttäjälle käyttöön OpenSSH:n sisäisen SFTP-alaohjelmiston. Tällä mahdollistetaan käyttäjän lukitseminen omaan juurihakemistoonsa.

`ChrootDirectory` asettaa juurihakemiston johon käyttäjä lukitaan. Periaatteessa tämä vaihtaa (change root) käyttäjän session aikaisen juurihakemiston haluttuun. `/var/sftp/files/` määrittää tämän olevan hakemisto, jossa käyttäjän oma kotihakemisto sijaitsee. Tässä konfiguraatiossa ei saa määritellä itse kotihakemistoa; chroot-hakemiston tulee olla root-käyttäjän omistama.

Muilla konfiguraatioilla varmistetaan ettei käyttäjä pääse, ainakaan helposti, pakenemaan asetetusta tilasta (lisätietoja: `man sshd_config`).

Voimme myös tehdä ryhmäkohtaisia konfiguraatioita asettamalla rivin `Match Group <ryhmän_nimi>`. Käytännössä nämä toimivat samalla tavalla kuin käyttäjäkohtaiset, mutta kokonaisille ryhmille.

Konfigurointi - Avainparin luominen

Jotta avainpohjainen autentikointi onnistuu, tulee ensin luoda tiedosto johon julkiset avaimet tallennetaan. Yleensä nämä avaimet tallennetaan käyttäjän kotihakemistoon, mutta tapauksessamme käyttäjällä ei ole varsinaista kotihakemistoa.

Luodaan julkisille avaimille hakemisto ja tiedosto aiemmin konfiguroimaamme polkuun:

```
$ mkdir -p /etc/ssh/authorized_keys
$ touch /etc/ssh/authorized_keys/aim-sftp
```

Avainparin saa luotua samalla tavalla kuin tavallisen SSH-yhteyden kanssa. Koska kyse on eri käyttäjistä, hyvä suositus on luoda tälle täysin oma avainpari.

Ennen muutoksia tiedoston oikeuksiin, tulee tähän asettaa oma **JULKINEN AVAIN**, esimerkiksi kopioimalla se terminaalista toiseen SSH-yhteyden ylitse. Kun avain on tiedostossa, muutetaan oikeudet hakemistoon ja tiedostoon.

Muutetaan tiedostojen omistajuus ja oikeudet:

```
$ chown :aim-sftp /etc/ssh/authorized_keys/aim-sftp
$ chmod 640 /etc/ssh/authorized_keys/aim-sftp
```

Voimme vielä varmistaa että kaikki omistajuudet ja oikeudet ovat oikein:

```
$ ls -alhR /etc/ssh/authorized_keys/
/etc/ssh/authorized_keys/:
total 12K
drwxr-xr-x 2 root root     4.0K Jun  3 16:08 .
drwxr-xr-x 5 root root     4.0K Jun  3 15:59 ..
-rw-r----- 1 root aim-sftp  100 Jun  3 16:08 aim-sftp
```

Jotta kaikki edellisissä vaiheissa tehdyt konfiguraatiot tulevat käyttöön, tulee `sshd.service` käynnistää nyt uudelleen.

```
$ systemctl restart sshd
```

Konfigurointi - Hakemistojen ja tiedostojen oikeudet

Käyttäjää luodessamme loimme SFTP-käyttäjälle oman hakemiston. Hakemisto on tällä hetkellä `root` -käyttäjän omistama. Joudumme siis lisäämään käyttäjän oman hakemistonsa omistajaksi.

```
$ chown aim-sftp:aim-sftp /var/sftp/files/aim-sftp/
```

Lopuksi varmistamme että SFTP-juurihakemistoissa on oikeat oikeudet. Käytännössä ainoastaan `root` -käyttäjälle varataan kirjoitusoikeus `/var/sftp/files` -hakemistoon.

```
$ chmod 755 /var/sftp/files/
$ ls -alhR /var/sftp/
/var/sftp/:
total 12K
drwxr-xr-x  3 root root 4.0K Jun  3 17:44 .
drwxr-xr-x 13 root root 4.0K Jun  3 17:44 ..
drwxr-xr-x  3 root root 4.0K Jun  3 17:53 files

/var/sftp/files:
total 12K
drwxr-xr-x 3 root     root     4.0K Jun  3 17:53 .
drwxr-xr-x 3 root     root     4.0K Jun  3 17:44 ..
drwxr-xr-x 2 aim-sftp aim-sftp 4.0K Jun  3 17:53 aim-sftp
```


Euroopan unionin osarahoittama

Kun olemme varmoja että oikeudet ovat oikein, voimme kirjautua palvelimelle SFTP-yhteydellä käyttämällä avainpohjaista autentikointia:

```
$ sftp -i .ssh/privaatti_avain -P 22022 aim-sftp@192.168.122.4
Enter passphrase for key '.ssh/privaatti_avain':
Connected to 192.168.122.4.
sftp>
```

Voimme nyt käyttää `/var/sftp/files/aim-sftp` -hakemistoa tiedostonsiirtoon palvelinten välillä:

```
sftp> pwd
Remote working directory: /
sftp> ls
aim-sftp
sftp> cd aim-sftp/
sftp> mkdir test_dir
sftp> ls -lh
drwxr-xr-x    ? 1001     1001         4.0K Jun  4 15:06 test_dir
```

Tarvittaessa SFTP:n käytössä olevat komennot saadaan listattua:

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp [-h] grp path                Change group of file 'path' to 'grp'
chmod [-h] mode path               Change permissions of file 'path' to 'mode'
chown [-h] own path                Change owner of file 'path' to 'own'
copy oldpath newpath               Copy remote file
cp oldpath newpath                 Copy remote file
...
```

Konfigurointi - wp-content hakemiston lisääminen SFTP-hakemistoon

SFTP-käyttäjän hakemisto toimii nyt kuten pitää ja yhteys on saatu. Tarkoituksena on kuitenkin sallia pääsy Wordpressin `wp-content` -hakemistoon SFTP:n ylitse.

Ensimmäisenä meidän tulee muuttaa `wp-content` -hakemiston oikeuksia rekursiivisesti niin, että `www-data` -ryhmässä olevat käyttäjät voivat myös kirjoittaa hakemistoon:

```
$ chmod g+w /var/lib/wordpress/wp-content/ -R
$ ls -alh
total 36K
drwxrwxr-x 6 www-data www-data 4.0K Jun  4 15:19 .
drwxr-xr-x 3 root     root     4.0K May  8 10:27 ..
-rw-rw-r-- 1 www-data www-data   28 Jan  8  2012 index.php
drwxrwxr-x 5 www-data www-data  12K May 12 14:43 languages
drwxrwxr-x 2 www-data www-data 4.0K May 14 11:17 plugins
drwxrwxr-x 2 www-data www-data 4.0K May 14 11:17 themes
drwxrwxr-x 3 www-data www-data 4.0K May  8 12:08 uploads
```

Koska SFTP-käyttäjä on chroot-jailissa, ei hakemistojen tavallinen linkitys toimi. On kuitenkin mahdollista lisätä SFTP-käyttäjän hakemistoon mount-piste, joka ikään kuin yhdistää halutut pisteet tietojärjestelmässä. Tämä ei kuitenkaan vapauta käyttäjää asettamastamme alueesta.

Periaatteessa tämän saavuttamiseen on kaksi eri tapaa. Suoraan komennolla mountaamalla, voimme saada aikaan mount-pisteen. **Tämä ei kuitenkaan säily, jos palvelin käynnistetään uudelleen.**

```
$ mkdir /var/sftp/files/aim-sftp/wp-content/
$ chown aim-sftp:aim-sftp /var/sftp/files/aim-sftp/wp-content/
$ mount --bind /var/lib/wordpress/wp-content /var/sftp/files/aim-sftp/wp-content/
```

Pysyvä tapa on lisätä `/etc/fstab` -tiedostoon käytännössä samat tiedot kuin mount -komennolle. Käynnistyksen yhteydessä käyttöjärjestelmä lukee tiedoston sisällön ja asettaa mount-pisteet automaattisesti.

Tiedosto sisältää jo valmiiksi tietoja, eikä näitä tule muuttaa. Tiedoston loppuun lisätään **korostettua** vastaava rivi:

```
...
 7 # <file system>             <mount point>  <type>  <options>  <dump>  <pass>
 8 UUID=885e8c21-3674-4b91-8532-9b16d68f56bb / ext4 defaults,noatime 0 1
 9 UUID=1721553b-c3ce-4b06-bc17-709107785578 swap swap defaults,noatime 0 0
10 /var/lib/wordpress/wp-content /var/sftp/files/aim-sftp/wp-content/ none bind
0 0
```


Euroopan unionin osarahoittama

SFTP:n ylitse päästään nyt navigoimaan wp-content -hakemistoon:

```
sftp> ls -lh aim-sftp/wp-content/
-rw-rw-r--    ? 33       33     28B Jan  8  2012 aim-sftp/wp-content/index.php
drwxrwxr-x    ? 33       33     12.0K May 12 14:43 aim-sftp/wp-content/languages
drwxrwxr-x    ? 33       33     4.0K May 14 11:17 aim-sftp/wp-content/plugins
drwxrwxr-x    ? 33       33     4.0K May 14 11:17 aim-sftp/wp-content/themes
drwxrwxr-x    ? 33       33     4.0K May  8 12:08 aim-sftp/wp-content/uploads
```

Toteutuksessa on kuitenkin rajoituksensa. Wordpress jakaa tiedostojensa sijainnin muutamaan eri hakemistoon. Monesti nämä tiedostot ovat täysin root-käyttäjän omistamia, eikä niihin varsinaisesti tulisi edes koskea.

Koska SFTP-käyttäjämme on chroot-vangittuna, ei tällä pääse symlinkkien kautta sille asetetun alueen ulkopuolelle:

```
sftp> ls -lh aim-sftp/wp-content/themes/
lrwxrwxrwx    ? 0        0             56B May  8  2024
aim-sftp/wp-content/themes/twentytwentythree
sftp> cd aim-sftp/wp-content/themes/twentytwentythree
realpath /aim-sftp/wp-content/themes/twentytwentythree: No such file
```

Todellisuudessa kyseinen tiedosto on olemassa, mutta ei SFTP-käyttäjälle. SSH-yhteyden kautta pääsemme tarvittaessa käsiksi tiedostoon mistä tahansa sijainnista:

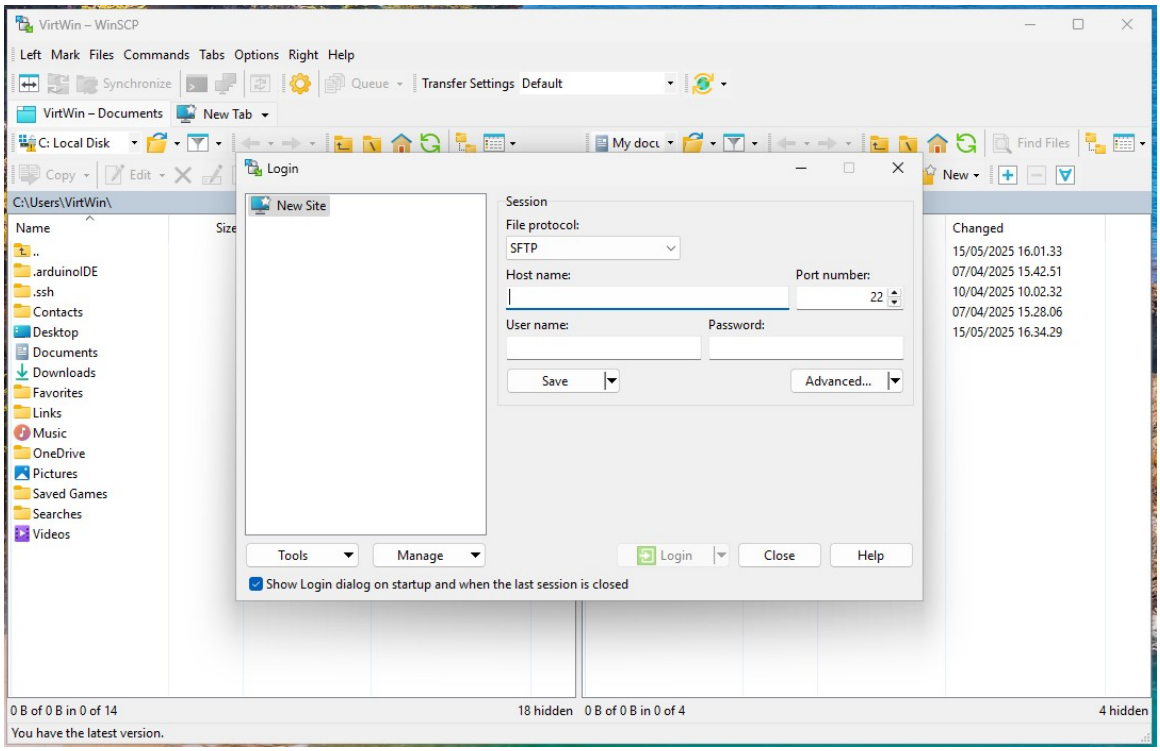
```
$ ls -lh /var/sftp/files/aim-sftp/wp-content/themes/
total 0
lrwxrwxrwx 1 root root 56 May  8  2024 twentytwentythree ->
/usr/share/wordpress/wp-content/themes/twentytwentythree

$ ls -lh /var/lib/wordpress/wp-content/themes/
total 0
lrwxrwxrwx 1 root root 56 May  8  2024 twentytwentythree ->
/usr/share/wordpress/wp-content/themes/twentytwentythree

$ ls -lh /usr/share/wordpress/wp-content/themes/
total 4.0K
drwxr-xr-x 7 root root 4.0K May 12 14:43 twentytwentythree
```

SFTP-yhteys Windowsista - WinSCP

Jos pääasiallisena työasemana on Windows-kone, kannattaa terminaalipohjaisten SFTP-ohjelmien sijaan käyttää graafisia ohjelmistoja. Yksi monipuolisimmista ohjelmista on WinSCP. Tämän saa ladattua osoitteesta: https://winscp.net/eng/download.php.

WinSCP on avoimen lähdekoodin ohjelmisto, joka tukee SFTP-yhteyden lisäksi SCP-protokollaa, hakemistojen synkronointia ja skriptausta automaatioiden toteuttamiseen. WinSCP:stä löytyy myös integraatio PuTTY:n ohjelmistopakettiin, joten avainpareja voidaan hallita jälleen Pageantista.

Kun WinSCP käynnistetään, aukeaa seuraava näkymä:

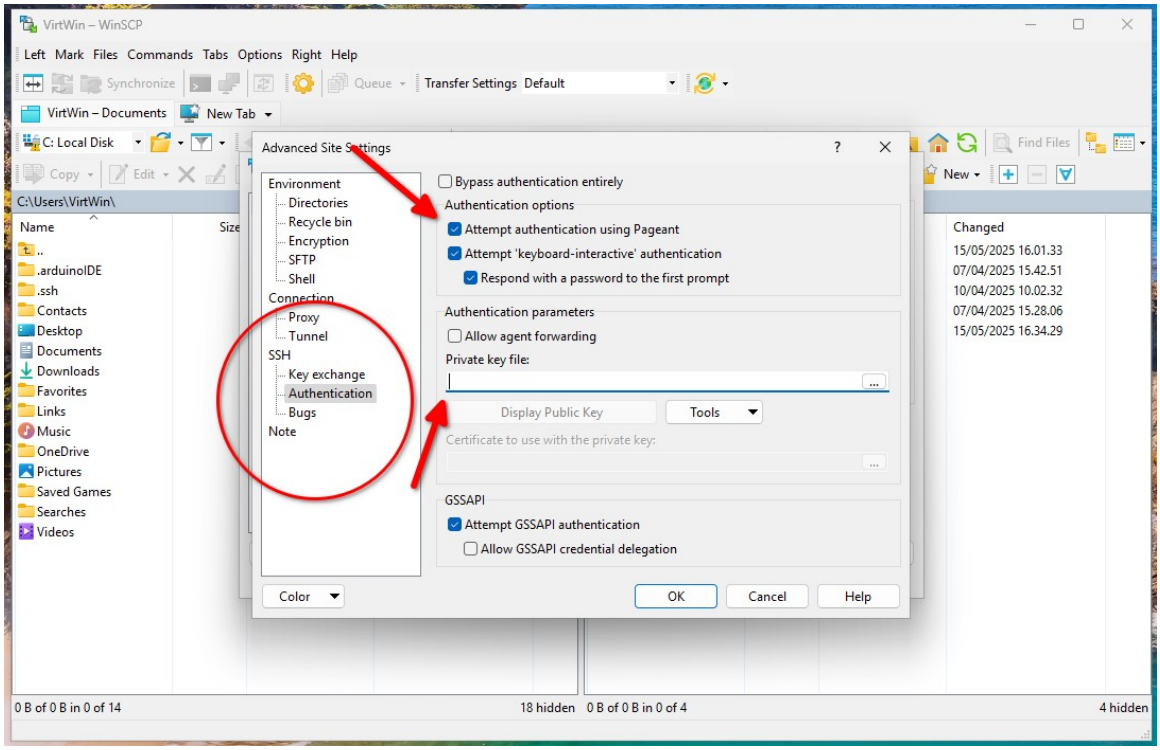


Kuva 12: WinSCP:n oletusnäkymä

Login-näkymässä on periaatteessa kaikki oleellinen. Jos palvelimella ei ole käytössä avainparitunnistautumista, voidaan tästä kirjautua suoraan ilman isompaa säätöä. Jos avainparitunnistautuminen on käytössä, tulee SFTP-käyttäjälle luoda uusi avainpari. Tämä onnistuu PuTTYgen:illä, jolla luotu avain voidaan tallentaa Pageantiin. Tarkemmat ohjeet tähän löytyy tämän dokumentaation OpenSSH -luvusta; avainparin julkinen pari tulee myös tallentaa palvelimelle.

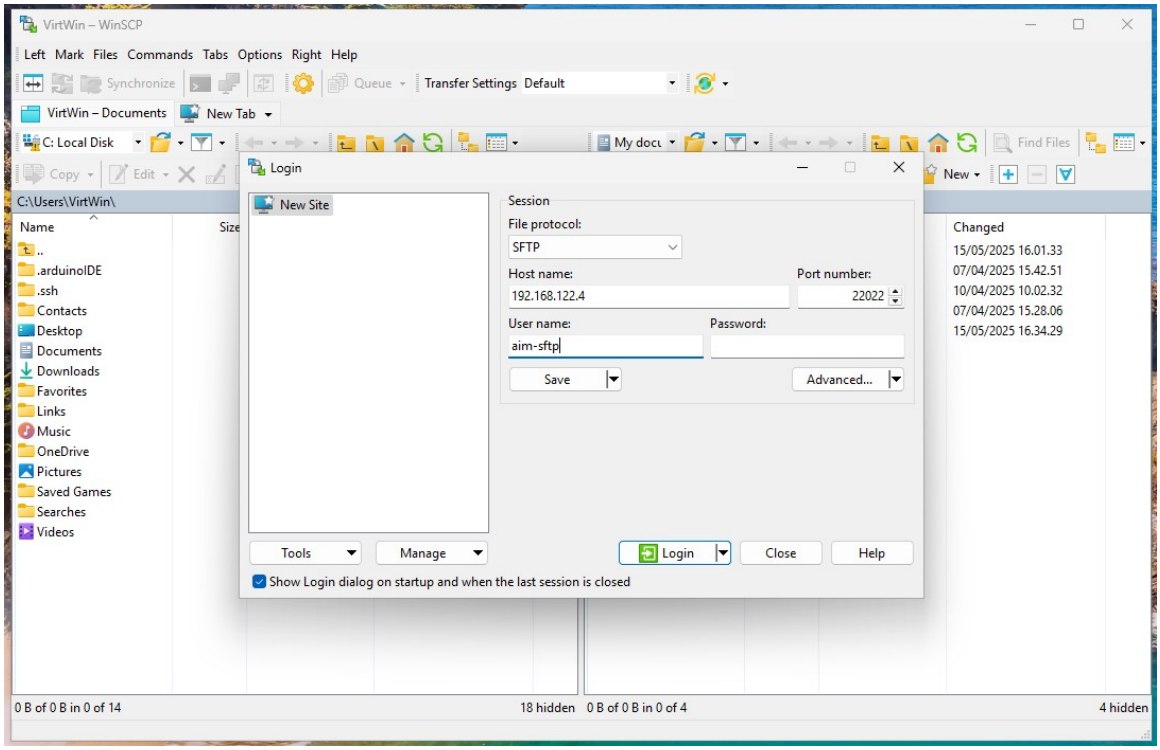
Euroopan unionin osarahoittama

Kun avainpari on luotu, klikkaamalla Login-näkymän ’Advanced...’ -painiketta, pääsemme ottamaan avaimen WinSCP:n käyttöön. Jos haluamme käyttää Pageantia avainten hallintaan, voidaan ’Attempt authentication using Pageant’ jättää päälle. Vaihtoehtoisesti avaintiedoston voi hakea suoraan (sertifikaattia ei tarvitse asettaa). Asetukset otetaan käyttöön ’OK’ -painikkeella.



Kuva 13: WinSCP:n autentikointiasetukset

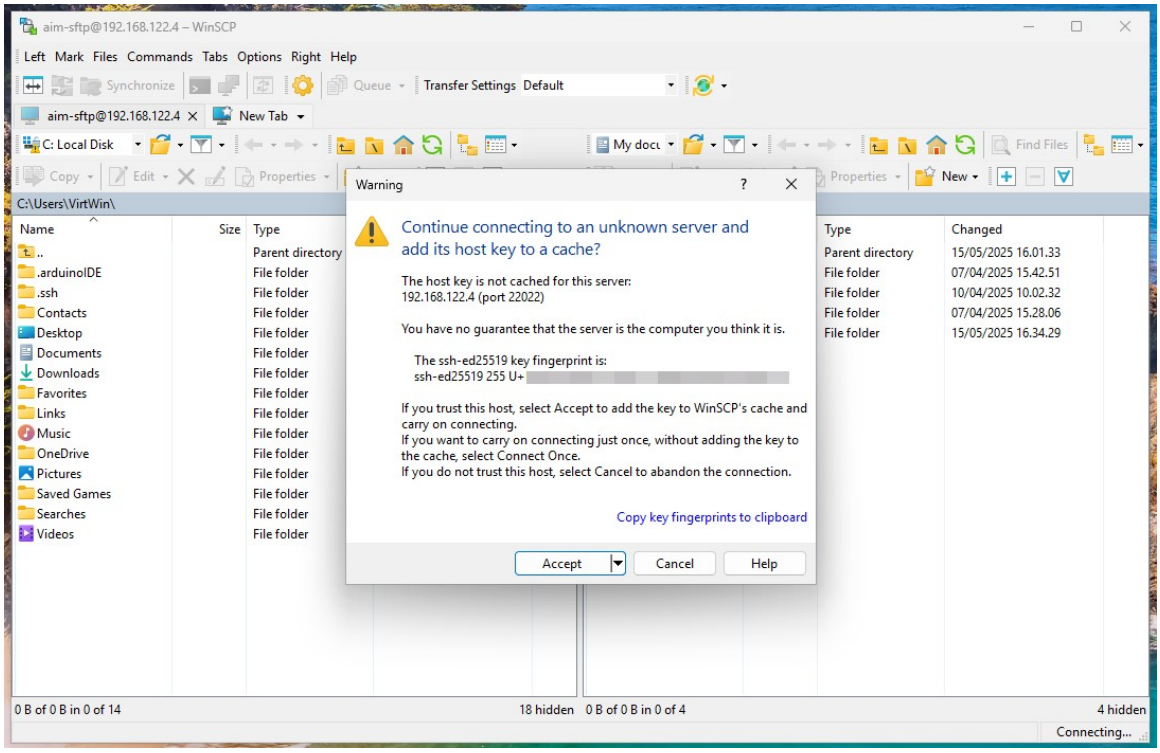
Kun avainparin asetukset ovat valmiina, voimme siirtyä kirjautumaan. Kirjautumisikkunassa syötetään tavalliseen tapaan palvelimen IP-osoite, SSH-portti ja SFTP-käyttäjän nimi. Avainpohjaista autentikointia käyttäessä salasanaa ei syötetä tähän näkymään lainkaan. ’Save’ -painikkeella tiedot voidaan tallentaa, jolloin ne ovat nopeasti saatavilla vasemman laidan palkissa. Painamalla ’Login’ -painiketta voimme aloittaa autentikoinnin.



Kuva 14: Palvelimen osoitetiedot ja käyttäjä syötetään WinSCP:hen

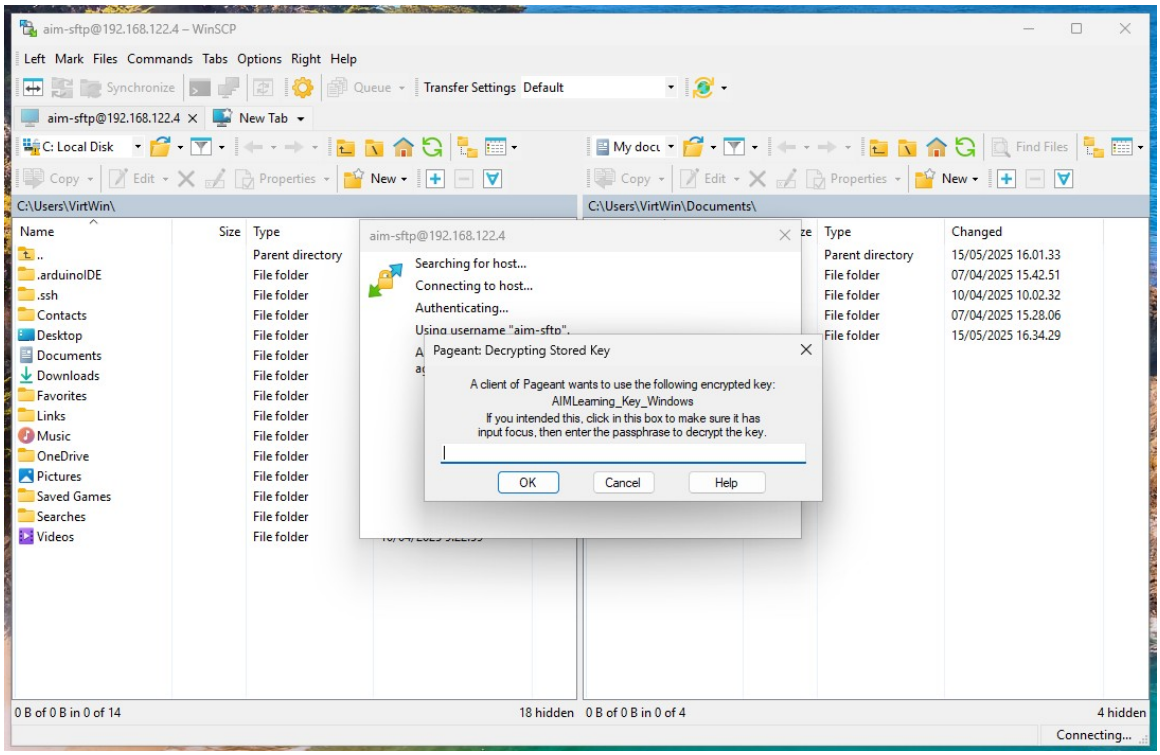
Euroopan unionin osarahoittama

Ensimmäistä kertaa palvelimelle autentikoidessa WinSCP pyytää lupaa tallentaa palvelimen avaimen tiedot välimuistiin. Tämä esitetään varoituksen muodossa, sillä potentiaalisesti tuntematon palvelin saattaa olla vaarallinen. Koska kyse on omasta palvelimestamme, voimme hyväksyä pyynnön ’`Accept`’ -painikkeella.

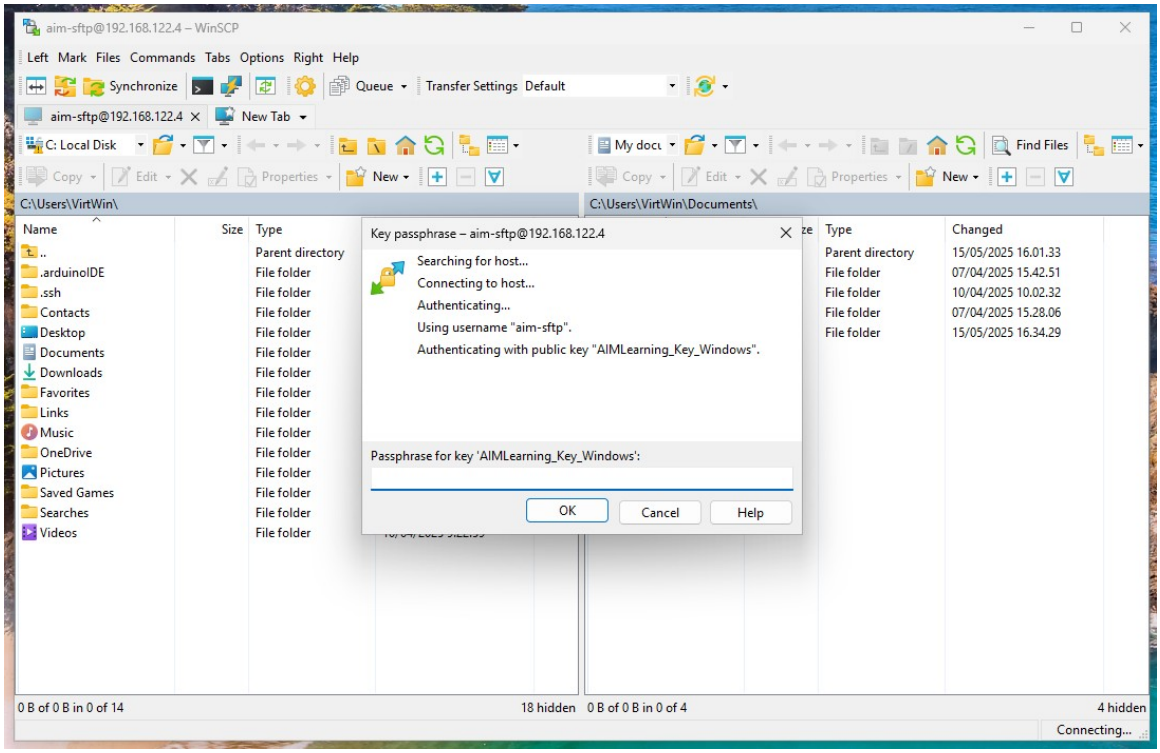


Kuva 15: WinSCP varoittaa sille ennestään tuntemattomasta palvelimesta

Riippuen valituista asetuksista, avautuu toinen seuraavista ikkunoista. Logiikka on molemmissa sama: yksityisen avaimen salasana syötetään ikkunaan, jotta avain voidaan lukea tunnistautumista varten.

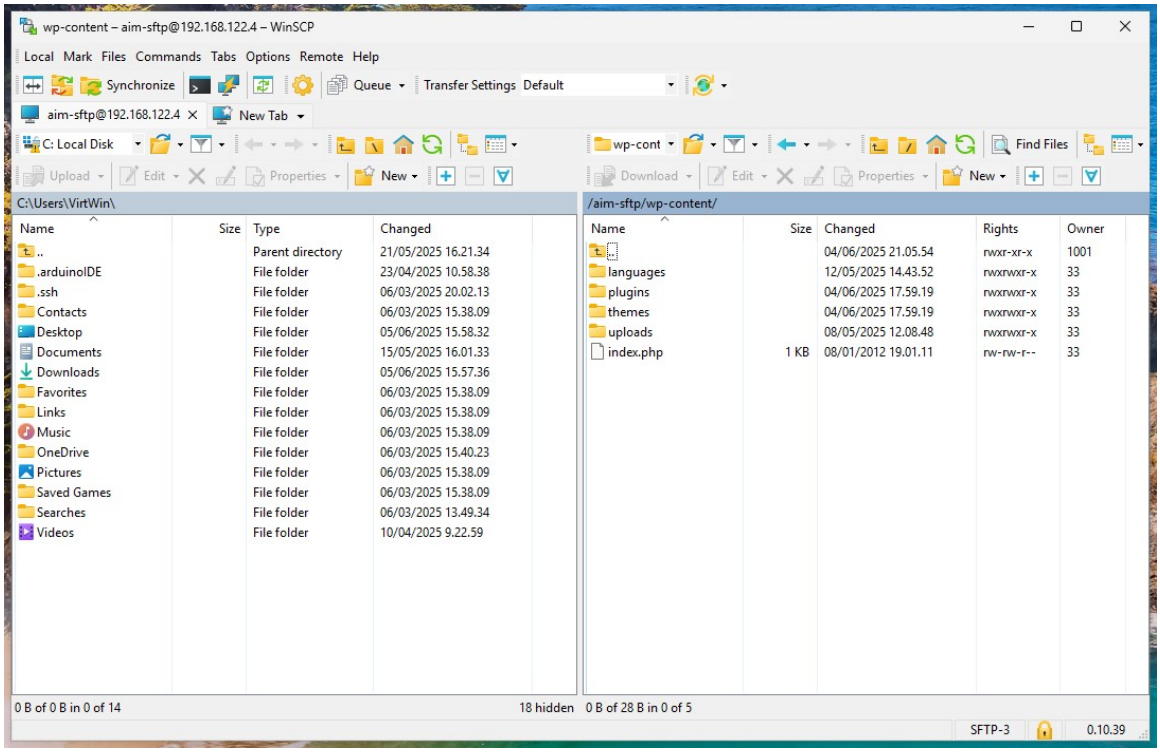


Kuva 16: Avaintunnistautuminen Pageantin avulla



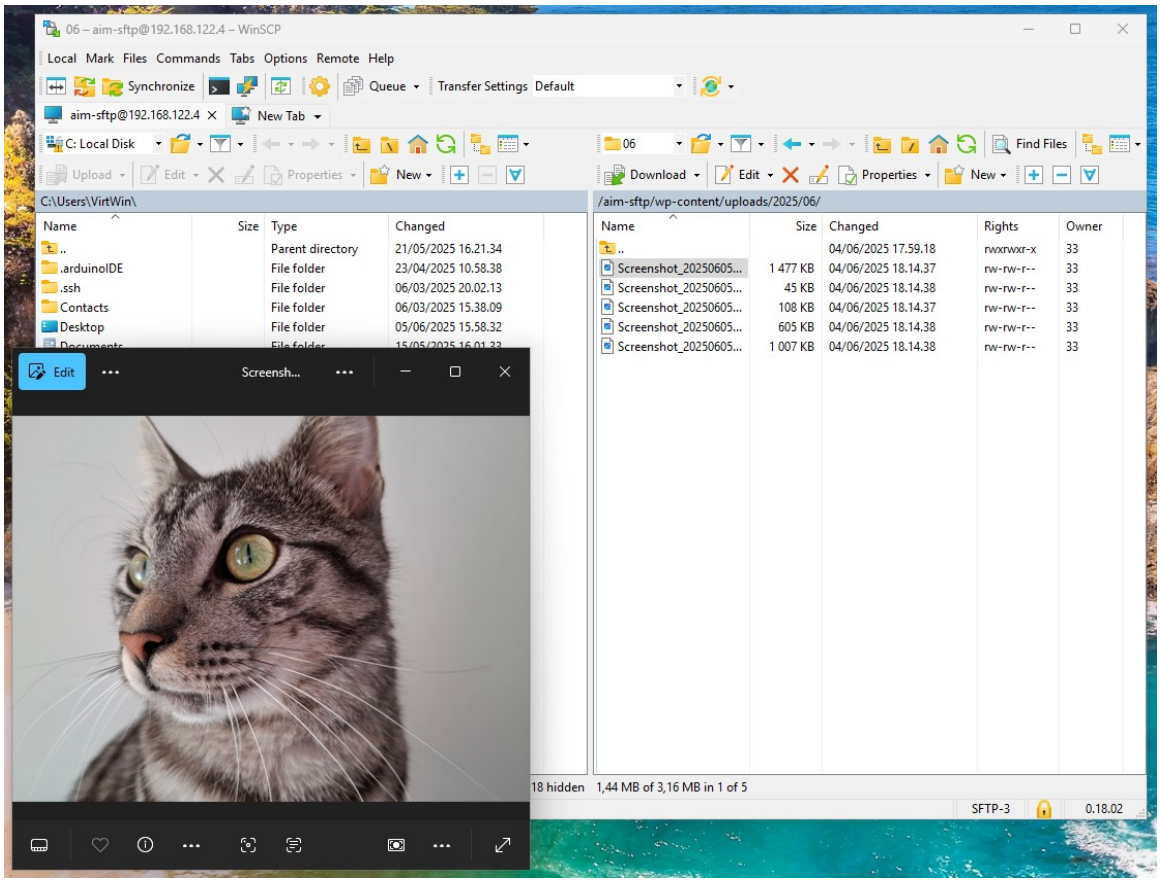
Kuva 17: Avaintunnistautuminen suoraan avaintiedoston avulla

Tunnistautuminen on nyt valmis ja SFTP-käyttäjän tiedostot ovat käytettävissä. Riippuen WinSCP:n asetuksista, näkymä saattaa olla erilainen. Käytössä on joko ainoastaan palvelimen tiedostonäkymä, joka toimii tavallisen Windows Explorerin tavoin. Vaihtoehtoisesti Commander-tyylisessä näkymässä on vasemmalla paikallisen koneen tiedostot ja oikealla palvelimen tiedostot. Molempia voi kuitenkin käyttää lähes tulkoon samoin kuin Windows Exploreria.



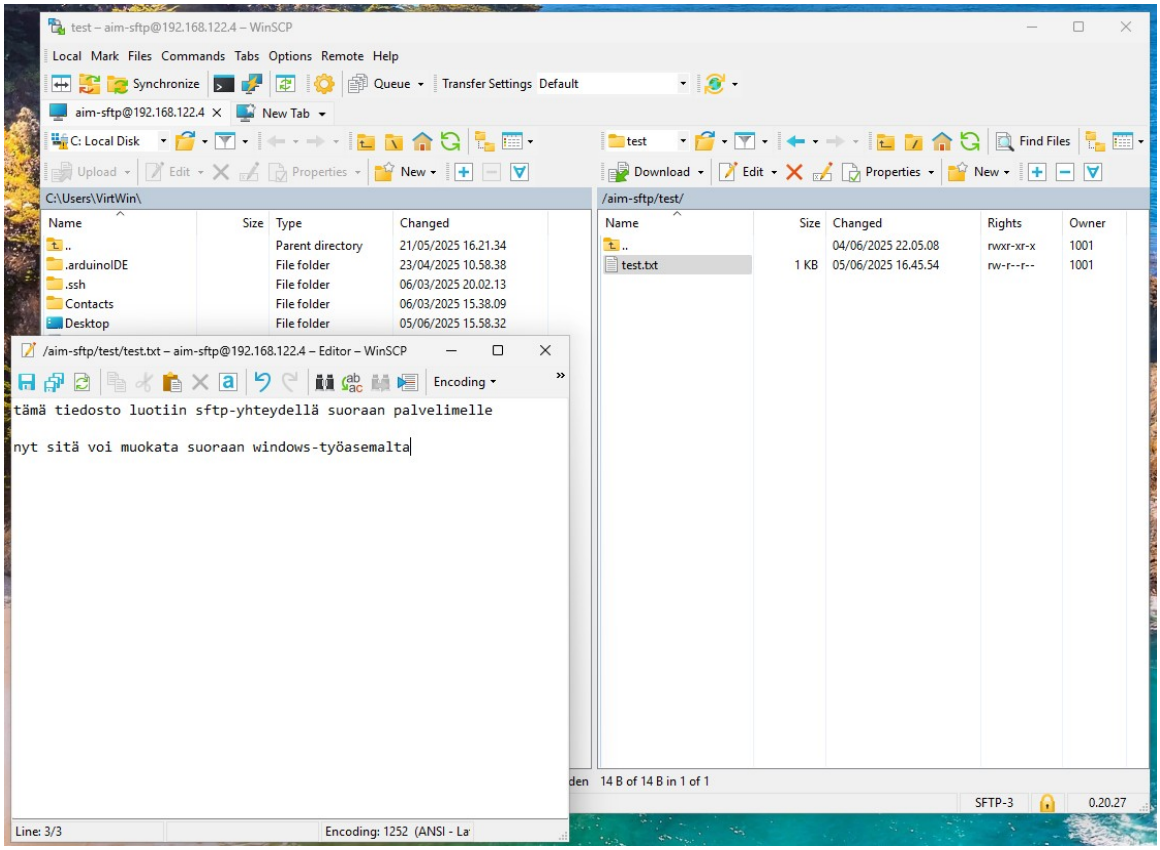
Kuva 18: WinSCP:n Commander-käyttöliittymä

Palvelinnäkymän puolelta voimme esimerkiksi ladata tarvittavia tiedostoja omaan työasemaan. Oikeuksista riippuen voimme myös tarkastella ja/tai luoda ja muokata tiedostoja suoraan palvelimella. Pääsemme tarvittaessa suoraan käsiksi vaikkapa verkkosivuston mediatiedostoihin.



Kuva 19: Verkkosivuston kuvatiedosto avattuna suoraan palvelimelta

Euroopan unionin osarahoittama



Kuva 20: Palvelimella sijaitsevaa tekstitiedostoa muokataan suoraan SFTP-yhteyden yli

Lähteet:

https://www.ssh.com/academy/ssh/sftp-ssh-file-transfer-protocol

https://www.ssh.com/academy/ssh/ftp#inherently-insecure

https://docs.rackspace.com/docs/set-up-sftp-users-in-linux-based-systems

https://winscp.net/eng/docs/introduction

https://www.baeldung.com/linux/openssh-internal-sftp-vs-sftp-server

https://serverfault.com/questions/448647/symbolic-link-and-filezilla-over-sftp

man mount

man fstab

Web-palvelimen SSL-tuki

Oletuksena käyttöönottamamme web-palvelimen liikenne toimii tavallisen HTTP-protokollan yli. Tämän johdosta kaikki verkkosivustoa koskeva liikenne on salaamatonta, jolloin jopa käyttäjien salasanat ovat paketeissa tavallisessa tekstimuodossa:

```
 1 POST /wp-login.php HTTP/1.1
 2 Host: 192.168.122.4
 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:128.0) Gecko/20100101
Firefox/128.0
 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 5 Accept-Language: en-US,en;q=0.5
 6 Accept-Encoding: gzip, deflate
 7 Referer: http://192.168.122.4/wp-login.php
 8 Content-Type: application/x-www-form-urlencoded
 9 Content-Length: 137
10 Origin: http://192.168.122.4
11 DNT: 1
12 Connection: keep-alive
13 Cookie: wordpress_test_cookie=WP%20Cookie%20check; wp_lang=en_US
14 Upgrade-Insecure-Requests: 1
15 Priority: u=0, i
16
17 log=webmaster%40localhost.asd&pwd=salasana123_kissa9001&wp-
submit=Log+In&redirect_to=http%3A%2F%2F192.168.122.4%2Fwp-
admin%2F&testcookie=1HTTP/1.1 302 Found
18 Date: Thu, 12 Jun 2025 09:56:22 GMT
19 Server: Apache/2.4.62 (Debian)
20 Expires: Wed, 11 Jan 1984 05:00:00 GMT
21 Cache-Control: no-cache, must-revalidate, max-age=0
22 Set-Cookie: wordpress_test_cookie=WP%20Cookie%20check; path=/
23 X-Frame-Options: SAMEORIGIN
...
```


Euroopan unionin osarahoittama

Ratkaisuna tälle on muuttaa web-palvelin käyttämään HTTPS-protokollaa, joka salaa yhteyden SSL-salausprotokollaa hyödyntäen. Koska liikenne on salattua, ei kukaan pysty selvittämään siitä kriittisiä tietoja, kuten käyttäjien salasanoja:

```
 1 ....p...l......'.[.
 2 >.^..A.I'c2.h@.K.".vg.. ..N#J.c..N%.`J.Z..-_=....i.....]."......+./.....,.0.
 3 .  ........./.5.............
 4 ............................h2.http/1.1.........".
 5 ..........3.k.i... Y
 6 ....w$O...J..#....3....Q
 7 ....1...A.4.F.."...*..Z.\..K.R)...=...+p..u......+i..!.....(K.6.l...
     [.^...+........
 8 ..............................@..
 9 ......... .X.P..`A.59..".p.......a.`l_$.gT...d....BjB.Z......C...d.........
(.a..1!Ok........00..S4......b.9..m ..._.....E.H..*
10 .r..v...-..'...h.!....4..Q..,..L6s.6I..j..lG<7.]|/..G0...P.~..L.*.!
zq.7.S.V..2..M.Y.x"..
11 ..t3.@.....G.t...]Qt.^...z.......z..m$..<I....j.......~..Y<....V........z...v
..\[.&..|.l...p.x..!.9....D...b.&f ..N#J.c..N%.`J.Z..-_=....i.....]......
+.....3.$... p..!..53.PPE.r.*k.~G"..../.9x^.|..........&p.5.03.M..^.....
...
```

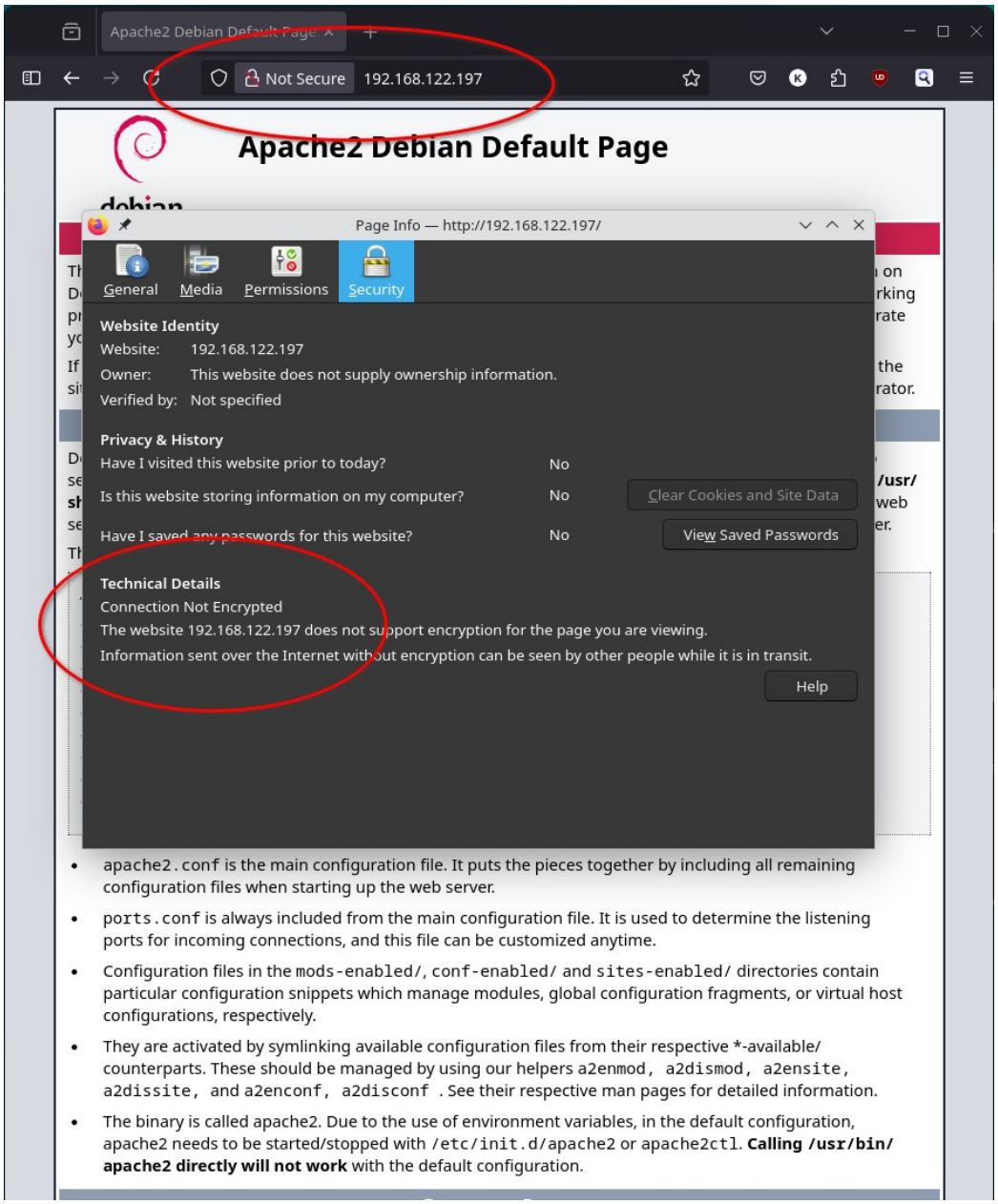
Liikenteen salaus on hyvin oleellinen osa minkä tahansa verkossa toimivan palvelun toimintaa, ja se on mahdollista ottaa käyttöön jo kehitysvaiheessa.

Kehityspalvelimelle tekemämme ratkaisu hyödyntää itse allekirjoitettua sertifikaattia. Tämä voidaan ottaa käyttöön vaikka sivustolla ei olisi rekisteröityä domain-nimeä. Varsin turvallisena sitä ei voida yleisesti pitää, mutta kehityksen aikaiseen tietojen turvaamiseen se sopii hyvin.

Lähtökohtaisesti tuotannossa olevilla sivustoilla ja palveluilla on sertifikaatteja hallinnoivien, luotettujen Certificate Authorities palveluiden myöntämät sertifikaatit. Selaimet pystyvät tällä tavalla varmistamaan että sivusto on varmasti turvallinen. Jos on tarve oikealle sertifikaatille, yksi varteenotettava, luotettava ja ilmainen vaihtoehto on Let’s Encrypt. Lisätietoja: https://letsencrypt.org/, https://certbot.eff.org/.

Self-signed sertifikaatti - Apache2

Apache2:ssa on suhteellisen helppoa ottaa käyttöön SSL-tuki verkkosivustoille. Tämä ei ole oletuksena päällä, joka huomataan kun siirrytään palvelimen verkkosivustolle:



Kuva 21: Apache2:n oletussivu suojaamattoman yhteyden ylitse

Itse allekirjoitetun sertifikaatin, eli Self-Signed sertifikaatin, käyttöönotto Apache2:ssa on yksi keino suojata sivuston web-yhteys kehityksen aikana. Tämä vaatii ainoastaan `openssl` -paketin, jolla sertifikaatti voidaan luoda. Asennetaan kyseinen paketti ja varmistetaan samalla että Apache2 on asennettuna:

```
$ apt update && apt upgrade -y
$ apt install apache2 openssl
```


Euroopan unionin osarahoittama

Aivan ensimmäisenä kannattaa tuplavarmistaa, että sivusto toimii tavallisen HTTP-protokollan yli. Tämä tapahtuu siirtymällä palvelimen selaimella `http://localhost/` -osoitteeseen, tai vaihtoehtoisesti tulostamalla sivun sisältö komentoriville:

```
$ curl http://localhost/
```

Kun verkkosivustot toimivat normaalisti, voidaan siirtyä luomaan sertifikaatteja. Ensimmäisenä tulee luoda hakemistot, joihin sertifikaatit tallennetaan. Periaatteessa tämän sijainnilla ei ole väliä, mutta hyvänä käytäntönä on luoda nämä SSL:n hakemistoon. Lisäksi tulee huolehtia, että hakemistolla ja tämän sisällöllä on `root:root` -omistusoikeudet (nämä hakemistot tulee luoda `sudo` -oikeuksin).

```
$ mkdir -p /etc/ssl/localcerts/certs
$ mkdir /etc/ssl/localcerts/private
```

Uudet sertifikaatit saadaan luotua näille tarkoitettuihin hakemistoihin `openssl` -ohjelmistolla. Komennon suhteen kannattaa olla tarkkana, jotta kaikki parametrit ja tiedostosijainnit ovat varmasti oikein.

```
$ openssl req -newkey rsa:2048 -x509 -days 365 -noenc -out
/etc/ssl/localcerts/certs/apache-selfsigned.crt -keyout
/etc/ssl/localcerts/private/apache-selfsigned.key
```

Komennon selitykset:

`-x509`: tämä flag määrittää, että openssl luo sertifikaatin sertifikaattipyynnön sijaan

`-days`: kun -x509 komennossa, tällä määritetään sertifikaatin voimassaolon pituus. Oletuksena 30 päivää

`-noenc`: jos yksityinen avain luodaan, sitä ei salata. Käytännössä tässä tapauksessa tämä helpottaa elämää, sillä apache2 ei saisi avattua salattua avainta automaattisesti palvelimen käynnistyessä

`-newkey`: generoi uuden yksityisen avaimen, jos -key ei ole annettu samassa komennossa. Tässä tapauksessa RSA:2048 spesifioituna. Ohjelma kysyy käyttäjältä tarvittavia tietoja sertifikaattiin, ja generoi samalla uuden yksityisen avaimen tiedoilla, jotka annetaan konfiguraatiotiedostossa tai -newkey tai -pkeyopt flageilla TAI defaulttaa 2048-bit RSA-avaimeen

`-out`: kirjoitettavan tiedoston tiedostonimi

`-keyout`: luotavan yksityisen avaimen tiedoston nimi

Kun komento ajetaan, kysytään sertifikaattia varten tietoja. Nämä voidaan täyttää niin, että oma avain tunnistetaan jatkossa helposti:

```
$ openssl req -newkey rsa:2048 -x509 -days 365 -noenc -out
/etc/ssl/localcerts/certs/apache-selfsigned.crt -keyout
/etc/ssl/localcerts/private/apache-selfsigned.key
...
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:Keski-Pohjanmaa
Locality Name (eg, city) []:Kokkola
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Centria AIMlearning
Demo
Organizational Unit Name (eg, section) []:AIMlearning
Common Name (e.g. server FQDN or YOUR name) []:Kalle Vesantera
Email Address []:kalle.vesantera@example.com
```

Lopuksi muutetaan luotujen tiedostojen oikeudet niin, että ainoastaan omistajalla (root) on luku- ja kirjoitusoikeudet:

```
$ chmod 600 /etc/ssl/localcerts/private/apache-selfsigned.key
$ chmod 600 /etc/ssl/localcerts/certs/apache-selfsigned.crt
```

Luotu sertifikaatti itsessään näyttää tekstimuodossa seuraavalta:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            19:50:5f:76:01:44:42:1a:54:86:52:34:da:17:db:09:80:94:e2:c8
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FI, ST = Keski-Pohjanmaa, L = Kokkola, O = Centria
AIMlearning Demo, OU = AIMlearning, CN = Kalle Vesantera, emailAddress =
kalle.vesantera@example.com
        Validity
            Not Before: May 13 14:20:38 2025 GMT
            Not After : May 13 14:20:38 2026 GMT
```

```
        Subject: C = FI, ST = Keski-Pohjanmaa, L = Kokkola, O = Centria
AIMlearning Demo, OU = AIMlearning, CN = Kalle Vesantera, emailAddress =
kalle.vesantera@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c7:c3:79:b1:fe:fe:f6:dc:cc:4b:43:da:6a:05:
                    21:21:75:07:56:e6:5d:c5:eb:d3:75:e1:ef:5d:51:
                    67:27:79:1d:95:0b:4f:a7:3e:d7:a8:1b:1d:09:89:
                    ...
                    a9:c1:36:92:19:8f:46:07:cf:10:43:bc:0c:86:96:
                    9c:6a:b2:a3:7a:dc:10:c3:3c:43:5d:e5:10:5d:75:
                    e0:e8:96:44:47:a5:09:29:46:9f:79:cd:f3:c6:37:
                    5a:a9
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                BA:94:D4:2D:3F:01:9E:E0:C2:5C:15:1E:A0:CC:98:A5:F4:CB:7F:03
            X509v3 Authority Key Identifier:
                BA:94:D4:2D:3F:01:9E:E0:C2:5C:15:1E:A0:CC:98:A5:F4:CB:7F:03
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        22:f0:88:b2:a6:c9:90:88:2b:01:9e:c3:e1:ef:7a:64:4d:ab:
        ...
        6e:43:8b:5e:06:65:ff:ea:19:8a:bd:a4:5b:af:dc:20:71:08:
        48:2b:33:18:2a:08:af:cd:8b:6d:fc:70:2c:b8:de:01:83:ae:
        6f:19:11:a6
```

Kun sertifikaatti on luotu, käynnistetään Apache2:n SSL-moduuli. Tämä moduuli on Apache2:ssa sisäänrakennettuna ja sen avulla hoidetaan TLS/SSL-protokollan avulla tapahtuva yhteyden suojaus.

```
$ a2enmod ssl
$ systemctl restart apache2
```

Apache2:n sivustot eivät siirry automaattisesti käyttämään HTTPS-protokollaa. Tätä varten sivustolle tulee luoda uusi Virtual Host -konfiguraatiotiedosto, jossa otetaan käyttöön oikea verkkoportti ja sertifikaattitiedostot. Yksinkertaisimmillaan tämä voidaan toteuttaa kopioimalla esimerkkikonfiguraatio, jota sitten muokataan.

```
$ cd /etc/apache2/sites-available
$ cp default-ssl.conf ssl-demo.conf
```

Uutta tiedostoa voidaan muokata tekstieditorilla, kuten nanolla. Tiedostossa tulee varmistaa, että ensimmäisellä rivillä on merkitty verkkoportiksi `:443`. Lisäksi tiedostoon määritellään luotujen Self-Signed sertifikaattien sijainti.

```
$ nano ssl-demo.conf
1 <VirtualHost *:443>
2     ServerAdmin webmaster@localhost
...
31 SSLCertificateFile      /etc/ssl/localcerts/certs/apache-selfsigned.crt
32 SSLCertificateKeyFile   /etc/ssl/localcerts/private/apache-selfsigned.key
...
```

Virtual Host -tiedostoihin voidaan määritellä mielivaltaisesti melkein mikä tahansa portti. Apache2 ei osaa kuitenkaan kuunnella porttia, ilman että se määritellään `/etc/apache2/ports.conf` -tiedostoon. Tavallisen HTTPS-yhteyden portti on `:443`. Tämän tulisi löytyä kyseisestä tiedostosta:

```
1 Listen 80
2 <IfModule ssl_module>
3     Listen 443
4 </IfModule>
5 <IfModule mod_gnutls.c>
6     Listen 443
7 </IfModule>
```

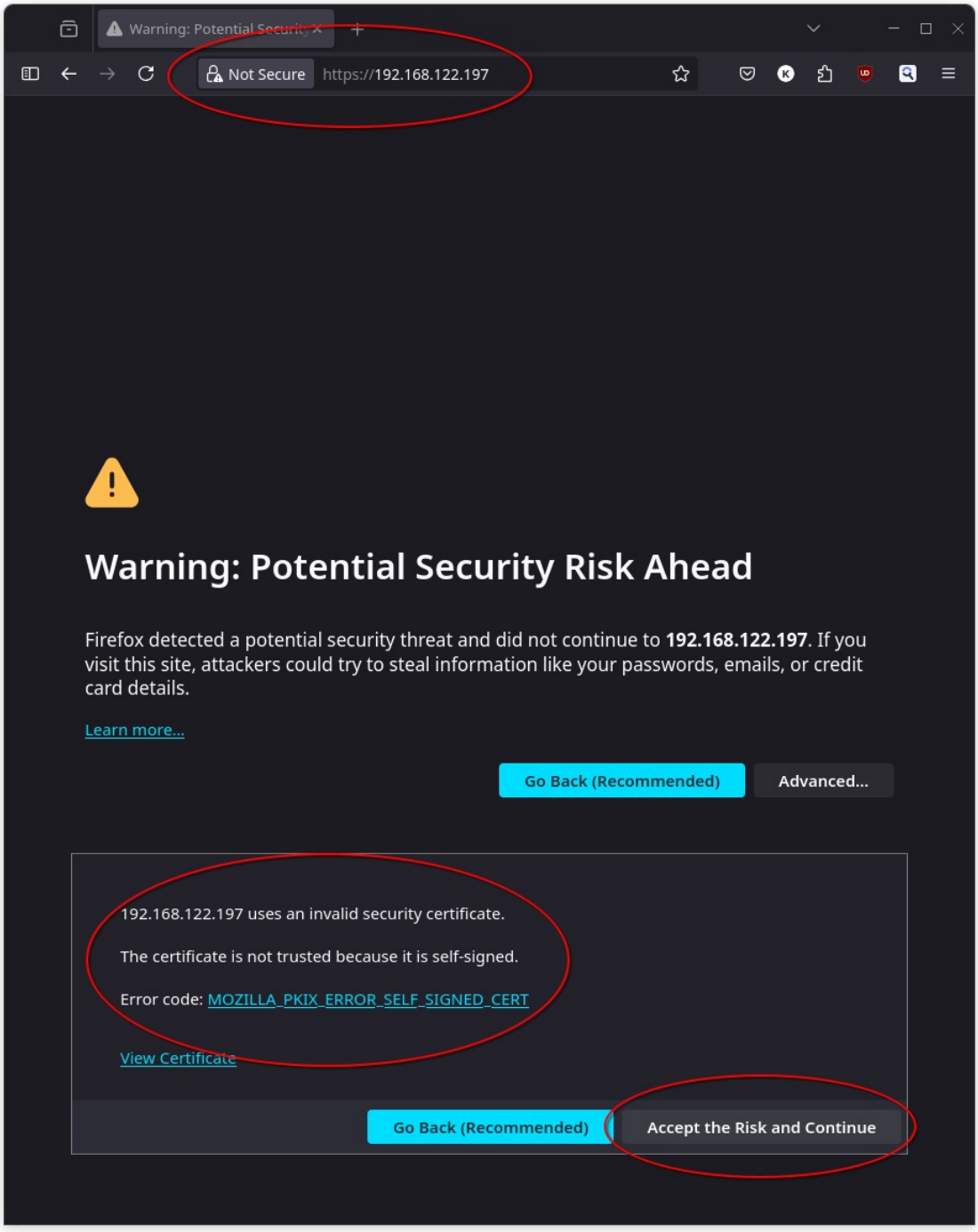
Jos näitä konfiguraatioita ei ole, voi ne lisätä tiedostoon tekstieditorilla. Vastaavasti, jos kyseiset rivit ovat kommentoituna, tulee ne ottaa käyttöön.

Kun Virtual Host ja portti on konfiguroituna, voidaan HTTPS-yhteyttä käyttävä tiedosto ottaa käyttöön:

```
$ a2ensite ssl-demo
$ systemctl reload apache2
```

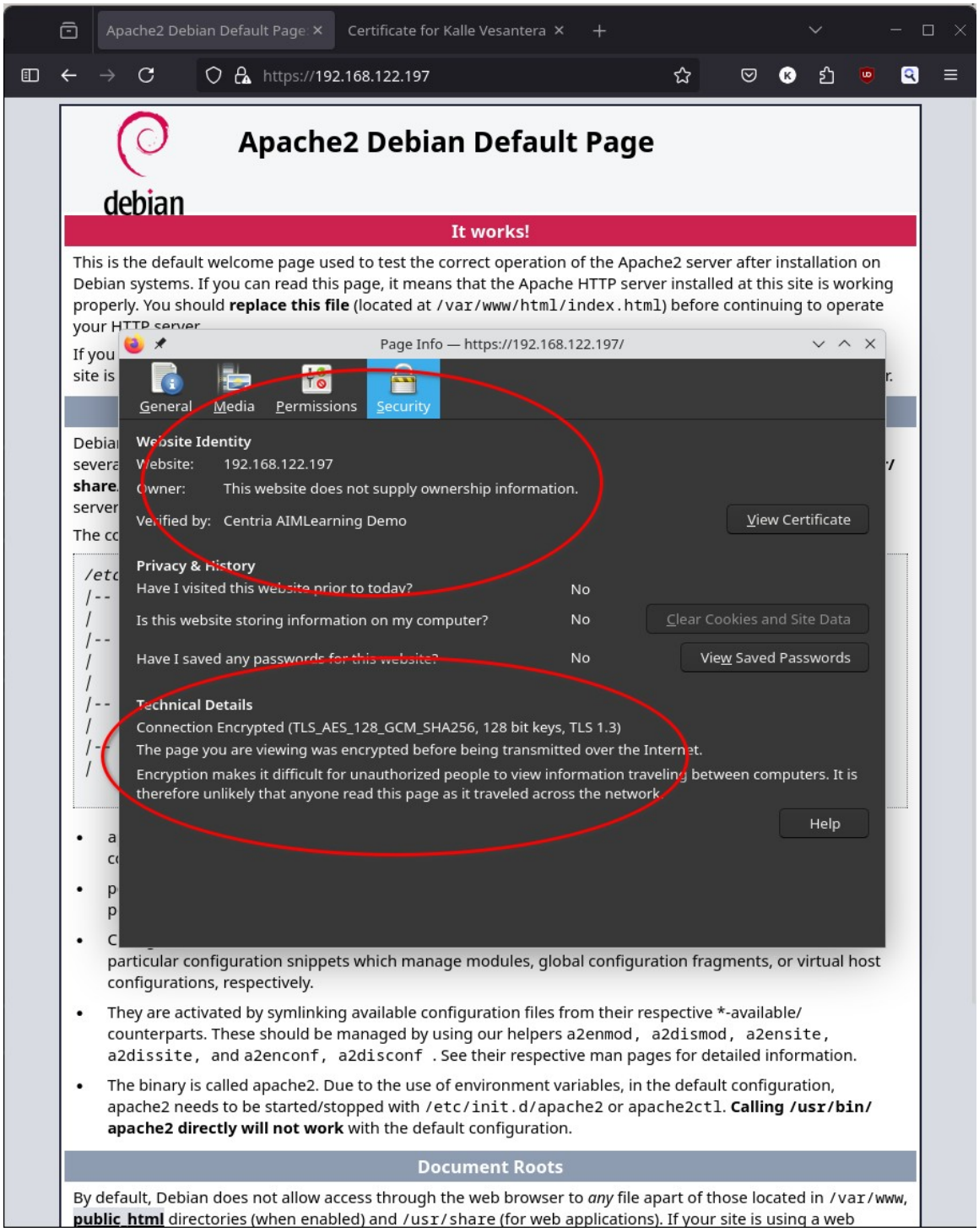

Euroopan unionin osarahoittama

Voimme varmistaa HTTPS-yhteyden toiminnan siirtymällä selaimella palvelimen osoitteeseen. Tällä kertaa protokollaksi asetetaan **`https://`**. Selain todennäköisesti valittaa vaarallisesta sivustosta. Tämä johtuu siitä, että sertifikaatti on itse allekirjoitettu. Varoituksen voi ohittaa turvallisesti ohittaa.



Kuva 22: Firefox varoittaa itse allekirjoitetusta sertifikaatista

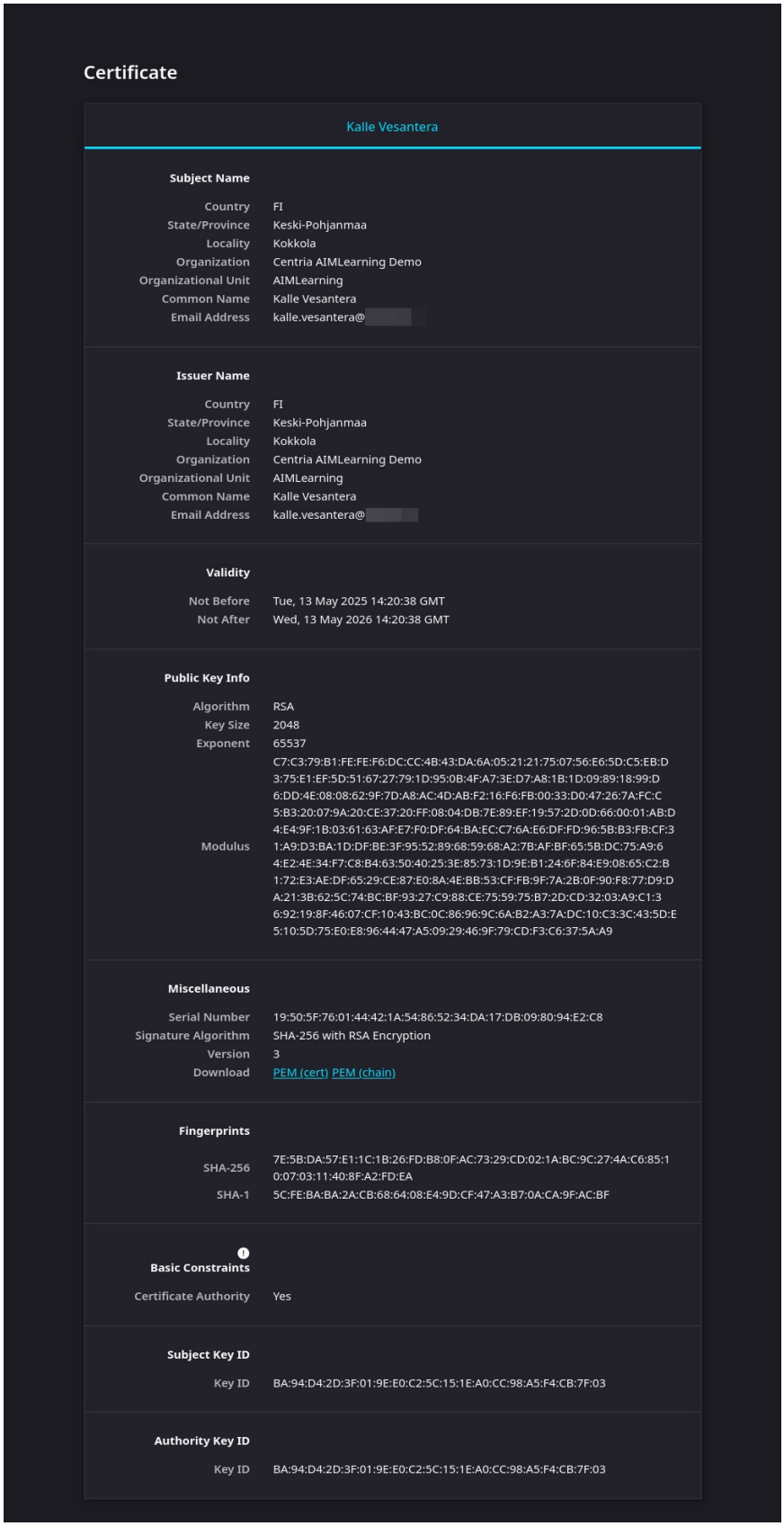
Sivusto toimii normaalisti; tiedoista näemme sen käyttävän itse allekirjoitettua sertifikaattia:



Kuva 23: Firefoxin Page Info kertoo sertifikaatin varmistajan

Euroopan unionin osarahoittama

Voimme myös selaimen kautta tarkistaa sertifikaatin tiedot:



Kuva 24: Firefoxin näkymä sertifikaatin kaikista tiedoista

Wordpress -sivuston SSL-tuki

Wordpress -sivustolle saadaan määriteltyä HTTPS-tuki melkein samalla tavalla kuin tavalliselle verkkosivustolle. Tämän käyttöönotto vaatii kuitenkin muutamia lisäkonfiguraatioita ja muutoksia Wordpressiin ja tämän tiedostoihin.

Apache2:een tulee luoda uusi Virtual Host -tiedosto, jonka päällä HTTPS-protokollaa hyödyntävä Wordpress toimii. Helpoiten tämän saa aikaan kopioimalla vanha Wordpress-konfiguraatio uuteen tiedostoon, jota sitten muokataan tarpeisiin sopivaksi.

```
$ cat /etc/apache2/sites-available/wp.conf | tee
/etc/apache2/sites-available/wp-ssl.conf
```

Uuteen tiedostoon tulee tehdä ensisijassa samat muutokset, kuin tavalliselle Apache2-sivustolle. Ensimmäisellä rivillä verkkoportti vaihdetaan HTTPS-oletusporttiin `:443`. Koska siirrymme nyt käyttämään suojattua verkkoyhteyttä, voidaan verkkosivustolle yhdistäminen sallia myös palvelimen ulkopuolelta. Tätä varten otamme käyttöön `ServerAlias` -konfiguraation, joka yhdistää IP- ja localhost -osoitteet loogisesti yhteen.

Myös sertifikaatin tiedostosijainnit tulee määritellä. Wordpress vaatii myös erillisiä konfiguraatioita SSL-tuen toteuttamiseksi. Nämä muutokset ovat alla **korostettuna**.

```
1  <VirtualHost *:443>
2        ServerName localhost
3        ServerAlias 192.168.122.4
4
5        ServerAdmin webmaster@localhost
6        DocumentRoot /usr/share/wordpress
7
8     #   SSL Engine Switch:
9     #   Enable/Disable SSL for this virtual host.
10    SSLEngine on
11
12    # Self-signed certificate files:
13    SSLCertificateFile      /etc/ssl/localcerts/certs/apache-selfsigned.crt
14    SSLCertificateKeyFile   /etc/ssl/localcerts/private/apache-selfsigned.key
15
16    ...
17
18    #SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
19    <FilesMatch "\.(?:cgi|shtml|phtml|php)$">
20          SSLOptions +StdEnvVars
21    </FilesMatch>
22    <Directory /usr/lib/cgi-bin>
23          SSLOptions +StdEnvVars
```


Euroopan unionin osarahoittama

```
    </Directory>

        Alias /wp-content /var/lib/wordpress/wp-content
        <Directory /usr/share/wordpress>
            Options FollowSymLinks
            AllowOverride Limit Options FileInfo
            DirectoryIndex index.php
            Require all granted
        </Directory>
        <Directory /var/lib/wordpress/wp-content>
            Options FollowSymLinks
            Require all granted
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Virtual Host -konfiguraatioiden lisäksi tulee varmistaa, että HTTPS-portti `:443` on Apache2:n käytössä. Tämä voidaan tarkistaa `/etc/apache2/ports.conf` -tiedostosta.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

# Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

Kun konfiguraatiotiedostot ovat valmiina, voidaan HTTPS-yhteyttä käyttävä sivusto ottaa käyttöön. Koska emme halua käyttää suojaamatonta sivustoa lainkaan, tulee tämä poistaa käytöstä.

```
$ a2ensite wp-ssl
$ a2dissite wp
$ systemctl reload apache2
```

Kun Wordpress otettiin ensimmäistä kertaa käyttöön palvelimella, luotiin tietokantayhteyttä varten `/etc/wordpress/config-localhost.php` -tiedosto. Wordpress asetetaan nyt toimimaan myös IP-osoitteen perusteella, ja tämä vaatii toimiakseen oman osoitepohjaisen konfiguraatiotiedoston. Yksinkertaisesti olemassa oleva tiedosto voidaan kopioida, muuttaen ainoastaan tiedoston nimi vastaamaan palvelimen osoitetta:

```
$ cp /etc/wordpress/config-localhost.php /etc/wordpress/config-192.168.122.4.php
```

Wordpress vaatii vielä erikseen tärkeän tietoturvakonfiguraation. Tämä pakottaa Wordpressin käyttämään suojattua yhteyttä, kun sivustolla suoritetaan autentikointitapahtumia. Ilman tätä konfiguraatiota ei voida olla varmoja, että turvallinen yhteys on toiminnassa. Lisäys tulee tehdä `/usr/share/wordpress/wp-config.php` -tiedostoon.

Varmuuden vuoksi tiedostosta on hyvä ottaa varmuuskopio, jos jotain sattuu menemään rikki:

```
$ cp wp-config.php wp-config.php.backup
```

`wp-config.php` -tiedoston loppupuolelle lisätään seuraava rivi:

```
define( 'FORCE_SSL_ADMIN', true );
```

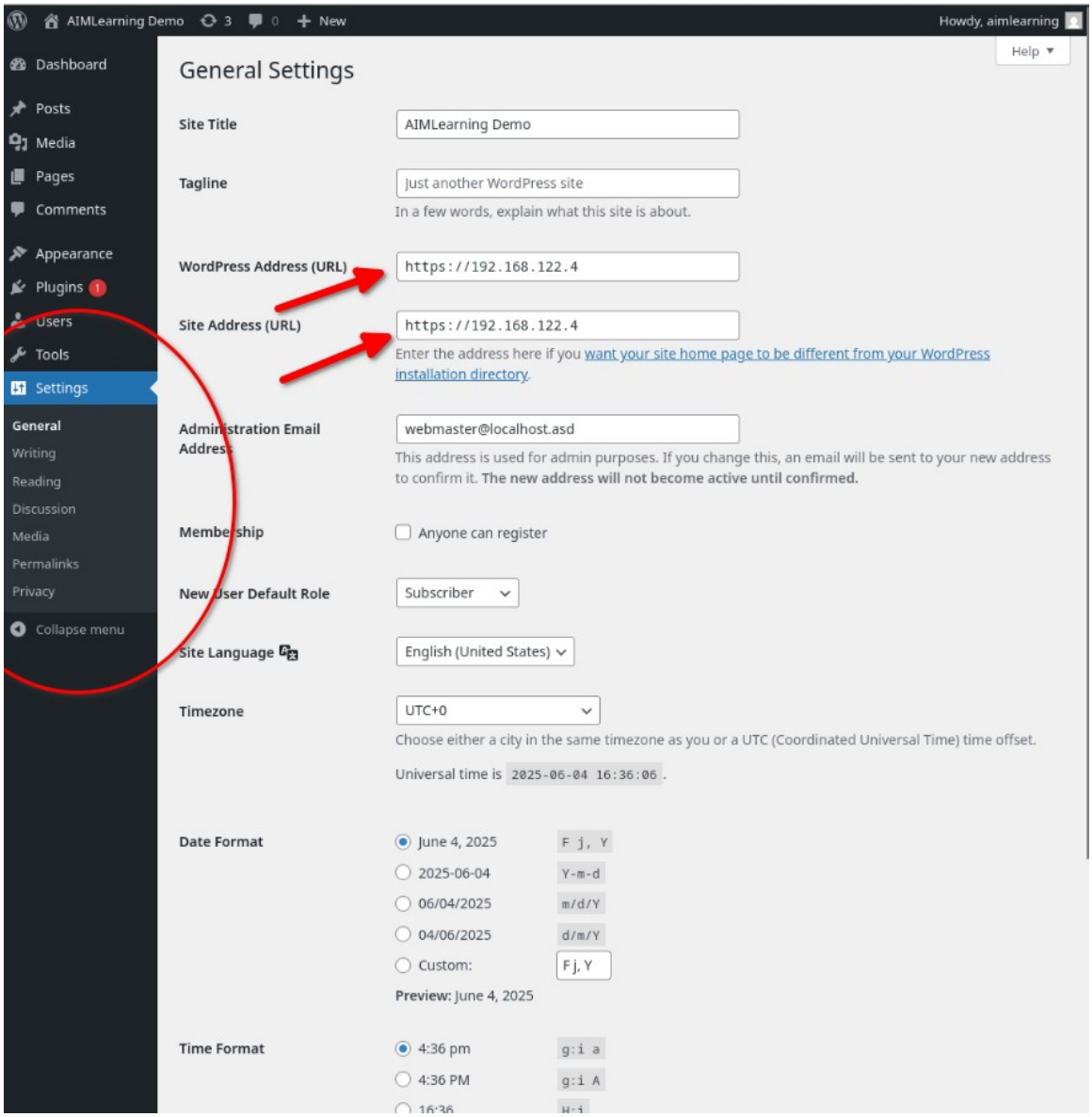
Tiedoston loppu näyttää siis seuraavalta:

```
...
70
71 define( 'FORCE_SSL_ADMIN', true );
72
73 // Make sure we're not using the short ("wp") tag
74 define('WP_FAIL2BAN_SYSLOG_SHORT_TAG', false);
75
76 // Don't include the HTTP host in the tag
77 define('WP_FAIL2BAN_SYSLOG_TAG_HOST', false);
78
...
```

Tässä vaiheessa on hyvä ladata Apache2 -palvelu uudelleen, jotta kaikki muutokset ovat varmasti käytössä.

```
$ systemctl apache2 reload
```

Web-palvelimen osalta kaiken pitäisi nyt toimia. Wordpressin Dashboardista tulee kuitenkin muuttaa osoitteita koskevat asetukset. Osoitteiden **`http://`** -protokolla muutetaan **`https://`** -protokollaksi. Myös osoitteena toimiva `localhost` muutetaan vastaamaan palvelimen IP-osoitetta. Ilman näitä muutoksia sivuston sisäinen linkitys ei toimi oikein. Varmimmin sivuston muutokset pääsee tekemään palvelimen selaimelta, siirtymällä sivustolle **`https://`** -protokollalla.



Kuva 25: Wordpress Dashboardin osoiteasetukset

Automaattinen ohjaus suojatulle sivustolle

Jos haluamme lisätä sivuston käyttäjäystävällisyyttä, voimme asettaa tavallisen HTTP-yhteyden ohjaamaan suoraan suojatun HTTPS-yhteyden päässä toimivaan sivustoon. Tämän saa luotua uudella Virtual Hostilla, joka hoitaa ohjaamisen automaattisesti.

Helpoimmin tämä tapahtuu tekemällä kopio oletus-konfiguraatiotiedostosta:

```
$ cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/redirect-to-ssl.conf
```

Automaattinen uudelleenohjaus saadaan aikaan lisäämällä uuteen tiedostoon vastaavanlainen rivi:

```
...
14         RedirectMatch 301 (.*) https://192.168.122.4
...
```

Tämän lisäksi Apache2 tulee asettaa kuuntelemaan HTTP-protokollan oletusporttia `:80`. Muutos tehdään `/etc/apache2/ports.conf` -tiedostoon, jossa portti sallitaan:

```
1 # If you just change the port or add more ports here, you will likely also
2 # have to change the VirtualHost statement in
3 # /etc/apache2/sites-enabled/000-default.conf
4
5 Listen 80
...
```

Nyt luotu uudelleenohjaussivusto voidaan ottaa Apache2:n käyttöön.

```
$ a2ensite redirect-to-ssl
$ systemctl restart apache2
```

Uudelleenohjausta voidaan kokeilla yhdistämällä selaimella sivustolle käyttäen tavallista `http://` -protokollaa. Selaimen tulisi tällöin ohjautua välittömästi suojatulle sivustolle.

Lähteet:

https://wiki.debian.org/Self-Signed_Certificate

https://linux.die.net/man/1/openssl
(man openssl)

https://linux.die.net/man/1/req
(man openssl-req)

https://bayton.org/blog/2016/01/switching-to-https-on-wordpress/

https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-20-04

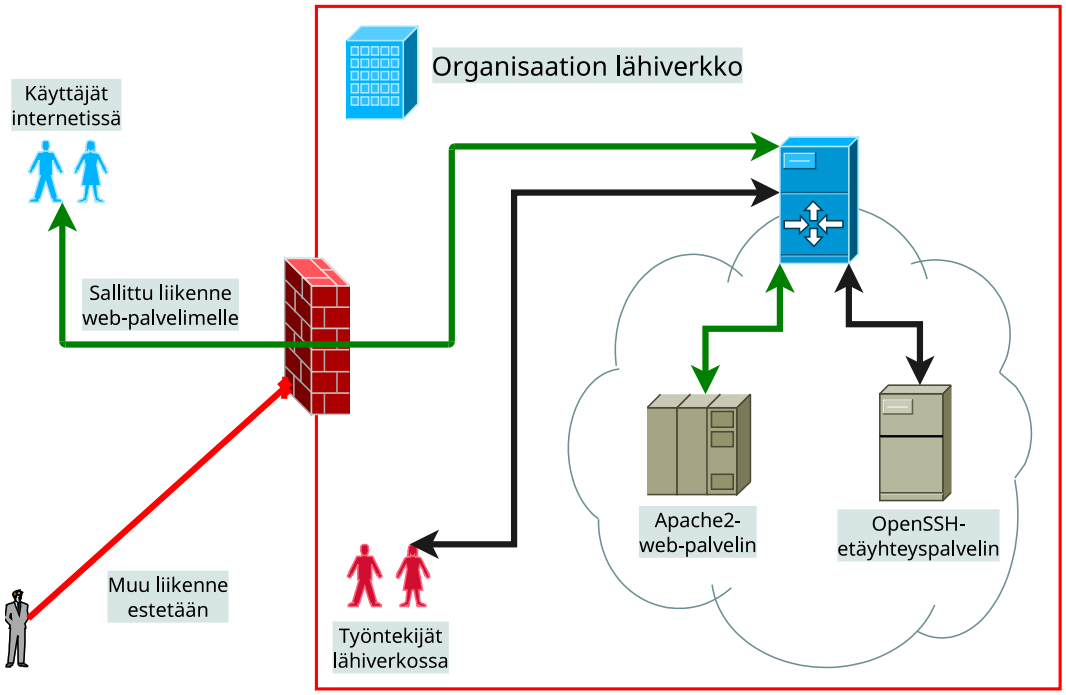
https://developer.wordpress.org/advanced-administration/security/https/

Euroopan unionin osarahoittama

Palomuuri

Palomuuri on yksi tärkeimmistä tietokoneiden ja/tai -järjestelmien osista. Palomuuriratkaisuja on käyttökohteiden mukaan erilaisia. Työasemissa, kuten kannettavissa tietokoneissa, on yleensä ohjelmistopohjainen palomuuriratkaisu. Tietoverkkojen ulkoreunoilla on fyysisiä palomuurilaitteita, jotka pystyvät käsittelemään reaaliajassa suuren määrän liikennettä.

Palomuurin tärkein tehtävä on toimia ikään kuin seinänä suojatun alueen ja ulkomaailman välillä. Palomuuriin voidaan asettaa sääntöjä, joiden mukaan tietynlainen liikenne päästetään läpi, mutta kaikki muu estetään.



Kuvaaja 2: Palomuurin toiminta lähiverkon ulkoreunalla

Esimerkiksi web-palvelimella säännöt voivat sallia `:80`- ja `:443`-porttien kautta sisään tulevan liikenteen, kaiken muun pysyessä estettynä. Vastaavasti palomuuriin voidaan tehdä sääntö, jossa SSH-yhteydet sallitaan ainoastaan sisäverkon IP-osoiteavaruuksista. Tällöin palvelimen ylläpitäjät voivat ottaa etäyhteyden palvelimeen, mutta sisäverkon ulkopuolelta se on estetty.

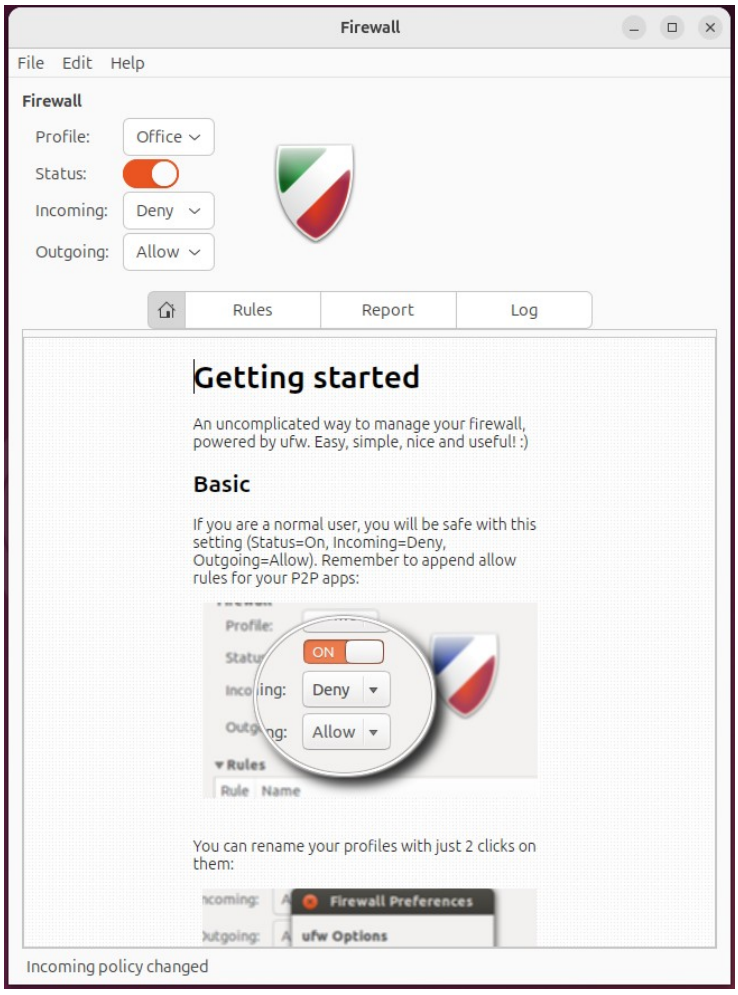
Monessa Linux-jakelussa palomuuritoiminnallisuus on nykyään vakiona asennettuna, jonkinlaisilla oletuskonfiguraatioilla. Näitä on kuitenkin syytä tarkastella, jotta hyökkäyspinta-ala saadaan minimoitua. Käytännön hyvä taktiikka palvelinkäytössä on estää kaikki liikenne mitä ei oikeasti tarvita.

UFW (Uncomplicated Firewall)
Helppo, mutta rajoitettu

Tässä osiossa otetaan käyttöön UFW-palomuuriratkaisu. UFW ei itsessään ole palomuuri, vaan käyttöliittymä ja rajapinta jolla voidaan helposti hallita netfilterin tai iptablesin palomuurio minaisuuksia.

UFW on hyvin suosittu paketti palomuurien hallintaan, sillä se on hyvin yksinkertainen ja helppo omaksua myös tavan käyttäjälle. Esimerkiksi Ubuntussa UFW on oletuksena asennettuna. Se ei myöskään vaadi mitään erityisiä konfiguraatioita. Palvelun aktivoiminen käyttöjärjestelmässä ja UFW:n käyttöliittymässä riittää palomuurin käynnistämiseksi.

UFW on komentorivipohjainen ohjelma, mutta siitä on saatavilla myös GUFW-versio graafiselle käyttöliittymälle. GUFW:ssä on myös valmiita alueita, kuten koti- ja työ-alueet, joissa tavallisimmat konfiguraatiot ovat helposti saatavilla.



Kuva 26: GUFW -käyttöliittymä

Euroopan unionin osarahoittama

Tarvittaessa UFW voidaan asentaa seuraavasti:

```
$ apt update && apt upgrade -y
$ apt install ufw
```

Jos UFW-komentojen suorittaminen epäonnistuu, tulee nämä suorittaa `sudo` -komentoa hyödyntäen.

Oletuksena UFW:n tulisi olla inaktiivinen. Jos palvelimelle ollaan etäyhteydessä esim. SSH:lla, emme vielä aktivoi UFW:tä.

```
$ ufw status
Status: inactive
```

Ensimmäisenä haluamme vaihtaa UFW:n sääntöihin SSH-portin vastaamaan SSH:n konfiguraatiota. Jos porttia ei aseteta oikein ja käynnistämme palomuurin, saattaa yhteys palvelimelle katketa välittömästi. Yhteyden katkettua muutosta ei pääse korjaamaan, ainakaan SSH-yhteydellä.

Vaihdetaan SSH:n portti UFW:hen, tekemällä vaikkapa ’`custom`’ -niminen tiedosto sääntöjä varten:

```
$ cat /etc/ufw/applications.d/openssh-server | tee
/etc/ufw/applications.d/custom
```

Yllä oleva putkitettu komento tulostaa ensin OpenSSH-Serverin oletus-konfiguraation terminaaliin. Tämän onnistuttua `tee` -komento tekee tulosteesta uuden tiedoston valitsemaamme polkuun.

Luotuun tiedostoon muutetaan palvelimen SSH:n konfiguraatiota vastaava portti. Tämä sääntö tulee myös nimetä uudelleen, jotta se ei sekoitu oletussääntöön:

```
1 [Custom-OpenSSH]
2 title=Secure shell server, an rshd replacement
3 description=OpenSSH is a free implementation of the Secure Shell protocol.
4 ports=22022/tcp
```

Nyt meillä on käytössä kaksi eri sääntöä SSH:lle; OpenSSH ja Custom-OpenSSH. Custom-tiedosto luotiin erikseen, sillä jos muutamme UFW:n luomia sääntötiedostoja suoraan, tulemme menettämään kaikki muutokset UFW-paketin päivittyessä. Omia sääntöjä on myös huomattavasti helpompi hallita omasta tiedostostaan.

Luotu sääntö voidaan nyt sallia seuraaavalla komennolla:

```
$ ufw allow Custom-OpenSSH
```

Varmuuden vuoksi voimme estää oletussäännöt, jotta portti `:22` on varmasti estettynä:

```
$ ufw deny OpenSSH
```

Haluamme pelata varman päälle ja varmistamme SSH-yhteyden toiminnan palomuurin kanssa. Tätä varten tulee olla auki 2 erillistä terminaalia, joista toinen ei ole vielä SSH-yhteydessä palvelimeen.

```
$ ufw enable && sleep 60 && ufw disable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

Komento laittaa ensimmäisenä UFW-palomuurin päälle (älä unohda ajaa molempia UFW-komentoja `sudo`:lla). Sen jälkeen shell odottaa 60 sekuntia, jonka aikana voidaan testata toimiiko SSH-yhteyden luominen toisella terminaalilla. Jos yhteys toimii normaalisti, on kaikki kuten haluamme. Lopulta komento automaattisesti sulkee palomuurin, jolloin yhteydet taas sallitaan ja mahdollinen rikkinäinen sääntö päästään korjaamaan.

Kyseinen komentojen ketjuttaminen on hyvä tapa kokeilla palomuurisääntöjen toimintaa käytännössä, sillä mahdollinen epätoivottu muutos perutaan automaattisesti.

Kun olemme varmoja sääntöjen toiminnasta, voimme asettaa palomuurin päälle:

```
$ ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Asetetut säännöt päästään tarkastamaan komennolla:

```
$ ufw status numbered
Status: active
     To                         Action      From
     --                         ------      ----
[ 1] Custom-OpenSSH             ALLOW IN    Anywhere
[ 2] OpenSSH                    DENY IN     Anywhere
[ 3] Custom-OpenSSH (v6)        ALLOW IN    Anywhere (v6)
[ 4] OpenSSH (v6)               DENY IN     Anywhere (v6)
```


Euroopan unionin osarahoittama

Komennossa `numbered` on käytännöllinen lisäkomento, sillä nyt näemme säännöille asetetut numerot. Tarvittaessa sääntöjä on helppo muuttaa tai poistaa numeron perusteella. Lista kuitenkin päivittyy joka kerta kun sääntöjä poistetaan. Tämän takia lista kannattaa tulostaa jokaisen muutoksen jälkeen, jotta numerot eivät mene sekaisin.

SSH-yhteydet ovat nyt sallittuna kaikkialta porttiin `:22022`. Haluamme kuitenkin rajoittaa yhteydet ainoastaan omaan IPv4-aliverkkoomme. Tämä voidaan tehdä kirjoittamalla tarkempi sääntö:

```
$ ufw insert 1 allow from 192.168.122.0/24 proto tcp to any port 22022
```

Komennon selitykset:

`insert <luku>`: asettaa säännön tiettyyn kohtaan sääntölistassa

`allow/deny/reject/limit`: säännön luonne

`from <IP-osoite/aliverkko/palvelu>`: mihin ulkopuoliseen IP-osoitteeseen/aliverkkoon ja/tai palveluun sääntöä sovelletaan

`proto <protokolla>`: millä protokollalla yhteys sallitaan

`to <IP-osoite/aliverkko/palvelu>`: mihin sisäiseen IP-osoitteeseen/aliverkkoon ja/tai palveluun sääntöä sovelletaan

`port`: mihin porttiin sääntöä sovelletaan

`insert 1` on tärkeä osa komentoa, sillä sen avulla sääntö laitetaan listan ensimmäiseksi. Koska säännöt tarkistetaan järjestyksessä pienimmästä suurimpaan, haluamme tämän säännön listan kärkeen.

Nyt sääntölistaus näyttää seuraavalta:

```
$ ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22022/tcp                  ALLOW IN    192.168.122.0/24
[ 2] Custom-OpenSSH             ALLOW IN    Anywhere
[ 3] OpenSSH                    DENY IN     Anywhere
[ 4] Custom-OpenSSH (v6)        ALLOW IN    Anywhere (v6)
[ 5] OpenSSH (v6)               DENY IN     Anywhere (v6)
```

Jotta voimme estää yhteydet muualta, voimme muokata muut säännöt estämään loput yhteydet muista verkoista. Jos olet epävarma oletko asettanut oman sisäverkkosi osoitteen oikein, muista komentojen ketjuttaminen!

```
$ ufw deny Custom-OpenSSH && sleep 60 && ufw allow Custom-OpenSSH
$ ufw deny Custom-OpenSSH
$ ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22022/tcp                  ALLOW IN    192.168.122.0/24
[ 2] Custom-OpenSSH             DENY IN     Anywhere
[ 3] OpenSSH                    DENY IN     Anywhere
[ 4] Custom-OpenSSH (v6)        DENY IN     Anywhere (v6)
[ 5] OpenSSH (v6)               DENY IN     Anywhere (v6)
```

Palvelimellamme pyörivä Wordpress-sivusto on konfiguroitu toimimaan HTTPS-protokollan yli. Tämä protokolla käyttää porttia `:443`, joten se tulee sallia palomuurissa. Lisäksi olemme konfiguroineet web-palvelimen ohjaamaan kaikki HTTP-protokollaa käyttävät pyynnöt HTTPS-sivustolle. Voimme halutessamme sallia siis myös portin `:80`.

```
$ ufw allow 443 && ufw allow 80
$ ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22022/tcp                  ALLOW IN    192.168.122.0/24
[ 2] Custom-OpenSSH             DENY IN     Anywhere
[ 3] OpenSSH                    DENY IN     Anywhere
[ 4] 80                         ALLOW IN    Anywhere
[ 5] 443                        ALLOW IN    Anywhere
[ 6] Custom-OpenSSH (v6)        DENY IN     Anywhere (v6)
[ 7] OpenSSH (v6)               DENY IN     Anywhere (v6)
[ 8] 80 (v6)                    ALLOW IN    Anywhere (v6)
[ 9] 443 (v6)                   ALLOW IN    Anywhere (v6)
```


Euroopan unionin osarahoittama

Kun olemme tyytyväisiä sääntöihin, voimme asettaa palomuurin hylkäämään kaikki niihin kuulumattomat pyynnöt. Jälleen, muutoksen peruvan komennon voi ketjuttaa, jos sen vaikutuksista ollaan epävarmoja.

```
$ ufw default reject && sleep 60 && ufw default allow
...
$ ufw default reject
Default incoming policy changed to 'reject'
```

Lopuksi voimme varmuuden vuoksi ladata palomuurin uudelleen, jolloin kaikki konfiguraatiot tulevat varmasti voimaan:

```
$ ufw reload
Firewall reloaded
```

`ufw.service` tulee myös asettaa aktiiviseksi, jotta palomuuri on käytössä jokaisella uudelleenkäynnistyksellä:

```
$ systemctl enable ufw && systemctl start ufw
Synchronizing state of ufw.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ufw


$ systemctl status ufw
● ufw.service - Uncomplicated firewall
     Loaded: loaded (/lib/systemd/system/ufw.service; enabled; preset: enabled)
     Active: active (exited) since Wed 2025-06-04 17:18:52 EEST; 5s ago
       Docs: man:ufw(8)
    Process: 9664 ExecStart=/lib/ufw/ufw-init start quiet (code=exited,
status=0/SUCCESS)
   Main PID: 9664 (code=exited, status=0/SUCCESS)
        CPU: 12ms
```

Jos haluamme varmistaa että säännöt varmasti toimivat, se onnistuu jälleen ketjuttamalla komentoja:

```
$ ufw delete 1 && ufw reload && sleep 60 && ufw insert 1 allow from
192.168.122.0/24 proto tcp to any port 22022 && ufw reload
```

Lähteet:

https://www.linux.fi/wiki/Ufw

https://wiki.archlinux.org/title/Uncomplicated_Firewall

man ufw-framework

https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands

man ufw

firewalld (nftables backend)
Tekninen, ominaisuuksiltaan voimakas

Tässä osioissa otetaan käyttöön nftables ja firewalld -pohjainen palomuuriratkaisu. Netfilter Projectin nftables on ”uusi” verkkoliikenteen hallintajärjestelmä Linuxille, johon on yhdistetty ja modernisoitu vanhojen ratkaisujen sekamelska (iptables, ip6tables, arptables, ebtables). Nftables on saatavilla Linux-käyttöön kernel versiosta 3.13 eteenpäin, eikä vanhempia ratkaisuja suositella enää otettavan käyttöön. Red Hatin kehittämä Firewalld on puolestaan eräänlainen rajapinta suoralla integraatiolla moneen Linuxin palomuuri-, verkko-, IPS-järjestelmään. Nftables toimii firewalld:n backendinä, vaikka firewalld on varsinaisesti käyttämämme palomuurijärjestelmä.

Seuraava ohjeistus on tehty Debian 12 -jakelulla. Muita jakeluita käyttäessä, erityisesti jos nämä eivät ole Debian-pohjaisia, saattaa joissakin oleellisissa konfiguraatioissa ja komennoissa olla eroja. Esimerkiksi Red Hat -pohjaisissa jakeluissa firewalld on käytössä automaattisesti, yleensä ylimääräisillä kustomoiduilla alueilla.

Toimintaan tutustuminen

Linux-jakelusta riippuen palomuurijärjestelmä saattaa hyvinkin olla valmiiksi asennettuna, sekä toiminnassa. Jakeluita on kuitenkin hyvin suuri määrä, eikä ole mitenkään tavatonta että minkäänlaista ratkaisua ei oletuksena ole. Linux-kernelissä ”oletuksena” nftables on nykyään käytössä oleva järjestelmä. Jakelussa tämä voidaan varmistaa seuraavalla komennolla:

```
$ systemctl status nftables
○ nftables.service - nftables
     Loaded: loaded (/lib/systemd/system/nftables.service; disabled; preset:
      enabled)
     Active: inactive (dead)
       Docs: man:nft(8)
             http://wiki.nftables.org
```

Esimerkkitapauksessa nftables on asennettuna ja inaktiivisessa tilassa. Jos nftables ei ole asennettuna, tuloste näyttää vastaavalta:

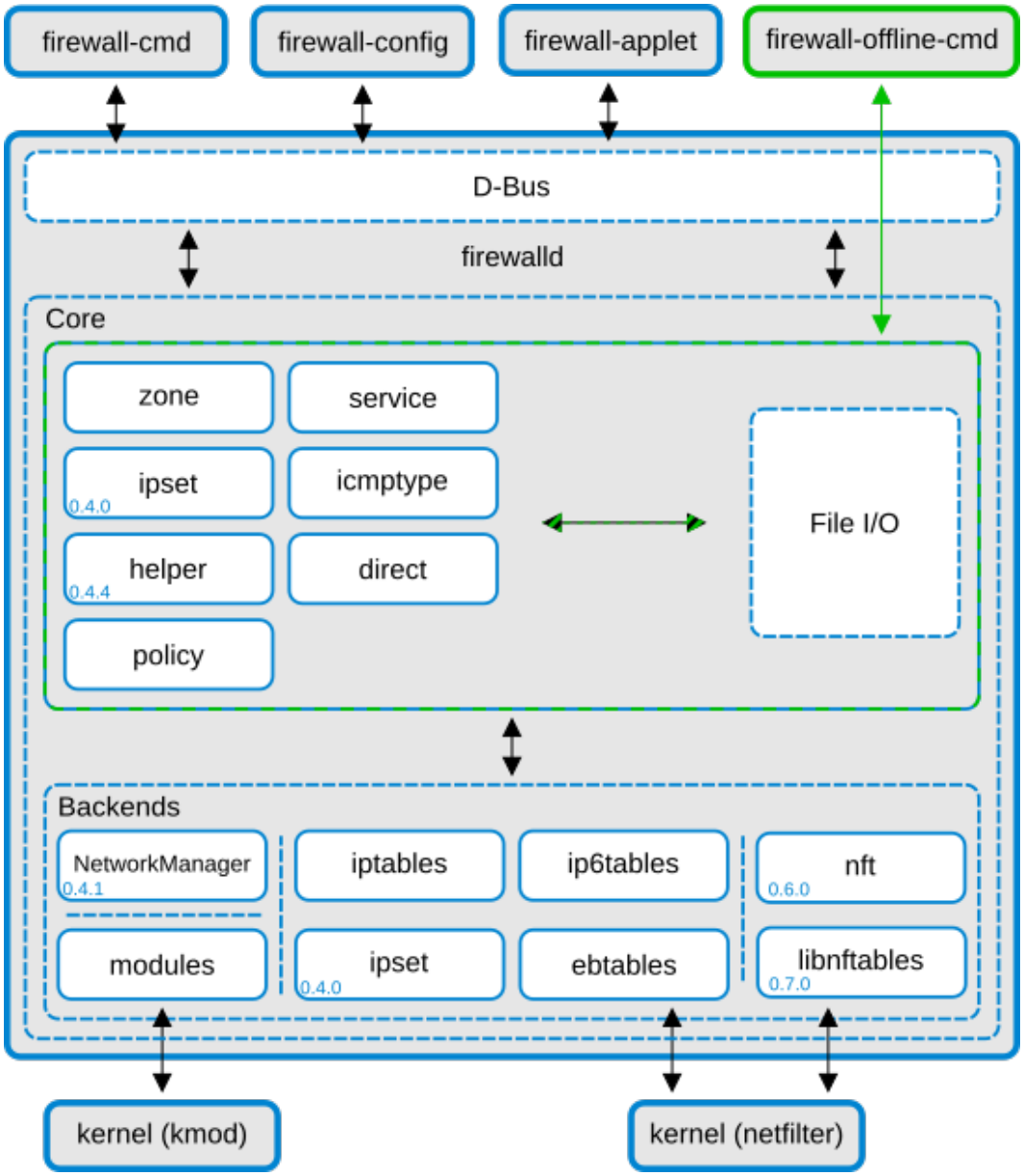
```
Unit nftables.service could not be found.
```

Tarpeen tullen nftables voidaan asentaa pakettirepositoriosta:

```
$ apt update && apt upgrade -y
$ apt install nftables
```


Euroopan unionin osarahoittama

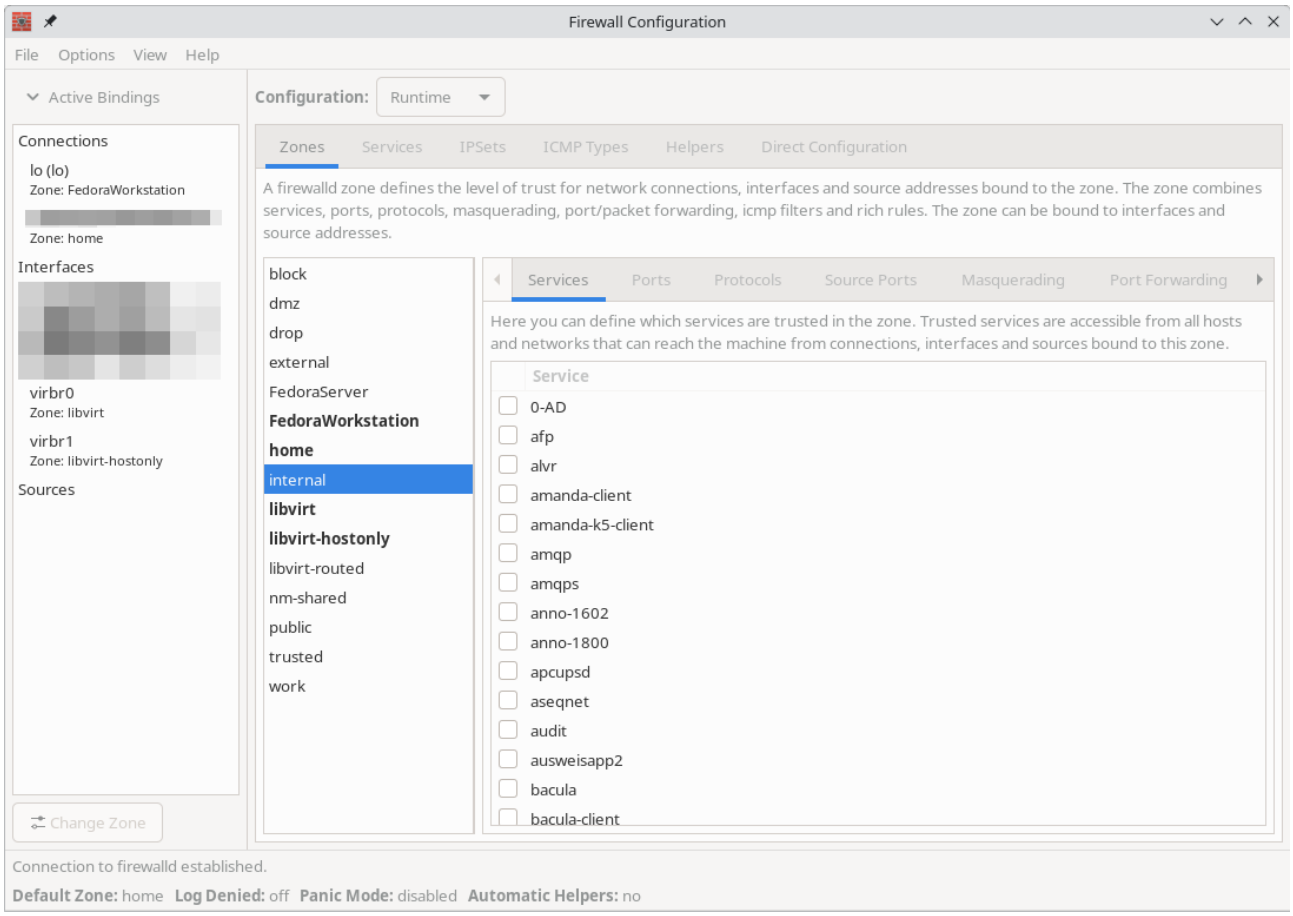
Koska käyttöön otetaan firewalld, joka toimii pääasiallisena palomuurina, tulee nftables jättää inaktiiviseen tilaan. Firewalld käyttää nftablesin pakettienhallintaa ja -suodatusta palomuurin toiminnallisuuden toteuttamiseen. Se tapahtuu kuitenkin käyttöjärjestelmän arkkitehtuurissa huomattavasti syvemmällä tasolla, eikä molemmat palvelut voi olla päällepäin samanaikaisesti aktiivisia.



Kuvaaja 3: Firewalld:n arkkitehtuuri käyttöjärjestelmän sisällä
Lähde: https://firewalld.org/documentation/architecture.html (Unlicense)

Firewalld helpottaa nftables:in konfigurointia huomattavasti. Peruskäyttöliittymä toimii komentorivin kautta, mutta graafiselle käyttöliittymälle on saatavilla `firewall-config` -paketti. Allekirjoittaneen kokemuksen mukaan graafinen käyttöliittymä sisältää kuitenkin satunnaisia bugeja, joten tarkempi konfigurointi on varmempaa komentoriviltä. Komentoriviohjelmisto kannattaa olla muutenkin hyvin hallinnassa, sillä graafista

käyttöliittymää ei ole läheskään aina mahdollista käyttää palvelinympäristössä. Firewall-config on kuitenkin työpöytäkäyttöön kätevä lisä.



Kuva 27: Firewall-config työpöytäkäyttöliittymä

Jos firewalld ei ole asennettuna, voidaan se asentaa pakettirepositoriosta:

```
$ apt update && apt upgrade -y
$ apt install firewalld
```

Asennuksen jälkeen firewalld:n tulisi olla automaattisesti aktiivisena:

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; enabled; preset:
enabled)
     Active: active (running) since Fri 2025-06-06 11:45:40 EEST; 11s ago
       Docs: man:firewalld(1)
   Main PID: 2380 (firewalld)
      Tasks: 2 (limit: 9416)
     Memory: 30.7M
...
```


Euroopan unionin osarahoittama

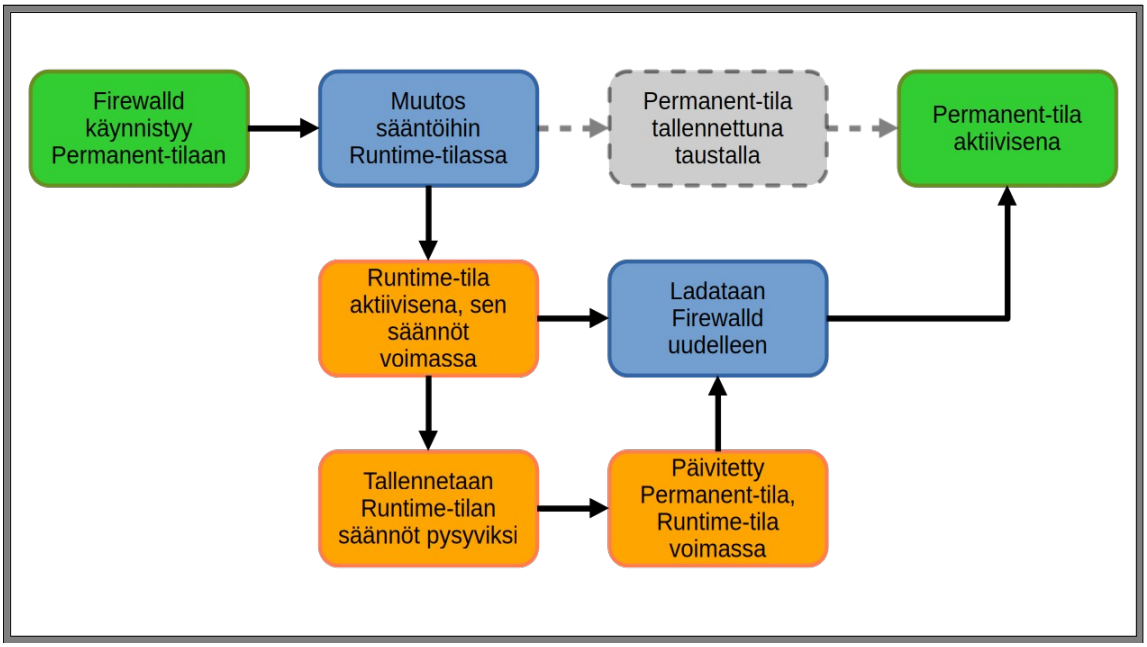
firewall-cmd

Firewalld:n sääntöjä päästään tarkastelemaan ja muokkaamaan `firewall-cmd` -komennolla. Komento sisältää melko massiivisen listan flageja:

```
$ firewall-cmd --help
Usage: firewall-cmd [OPTIONS...]

General Options
  -h, --help            Prints a short help text and exits
  -V, --version         Print the version string of firewalld
  -q, --quiet           Do not print status messages

Status Options
  --state               Return and print firewalld state
  --reload              Reload firewall and keep state information
  --complete-reload     Reload firewall and lose state information
  --runtime-to-permanent
                        Create permanent from runtime configuration
  --reset-to-defaults
                        Reset configuration to firewalld's default configuration
  --check-config        Check permanent configuration for errors

Log Denied Options
  --get-log-denied      Print the log denied value
  --set-log-denied=<value>
                        Set log denied value

Permanent Options
  --permanent           Set an option permanently
                        Usable for options marked with [P]

Zone Options
  --get-default-zone    Print default zone for connections and interfaces
  --set-default-zone=<zone>
                        Set default zone
  --get-active-zones    Print currently active zones
  --get-zones           Print predefined zones [P]
  --get-services        Print predefined services [P]
  --get-icmptypes       Print predefined icmptypes [P]
  --get-zone-of-interface=<interface>
...
```

Tätä ei kuitenkaan kannata pelästyä; tai ainakaan siitä ei kannata lannistua. Komennot ovat lopulta melko verbooseja, eikä läheskään kaikkia tarvitse muistaa ulkoa. Käytännössä tärkeimmät komennot oppii hyvin nopeasti. Hyvä taktiikka on myös pitää kahta terminaalia avoinna vierekkäin. Toisessa näistä voi olla `firewall-cmd --help` -listaus jatkuvasti saatavilla, kun itse komennot tehdään toisessa terminaalissa.

Firewalld:n säännöt toimivat kahdessa tilassa: `Runtime` ja `Permanent`. Runtime-tilassa tehdyt muutokset ovat voimassa reaaliaikaisesti, eivätkä jää automaattisesti pysyvään käyttöön. Permanent-tila puolestaan on voimassa taustalla, luoden oletustilan palomuurisäännöille. Kun palvelin käynnistetään, otetaan Permanent-tilan säännöt käyttöön. Muutokset Runtime-tilassa yliajavat Permanent-tilan, mutta nämä kaikki voidaan perua siirtymällä Permanent-tilaan tallentamatta muutoksia.



Kuvaaja 4: Firewalld:n tilojen toimintalogiikka

Jos teemme joitakin muutoksia, mutta emme ole varmoja niiden käytännön vaikutuksista, kannattaa ne tehdä Runtime-tilassa. Jos kaikki toimii kuten pitää, voidaan nämä muutokset siirtää Permanent-tilaan. Runtime-tilaa voidaan kuitenkin hyödyntää vain väliaikaisiin muutoksiin. Jos esimerkiksi portti `:32463` tulee avata väliaikaisesti, voidaan se tehdä ilman pysyviä muutoksia Runtime-tilassa.

Sääntömuutoksissa kannattaa tilasta huolimatta olla varovainen. Jos olemme etäyhteydessä palvelimelle esimerkiksi VPN-tunnelin kautta, ja estämme kaiken TCP-liikenteen palvelimelle, saatamme välittömästi menettää yhteyden palvelimeen. Vaikka firewalld olisi Runtime-tilassa, tätä muutosta ei voi enää perua, jolloin sääntö on korjattava muulla keinolla. Firewalld on kuitenkin ratkaisuna suhteellisen fiksu, sillä jo avoimena olevat yhteydet pyritään pitämään auki. Tämän takia SSH:n kautta konfiguroidessa kannattaa pitää avoinna yksi terminaali, jota voidaan käyttää hätävarana tilanteessa jossa jokin menee vikaan.

Euroopan unionin osarahoittama

Alueet (zone)

Firewalld toimii alueiden (`zone`) perusteella. Eri käyttötarkoituksia varten voidaan määritellä omia alueita, jotka puolestaan sisältävät eri sääntöjä. Alueiden sisällä voidaan säätää sallittuja IP-osoiteavaruuksia tai aliverkkoja, avoimia portteja, sallittuja protokollia ja palveluita.

Oletuksena luodut alueet tulostetaan seuraavalla komennolla:

```
$ firewall-cmd --list-all-zones
block
  target: %%REJECT%%
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

dmz
  target: default
  icmp-block-inversion: no
  interfaces:
...
```

Yleisesti ottaen alueet sidotaan verkkokorttiin, joka voi olla todellinen tai virtuaalinen. Käytössä olevat alueet merkitään `(active)` -merkinnällä. Tässä tapauksessa `public` -alue on sidottuna `enp1s0` -verkkokorttiin:

```
...
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: dhcpv6-client ssh
...
```

Palvelut (services)

Palvelusäännöt ovat firewalld:ssä osin valmiiksi määriteltyjä sääntöjä, jotka koskevat jotain tiettyä käyttöjärjestelmän palvelua. Näillä säännöillä voidaan helposti sallia tiettyjä ominaisuuksia, kuten HTTPS-yhteydet web-palvelimelle. Sääntöjä voidaan myös muokata tarpeen mukaan. Jos jokin palvelu on sallittu, mutta eri portissa kuin oletuksena, voidaan se helposti muuttaa valmiiseen sääntöön ilman suurempia konfigurointeja.

Firewalld:ssä palvelut voidaan ottaa käyttöön reaaliajassa. Jos vaikkapa RDP-yhteydet halutaan sallia vain väliaikaisesti, voidaan palvelu sallia Runtime-tilassa. Kun tarvetta yhteydelle ei enää ole, voidaan se vastaavasti sulkea.

Myös kustomoituja palvelusääntöjä voidaan luoda. Tämä voidaan tehdä komentoriviltä `--new-service=<service>` -flagilla. Tämän jälkeen sääntöä voidaan muokata halutulla tavalla. Vastaavasti sääntö voidaan luoda omaan XML-tiedostoonsa, josta se tuodaan firewalld:hen. Tuonti voidaan tehdä `--new-service-from-file=<filename> --name=<service>` -flageilla. Tiedoston sisältö voi olla seuraavankaltainen:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <service>
3   <short>SSH</short>
4   <description>Secure Shell (SSH) is a protocol for logging into and executing
      commands on remote machines. It provides secure encrypted communications.
      If you plan on accessing your machine remotely via SSH over a firewalled
      interface, enable this option. You need the openssh-server package
      installed for this option to be useful.</description>
5   <port port="22022" protocol="tcp"/>
6 </service>
```

Kustomoidut säännöt tallennetaan yleensä `/etc/firewalld/services/` -hakemistoon.

Rich rules

Rich rules on abstrakti syntaksi yksityiskohtaisten sääntöjen luomiseen. Jos on tarve sallia tiettyyn palveluun yhdistäminen, ainoastaan tietyistä IP- ja MAC-osoitteista, ohjaten nämä eteenpäin toisaalle, samalla logaten ne erikseen, onnistuu se rich rulesin avulla.

Rich rules -sääntö voi sisältää seuraavanlaisia asetuksia:

```
# rule [family="rule family"]
#     [ source [NOT] [address="address"] [mac="mac-address"] [ipset="ipset"] ]
#     [ destination [NOT] address="address" ]
#     [ element ]
#     [ log [prefix="prefix text"] [level="log level"]
          [limit value="rate/duration"] ]
#     [ audit ]
#     [ action ]
```


Euroopan unionin osarahoittama

Aluekohtaisia rich rules -sääntöjä voidaan luoda `--zone=<alue> --add-rich-rule='<säännön sisältö>'` -flageilla.

Toiminta käytännössä

Firewalld:n toiminta tulee parhaiten selville, kun luomme palvelintamme varten uuden alueen omilla säännöillään. Tähän mennessä olemme ottaneet palvelimella käyttöön SSH-palvelun sekä HTTPS-protokollaa käyttävän Wordpress-sivuston.

SSH-yhteydet halutaan sallia ainoastaan omasta lähiverkosta, HTTPS-liikenne kaikkialta verkosta. Voimme myös olettaa, että DHCPv6-palvelu on käytössämme hyödyllinen. Haluamme kuitenkin turvata palvelimen muilta osilta niin, että palomuuri estää yhdistämisen tarpeettomiin portteihin ja palveluihin.

Tietyt komennot vaativat aina `--permanent` flagin toimiakseen. Tämä johtuu sääntöjen luonteesta, jolloin esimerkiksi uuden alueen luominen ainoastaan Runtime-tilaan olisi epäkäytännöllistä. Nämä komennot pystyy tarkistamaan `firewall-cmd --help` -komennolla. Moni komennoista vaatii myös `sudo`:n käyttöä, sillä komennot muokkaavat asioita huomattavasti tavallisen käyttäjän oikeuksia syvemmällä tasolla.

Palvelut (services)

Allekirjoittanut on aloittanut etäyhteyden palvelimeen SSH:lla ennen firewalld:n asennusta. Koska firewalld:n säännöissä SSH-palvelun portti on `:22` ja SSH on konfiguroitu käyttämään porttia `:22022`, tulee tämä muuttaa. Olemassa oleva tunneli ei katoa, mutta uusia yhteyksiä ei saada tämän takia tehtyä. **Tämän takia ainakin yksi SSH-terminaali kannattaa pitää aina avoinna palvelimelle, jotta mahdolliset vahingot päästään korjaamaan.**

```
$ firewall-cmd --service=ssh --add-port=22022/tcp --permanent
$ firewall-cmd --service=ssh --remove-port=22/tcp --permanent
$ firewall-cmd --reload
```

Uuden SSH-yhteyden muodostamisen tulisi taas toimia. SSH-palvelun muutetut säännöt voidaan varmistaa komennolla:

```
$ firewall-cmd --info-service=ssh --permanent
ssh
  ports: 22022/tcp
  protocols:
  source-ports:
  modules:
  destination:
  includes:
  helpers:
```

Haluamme luoda palvelimelle uuden alueen, johon konfiguroimme tarvittavat säännöt. Tämä tapahtuu komennolla:

```
$ firewall-cmd --new-zone=aimserver --permanent
```

Alueen tiedot voidaan nyt tarkistaa. Tulosteessa näemme että mitään erityisiä sääntöjä ei ole vielä olemassa:

```
$ firewall-cmd --info-zone=aimserver --permanent
aimserver
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Seuraavaksi voimme ottaa alueelle käyttöön tarvitsemamme palvelut. Huomaa, että SSH-palvelua ei vielä oteta käyttöön, sillä aiomme asettaa sille myöhemmin yksityiskohtaisempia sääntöjä.

Sallitaan DHCPv6-Client, jotta toiminta DHCP-palvelimen kanssa olisi virheetöntä:

```
$ firewall-cmd --zone=aimserver --add-service=dhcpv6-client --permanent
```

DHCP on protokolla, joka automaattisesti tarjoaa aliverkon laitteille tarpeelliset tiedot verkkoon yhdistämistä varten. Tämän vuoksi esimerkiksi IP-osoitetta ei ole välttämätöntä kovakoodata laitteen tietoihin. DHCPv4 toimii IPv4-protokollan yhteydessä, DHCPv6 puolestaan IPv6-protokollan kanssa. DHCPv4-toiminnallisuus on rakennettuna Linux-kerneliin niin, ettei sitä tarvitse sallia erikseen. Tarvittaessa DHCPv6-Client tulee sallia palomuurissa, jotta tämä toimisi oikein.

Euroopan unionin osarahoittama

Sallitaan HTTPS-palvelu, jotta suojattu liikenne palvelimen web-sivustolle on mahdollista:

```
$ firewall-cmd --zone=aimserver --add-service=https --permanent
```

Tarkastamalla alueen tiedot, voimme todeta muutosten olevan kuten haluamme:

```
$ firewall-cmd --info-zone=aimserver --permanent
aimserver
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client https
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Rich rules

Haluamme estää SSH:n ja SFTP:n käytön kaikista muista osoiteavaruuksista, paitsi omasta lähiverkostamme. Tämä voidaan toteuttaa `rich rules` -sääntöjä hyödyntäen. `Rich rules` on ominaisuus ja syntaksi, jolla voidaan yhdistää eri sääntöjä yhteen kokonaisuuteen. Tästä kokonaisuudesta muodostuu yksityiskohtainen sääntö, jonka toiminta on kustomoitu juuri tarvitsemaamme käyttöön.

SSH:n yhteysrajoitukset saadaan aikaan seuraavilla komennoilla:

```
$ firewall-cmd --zone=aimserver --add-rich-rule='rule family=ipv4 priority=0
source address=192.168.122.0/24 service name=ssh accept' --permanent
$ firewall-cmd --zone=aimserver --add-rich-rule='rule family=ipv4 priority=1
service name=ssh reject' --permanent
$ firewall-cmd --zone=aimserver --add-rich-rule='rule family=ipv6 priority=1
service name=ssh reject' --permanent
```

Komentojen rich rules flagien selitykset:

`family=ipv4 | ipv6`: mihin IP-protokollan versioon sääntöä sovelletaan

`priority=0 | 1 | 2 | ...` : millä prioriteetilla sääntö tarkistetaan

`source=ip-osoite | mac-osoite | ipset`: sisääntuleva liikenne - mitä osoitetta sääntö koskee

`destination=ip-osoite | ipset`: ulospäin kulkeva liikenne - mihin osoitteisiin tämä sallitaan

`service name=palvelu`: minkä nimiseen palveluun sääntöä sovelletaan

`accept | reject | drop | mark`: mikä on säännön luonne

Ensimmäisen säännön prioriteetiksi asetetaan `0`, eli se tarkistetaan ensimmäisenä. Yhdistäminen hyväksytään, jos yhteys yritetään luoda määritetystä aliverkosta SSH-palveluun.

Muissa tapauksissa prioriteetti on `1`. Jos aiempien prioriteettien säännöt eivät toteudu, siirrytään prioriteettilistassa ylöspäin. Luodut säännöt estävät kaikki yritykset yhdistää SSH-palveluun.

Euroopan unionin osarahoittama

Säännöt voidaan tarkistaa komennolla:

```
$ firewall-cmd --info-zone aimserver --permanent
aimserver
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client https
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="192.168.122.0/24" service name="ssh"
      accept
        rule priority="1" family="ipv4" service name="ssh" reject
        rule priority="1" family="ipv6" service name="ssh" reject
```

Tähän mennessä kaikki muutokset on tehty Permanent-tilaan. Tämä on järkevää, sillä luotu alue on uusi eikä vielä käytössä. Muutokset eivät siis pääse vaikuttamaan yhteyksien toimintaan. Jotta muutokset voidaan ottaa käyttöön myös Runtime-tilassa, tulee firewalld ladata uudelleen:

```
$ firewall-cmd --reload
success
```

Alueet (zone) ja target

Muutokset tuotiin Runtime-tilaan tässä vaiheessa kahdesta syystä. Jatkossa haluamme varmistaa muutosten toiminnan ilman turhaa tallentamista; lisäksi muutokset saattavat rikkoa jotakin, jolloin Runtime-tila on turvallisempi vaihtoehto. Näiden sääntömuutosten eron tunnistaa `--permanent` -flagista (tai sen puutteesta).

Tarpeen vaatiessa Runtime-tilasta voidaan siirtyä takaisin Permanent-tilaan `--reload` -flagilla. Niin kauan kun muutoksia ei ole erikseen tehty Permanent-tilaan, tai niitä ei ole erikseen siirretty sinne, tulevat nämä peruuntumaan firewalld uudelleenladattaessa:

```
$ firewall-cmd --reload
```

Ensimmäisenä tulee selvittää mikä verkkokortti on firewalld:n ja palvelimen aktiivisessa käytössä:

```
$ firewall-cmd --get-active-zones
public
  interfaces: enp1s0
```

Tämän jälkeen verkkokortin alue vaihdetaan uuteen alueeseen:

```
$ firewall-cmd --zone=aimserver --change-interface=enp1s0
```

Kun käytössä olevan verkkokortin alue vaihdetaan, siirtyy alue samantien aktiiviseen tilaan. Tämän takia muutoksia ei vielä tehdä Permanent-tilaan, sillä voimme nyt helposti palata toimivaan tilaan.

```
$ firewall-cmd --get-active-zones
aimserver
  interfaces: enp1s0
```

Voimme vielä varmistaa, että kaikki säännöt ovat kuten haluamme:

```
$ firewall-cmd --info-zone=aimserver
aimserver (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: dhcpv6-client https
  ports:
  protocols:
  forward: no
  masquerade: no
```


Euroopan unionin osarahoittama

```
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="192.168.122.0/24" service name="ssh"
      accept
        rule priority="1" family="ipv6" service name="ssh" drop
        rule priority="1" family="ipv4" service name="ssh" drop
```

Kun olemme varmoja että kaikki toimii Runtime-tilassa, voimme siirtää Runtime-konfiguraation Permanent-tilaan:

```
$ firewall-cmd --runtime-to-permanent
success
```

Koska tarkoituksena on, ettei palvelimelle voi ottaa yhteyttä muuten kuin haluamillamme tavoilla, vaihdamme lopuksi target-säännön. `REJECT`-sääntö estää kaiken liikenteen, ellei muualla säännöissä ole määritelty toisin.

```
$ firewall-cmd --zone=aimserver --set-target=REJECT --permanent
success
```

Firewalld:n target-säännöt:

`ACCEPT`: sallii kaiken liikenteen, ellei muualla säännöissä määritellä toisin

`REJECT`: estää kaiken liikenteen, ellei muualla säännöissä määritellä toisin

`DROP`: pudottaa kaiken liikenteen, ellei muualla säännöissä määritellä toisin. DROP on ikäänkuin REJECT, mutta ei anna pakettien estämisestä mitään virhetietoja muille laitteille

Asetamme vielä luomamme alueen oletus-alueeksi, jotta tämä käynnistetään automaattisesti palvelimen uudelleenkäynnistysten yhteydessä:

```
$ firewall-cmd --set-default-zone=aimserver
success
```

Lopuksi käynnistämme firewalld:n uudelleen, jolloin muutokset tulevat varmasti voimaan:

```
$ firewall-cmd --reload
success
```

Lähteet:

https://www.linux.fi/wiki/NFTables

https://wiki.archlinux.org/title/Nftables

https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F

https://firewalld.org/

https://wiki.debian.org/nftables#What_is_nftables.3F

https://www.linux.fi/wiki/FirewallD

https://firewalld.org/documentation/man-pages/firewalld.richlanguage.html

man firewalld.richlanguage

https://serverfault.com/questions/157375/reject-vs-drop-when-using-iptables

https://github.com/firewalld/firewalld/issues/590#issuecomment-605200548

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/security_guide/configuring_complex_firewall_rules_with_the_rich-language_syntax

man firewalld

man firewall-cmd

man firewalld.zone

Intrusion Prevention Software

Intrusion Prevention System on yksi tietoverkkojen oleellisimpia turvallisuutta lisääviä ratkaisuja. Kyseisen järjestelmän tehtävänä on käydä läpi verkkoliikennettä, suodattaen sieltä epätoivottua liikennettä, joka mahdollisesti vaarantaa turvallisen toiminnan tietoverkossa. Usein nämä järjestelmät toimivat ainakin lähiverkon ulkolaidalla, ollen omia varsinaisia laitteitaan.

Pienemmässä skaalassa, kuten yksittäisellä palvelimella tai työasemalla, voidaan ottaa käyttöön Intrusion Prevention Software. Tämän toiminta on perusperiaatteeltaan hyvin samankaltaista; tosin liikenteen skaala on paljon pienempi.

Palvelinkäytössä kyseiselle ohjelmistolle voidaan asettaa sääntöjä, jotka estävät epätoivotun toiminnan verkon ylitse. Ohjelmisto voidaan konfiguroida toimimaan yhteistyössä palomuurin kanssa, jolloin epämääräiset porttiskannaukset tai bruteforce-hyökkäykset saadaan pysäytettyä.

Tässä osiossa otetaan käyttöön nimenomaisesti Intrusion Prevention Software -järjestelmä.

Fail2Ban

Fail2Ban on yksi suosituimmista Intrusion Prevention Software -ohjelmistoista Linuxille. Fail2Ban soveltuu muunmuassa brute force -hyökkäysten estämiseen. Bruteforce hyökkäyksissä palvelimelle (tai johonkin sen palveluun) koitetaan kirjautua kokeilemalla kaikkia mahdollisia merkkijonoja salasanana. **Fail2Banista onkin siis eniten hyötyä palveluissa, joihin tunnistaudutaan salasanapohjaisesti.**

Euroopan unionin osarahoittama

Esimerkiksi, jos hyökkääjä tietää että palvelimelle voidaan kirjautua ’admin’-käyttäjätunnuksella, voi hän kokeilla vaikkapa sanakirjahyökkäystä salasanan selvittämiseen. Sanakirjahyökkäyksessä kaikkia sanakirjatiedostosta löytyviä sanoja, mahdollisine numeroyhdisteineen, kokeillaan kunnes oikea yhdistelmä löytyy. Heikot salasanat kuten ’salasana123’ ja ’qwerty420’ on murrettavissa muutamassa sekunnissa, käytännössä millä tahansa laitteistolla.

Fail2Ban toimii skannaamalla konfiguroitujen palveluiden log-tiedostoja. Jos log-tiedostoon ilmestyy määritelty määrä epäonnistuneita kirjautumisyrityksiä, voidaan kyseinen IP-osoite estää määräajaksi tai kokonaan. Fail2Banista löytyy melko monipuolisesti konfigurointimahdollisuuksia, joilla turvallisuutta voidaan lisätä.

Tavallisesti Fail2Ban asettaa estot suoraan iptables:iin. Tähän toimintaan ei ole pakko koskea, mutta jos käytössä on firewalld-palomuuri, voidaan Fail2Ban konfiguroida asettamaan estot siihen. Vastaavasti saman toiminnan voi ohjata suoraan nftablesiin, jos tämä on palvelimella käytössä.

Seuraava ohjeistus on tehty Debian 12 -jakelulla. Jakelusta riippuen konfiguraatioissa saattaa olla eroavaisuuksia, jotka tulee ottaa huomioon konfiguraatioita tehdessä.

Fail2Ban ja sen riippuvuudet saadaan asennettua jakelun pakettirepositoriosta:

```
$ apt update && apt upgrade -y
$ apt install fail2ban
```

Kun tarkistamme onko `fail2ban.service` automaattisesti aktiivisena, huomaamme että käynnistys on epäonnistunut:

```
$ systemctl status fail2ban
× fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset:
enabled)
     Active: failed (Result: exit-code) since Mon 2025-05-26 14:57:20 EEST;
20min ago
   Duration: 72ms
       Docs: man:fail2ban(1)
    Process: 102151 ExecStart=/usr/bin/fail2ban-server -xf start (code=exited,
status=255/EXCEPTION)
   Main PID: 102151 (code=exited, status=255/EXCEPTION)
        CPU: 73ms
```

Palvelun logit tarkastamalla saamme hieman lisätietoa ongelmasta:

```
$ journalctl -u fail2ban
May 26 14:51:59 aimvmdebian120 systemd[1]: Started fail2ban.service - Fail2Ban Service.
May 26 14:51:59 aimvmdebian120 fail2ban-server[102057]: 2025-05-26 14:51:59,796 fail2ban.configreader   [102057]: WARNING 'allowipv6' not defined in 'Definition'. Using default one: 'auto'
May 26 14:51:59 aimvmdebian120 fail2ban-server[102057]: 2025-05-26 14:51:59,807 fail2ban    [102057]: ERROR   Failed during configuration: Have not found any log file for sshd jail
May 26 14:51:59 aimvmdebian120 fail2ban-server[102057]: 2025-05-26 14:51:59,812 fail2ban    [102057]: ERROR   Async configuration of server failed
May 26 14:51:59 aimvmdebian120 systemd[1]: fail2ban.service: Main process exited, code=exited, status=255/EXCEPTION
May 26 14:51:59 aimvmdebian120 systemd[1]: fail2ban.service: Failed with result 'exit-code'.
```

Debian 12 -jakelussa SSHD on asetettu oletuksena palveluksi jota Fail2Ban valvoo. Tässä ei ole kuitenkaan otettu huomioon sitä, että Debian 12 käyttää oletuksena erillisten `syslog` -tiedostojen sijaan `journald` -logausta. **Tämä ongelma riippuu täysin jakelusta, eikä se välttämättä ilmene lainkaan. Jos Fail2Ban toimii kuten pitää, ei backend -konfiguraatioon tarvitse tehdä muutoksia.**

Jotta saamme Fail2Banin käynnistymään oikein, tulee meidän muuttaa Fail2Banin jail -konfiguraatio vastaamaan tätä.

Fail2Banin konfiguraatiotiedostot löytyvät `/etc/fail2ban` -hakemistosta. Kyseisestä hakemistosta löytyvät valmiiksi seuraavat tiedostot ja alahakemistot:

```
$ ls -lh /etc/fail2ban/
total 64K
drwxr-xr-x 2 root root 4.0K May 26 14:51 action.d
-rw-r--r-- 1 root root 3.0K Nov  9  2022 fail2ban.conf
drwxr-xr-x 2 root root 4.0K Apr 21  2023 fail2ban.d
drwxr-xr-x 3 root root 4.0K May 26 14:51 filter.d
-rw-r--r-- 1 root root  26K Nov  9  2022 jail.conf
drwxr-xr-x 2 root root 4.0K May 26 15:30 jail.d
-rw-r--r-- 1 root root  645 Nov  9  2022 paths-arch.conf
-rw-r--r-- 1 root root 2.7K Nov  9  2022 paths-common.conf
-rw-r--r-- 1 root root  627 Nov  9  2022 paths-debian.conf
-rw-r--r-- 1 root root  738 Nov  9  2022 paths-opensuse.conf
```

Näistä tällä hetkellä tärkeimmät ovat ylemmässä listauksessa korostettuna. `.conf` päätteiset tiedostot ovat Fail2Banin automaattisesti luomia. Niitä itsessään ei kannata

Euroopan unionin osarahoittama

muuttaa, sillä mahdolliset muutokset ylikirjoittuvat seuraavan kerran kun Fail2Ban päivitetään.

Sen sijaan, esimerkiksi jail -konfiguraatioita varten tulee luoda joko oma tiedostonsa: `jail.local`. Toinen vaihtoehto on asettaa tiedostot jokaiselle palvelulle erikseen `jail.d` -hakemistoon: `jail.d/palvelun_nimi.conf`.

Tiedostot luetaan seuraavassa järjestyksessä, ja viimeisimpänä luettu palvelukohtainen konfiguraatio otetaan käyttöön:

1. `jail.conf`
2. `jail.d/*.conf` (aakkosjärjestyksessä)
3. `jail.local`
4. `jail.d/*.local` (aakkosjärjestyksessä)

Debian 12 -jakelussa `jail.d` -hakemistossa on automaattisesti seuraava tiedosto, jota ei tule muokata, sillä sekin todennäköisesti ylikirjoitetaan päivityksen yhteydessä.

```
$ cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
```

Jotta saamme SSHD-palvelun Fail2Banin valvottavaksi, samalla korjaten backend-virheen, luomme seuraavan tiedoston:

```
$ touch /etc/fail2ban/jail.local
```

Tiedostoon lisätään alustavasti seuraava sisältö:

```
1 [DEFAULT]
2 backend = systemd
3
4 [sshd]
5 enabled = true
```

Nyt Fail2Banin tulisi uudelleenkäynnistettäessä aktivoitua onnistuneesti:

```
$ systemctl restart fail2ban
$ systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset:
enabled)
     Active: active (running) since Mon 2025-05-26 16:30:06 EEST; 5s ago
```

Tiedoston `[DEFAULT]` -osion alle voimme laittaa yleisiä konfiguraatioita jotka pätevät kaikkiin palveluihin. Palvelukohtaisesti näitä voidaan toki muuttaa erilaisiksi.

`[sshd]` -osion alle tulevat kaikki SSHD-konfiguraatiot. `enabled = true` -rivi ottaa kyseisen palvelun jailit käyttöön Fail2Banissa.

Voimme halutessamme hienosäätää Fail2Banin toimintaa asettamalla tiedostoon esimerkiksi seuraavat konfiguraatiot:

```
1 [DEFAULT]
2 backend = systemd
3
4 # Asettaa estoajaksi yhden tunnin
5 # (oletuksena aika annetaan sekunneissa, mutta päätteillä
6 # voidaan tehdä konfiguraatio helpommaksi (3600 = 60m = 1h)
7 bantime = 1h
8
9 # Jos käytössä on firewalld-palomuuri, voidaan näillä konfiguraatioilla
10 # asettaa fail2ban tekemään estot suoraan firewalld:hen
11 banaction = firewallcmd-rich-rules[actiontype=]
12 banaction_allports = firewallcmd-rich-rules[actiontype=]
13
14 # Jos IP-osoite on estetty aiemmin, kasvatetaan estoaikaa jokaisen uuden
eston
15 # kanssa. Oletuksena: banTime * 1, 2, 4, 8, 16, 32...
16 bantime.increment = true
17
18 # Kirjautumisyritysten sallittu määrä ennen estoa
19 maxretry = 6
20
21
22
23 [sshd]
24 # Ottaa käyttöön palvelukohtaisen toimintamallin (oletus: normal)
25 mode = normal
26
27 # Jail käytössä tälle palvelulle
28 enabled = true
29
30 # Valitaan portti jota palvelu käyttää (oletus: ssh, palvelimella käytössä:
22022)
31 port = 22022
```


Euroopan unionin osarahoittama

Kun olemme tyytyväisiä konfiguraatioomme, voimme varmistaa sen toiminnan ’`dry-run`’ -tyylisesti seuraavilla komennoilla:

```
$ fail2ban-server --test
2025-05-27 11:44:21,255 fail2ban.configreader   [1313]: WARNING 'allowipv6' not defined in 'Definition'. Using default one: 'auto'
OK: configuration test is successful


$ fail2ban-client --test
2025-05-27 11:44:29,919 fail2ban.configreader   [1314]: WARNING 'allowipv6' not defined in 'Definition'. Using default one: 'auto'
OK: configuration test is successful
```

Esimerkkitapauksessa varoitukset eivät aiheuta syytä huoleen, ne vain kertovat että ’`allowipv6`’ konfiguraatiota ei ole asetettu erikseen, joten sen suhteen käännytään oletuskonfiguraatioon.

Kun olemme luoneet haluamamme konfiguraation ja varmistaneet sen toiminnan, tulee Fail2Ban käynnistää uudelleen ja varmistaa että se on aktiivisena:

```
$ systemctl restart fail2ban
$ systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-05-26 16:55:09 EEST; 5s ago
       Docs: man:fail2ban(1)
   Main PID: 102591 (fail2ban-server)
      Tasks: 5 (limit: 9421)
     Memory: 14.4M
        CPU: 100ms
     CGroup: /system.slice/fail2ban.service
             └─102591 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
```

Fail2Banin asennuksen mukana pitäisi tulla `fail2ban-client` -terminaaliohjelma, josta Fail2Bania voidaan ohjailla. Käytössä olevat jailit voidaan tarkistaa seuraavalla komennolla:

```
$ fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
```

Lisäksi estetyt IP-osoitteet voidaan tarkistaa seuraavasti:

```
$ fail2ban-client banned
[{'sshd': []}]
```

Seuraava esimerkki pätee jos SSH-yhteys on sallittu suoraan salasanalla. Avainparitunnistautuminen toimii oleellisesti toisella tavalla, joten se ei päde tähän tapaukseen.

Fail2Banin toimintaa voi testata helpoiten kun palvelimelle on suora fyysinen näppäimistöyhteys. Sen jälkeen uusi SSH-yhteys luodaan toisesta laitteesta, mutta niin että salasanaa ei syötetä ollenkaan. Tätä voidaan toistaa niin kauan, että SSH-yhteyden luominen epäonnistuu kokonaan. **Jos firewalld ei ole fail2banin käytössä, esto tulee rikkomaan kaikki SSH-yhteydet palvelimelle. Jos ainut vaihtoehto testaukseen on SSH-yhteyden yli, kannattaa bantime väliaikaisesti konfiguroida hyvin lyhyeksi!**

```
$ ssh -p 22022 aim@192.168.122.197
aim@192.168.122.197's password:
Permission denied, please try again.
aim@192.168.122.197's password:
Permission denied, please try again.
aim@192.168.122.197's password:
...
$ ssh -p 22022 aim@192.168.122.197
ssh: connect to host 192.168.122.197 port 22022: Connection refused
```

Nyt estetyissä IP-osoitteessa näkyy laitteemme IP-osoite:

```
$ fail2ban-client banned
[{'sshd': ['192.168.122.1']}]
```

Emme kuitenkaan halua estää itseämme palvelimelta, joten voimme poistaa eston seuraavalla komennolla:

```
$ fail2ban-client unban 192.168.122.1
1
```


Euroopan unionin osarahoittama

Wordpress-Fail2Ban -integraatio / WP fail2ban

Koska palvelimen SSH-yhteydet on konfiguroitu toimimaan avainparitunnistautumisen perusteella, ei fail2banista ole sen suhteen juurikaan hyötyä. Jos käyttäjän yksityinen avain ei sovi palvelimelle tallennettuun julkiseen osaan, ei kirjautuminen ole mahdollista vaikka raakaa voimaa käytettäisiin loputtomiin.

Fail2bania voidaan kuitenkin hyödyntää suojaamaan web-pohjaisia käyttöliittymiä. Esimerkiksi Wordpressin Dashboard, josta ylläpitotöitä yleisesti tehdään, on erittäin haavoittuvassa asemassa. Brute Force hyökkäyksillä saatetaan pyrkiä murtautumaan massana tuhansille erilaisille palvelimille, joista yksi saattaa olla meidän.

Hyökkäyksiltä suojautuminen on kuitenkin hieman hankalaa, sillä Wordpress toimii jotenkuten suljetusti kirjautumisyritysten logaamisen suhteen. Tähän käyttöön on olemassa erilaisia plugin-ratkaisuja, joista moni toimii täysin Wordpressin oman moottorin sisällä. On kuitenkin olemassa plugineja, joiden avulla Wordpressin toiminta voidaan integroida palvelimelta löytyvään fail2ban -ohjelmistoon.

Yksi näistä plugineista on Charles Leckliderin luoma **`WP fail2ban`**. Kyseinen plugin on saatavilla Wordpress.org sivustolta: https://wordpress.org/plugins/wp-fail2ban/ Lähdekoodi on saatavilla Githubissa: https://github.com/wp-fail2ban/wp-fail2ban.

WP fail2ban - Asentaminen

Plugineja voidaan yleisesti asentaa suoraan Wordpressin Dashboardista. Koska palvelimemme on paikallinen kehityspalvelin, eikä sille ole asetettu esimerkiksi domain-osoitetta, joudumme asentamaan sen suoraan palvelimen tiedostoihin.

Voimme ladata wordpress.org -sivustolta pluginin sisältävän .zip -tiedoston esimerkiksi omalle työasemallemme. .zip -tiedosto sisältää suurehkon kansion, joka puretaan .zip -tiedostosta hyödylliseen sijaintiin. Tämän jälkeen voimme siirtää kansion palvelimelle vaikkapa SFTP-yhteyttä hyödyntäen. Aiemmassa SFTP-ohjeistuksessa tehtiin käyttäjä, jonka oma hakemisto on linkitetty Wordpressin wp-content sisältöhakemistoon. Tämä hakemisto sisältää plugins -alahakemiston, johon palvelimen pluginit tallennetaan.

SFTP-yhteyden yli voidaan siirtää tiedostoja seuraavalla tavalla (Huomaa `-R` -flagi komennossa, jolla voidaan lähettää kokonainen hakemisto rekursiivisesti kaikkine sisältöineen):

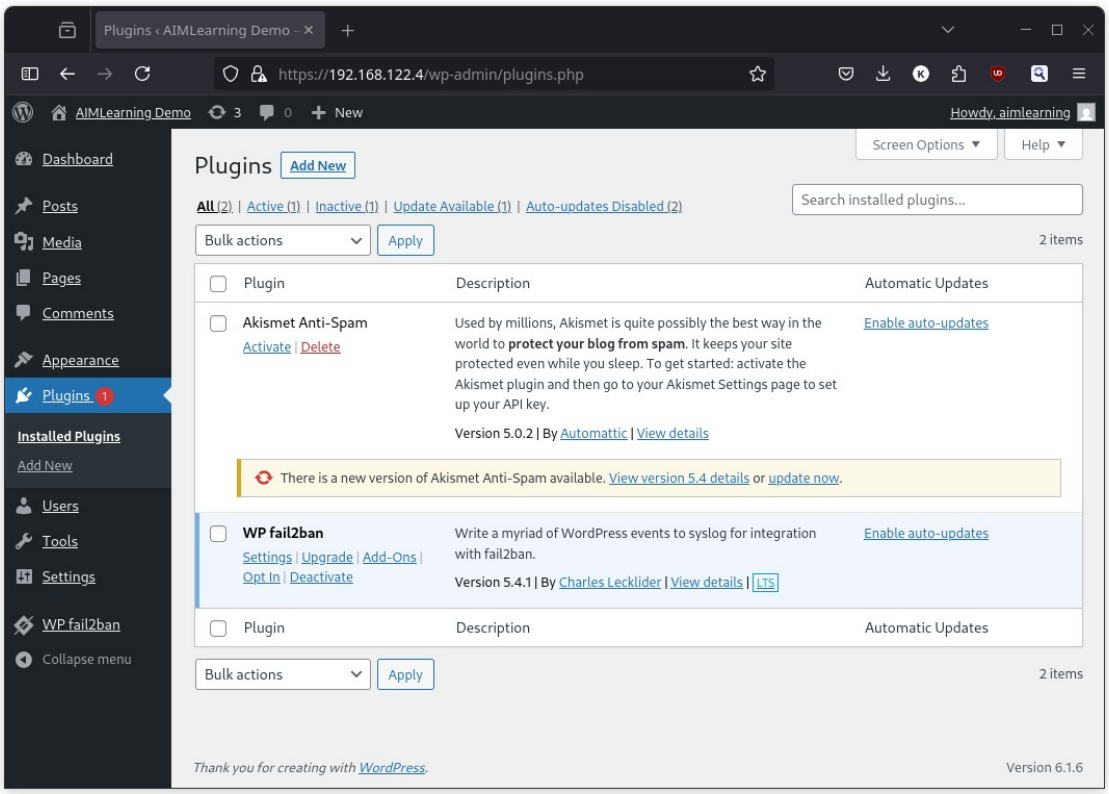
```
sftp> put -R /home/username/Downloads/wp-fail2ban /aim-sftp/wp-content/plugins
Uploading /home/username/Downloads/wp-fail2ban/ to
/aim-sftp/wp-content/plugins/./wp-fail2ban
Entering /home/username/Downloads/wp-fail2ban/
Entering /home/username/Downloads/wp-fail2ban/admin
admin.php
100%   13KB  10.9MB/s   00:00
Entering /home/username/Downloads/wp-fail2ban/admin/config
block.php
100% 5138     6.8MB/s   00:00
logging.php
100% 6050    10.0MB/s   00:00
plugins.php
100% 4708     8.7MB/s   00:00
remote-ips.php
100% 3628     5.0MB/s   00:00
syslog.php
100% 8681    13.4MB/s   00:00
config.php
100% 4269     8.6MB/s   00:00
...
Entering /home/username/Downloads/wp-fail2ban/vendor/wp-fail2ban/lib-badges/src
Entering /home/username/Downloads/wp-fail2ban/vendor/wp-fail2ban/lib-badges/
src/Badge
AbstractBadge.php
100% 2966    19.6MB/s   00:00
CanonicalBadge.php
100% 1237    10.2MB/s   00:00
LTSBadge.php
100%  748     5.9MB/s   00:00
NonCanonicalBadge.php
100% 1264    10.5MB/s   00:00
BadgeManager.php
100% 6188    41.8MB/s   00:00
wp-fail2ban.php
100% 1903    14.5MB/s   00:00
sftp>
```


Euroopan unionin osarahoittama

Tiedostoja on melkoisesti, mutta sisäverkon yli siirrettäessä tämä tapahtuu suhteellisen nopeasti. Koska tiedostot lisättiin `aim-sftp` -käyttäjänä, tulee näiden oikeudet vielä muuttaa. Nopeimmin tämä voidaan tehdä SSH-yhteyden yli:

```
$ chown www-data:www-data -R /var/sftp/files/aim-sftp/wp-content/plugins/wp-fail2ban/
```

Kun plugin-hakemisto ja tämän omistajuus ovat kunnossa, voidaan Wordpressin Dashboardista tarkistaa onko plugin aktivoitavissa Plugins-välilehdeltä.



Kuva 28: Wordpress Dashboard:in Plugins-välilehti

Kun plugin on Wordpressin käytössä ja aktivoituna, joudumme asettamaan palvelimen puolella muutaman erityisen konfiguraation Fail2Ban -integraatiota varten. Jakelusta riippuen, tämä tapahtuu hieman eri tavalla (https://docs.wp-fail2ban.com/en/5.4/configuration/fail2ban.html).

Esimerkkikäytössä oleva Debian 12 -jakelu on siirtynyt käytännössä täysin journald-logaukseen. Tästä johtuen joudumme tekemään lisäyksiä `/usr/share/wordpress/wp-config.php` -tiedostoon. Näillä lisäyksillä taataan `journald` -logauksen yhteensopivuus pluginin kanssa.

Tiedostoon lisätään korostettuna näkyvät rivit:

```
...
71 define( 'FORCE_SSL_ADMIN', true );
72
73 // Make sure we're not using the short ("wp") tag
74 define('WP_FAIL2BAN_SYSLOG_SHORT_TAG', false);
75
76 // Don't include the HTTP host in the tag
77 define('WP_FAIL2BAN_SYSLOG_TAG_HOST', false);
78
79 require_once(ABSPATH . 'wp-settings.php');
80 ?>
```

Joudumme myös lisäämään Fail2Banin tiedostoihin pluginissa määriteltyjä filter-konfiguraatioita. Nämä voidaan käytännössä vain kopioida pluginin tiedostoista Fail2Banin hakemistoon:

```
$ cp /var/sftp/files/aim-sftp/wp-content/plugins/wp-fail2ban/filters.d/
wordpress-hard.conf
/var/sftp/files/aim-sftp/wp-content/plugins/wp-fail2ban/filters.d/wordpress-
soft.conf /etc/fail2ban/filter.d/
```

Kun filter-tiedostot on paikallaan, voimme lopuksi konfiguroida jailit ennestään tekemäämme `/etc/fail2ban/jail.local` -tiedostoon. Jaileja tehdään kaksi, joista toinen automaattisesti estää erityisesti epäilyttäväksi määritellyn toiminnan. Toinen puolestaan toimii ennemmin sisäänkirjautumisen yhteydessä, ollen näin hieman ”vapaampi”:

Euroopan unionin osarahoittama

```
...
25 [wordpress-hard]
26 enabled = true
27 filter = wordpress-hard
28 backend = systemd
29 journalmatch = SYSLOG_IDENTIFIER=wordpress
30 maxretry = 1
31 port = http,https
32
33 [wordpress-soft]
34 enabled = true
35 filter = wordpress-soft
36 backend = systemd
37 journalmatch = SYSLOG_IDENTIFIER=wordpress
38 maxretry = 3
39 port = http,https
```

Konfiguraatiot voidaan vielä varmistaa:

```
$ fail2ban-client --test
2025-06-10 16:48:12,893 fail2ban.configreader   [6419]: WARNING 'allowipv6' not defined in 'Definition'. Using default one: 'auto'
OK: configuration test is successful
```

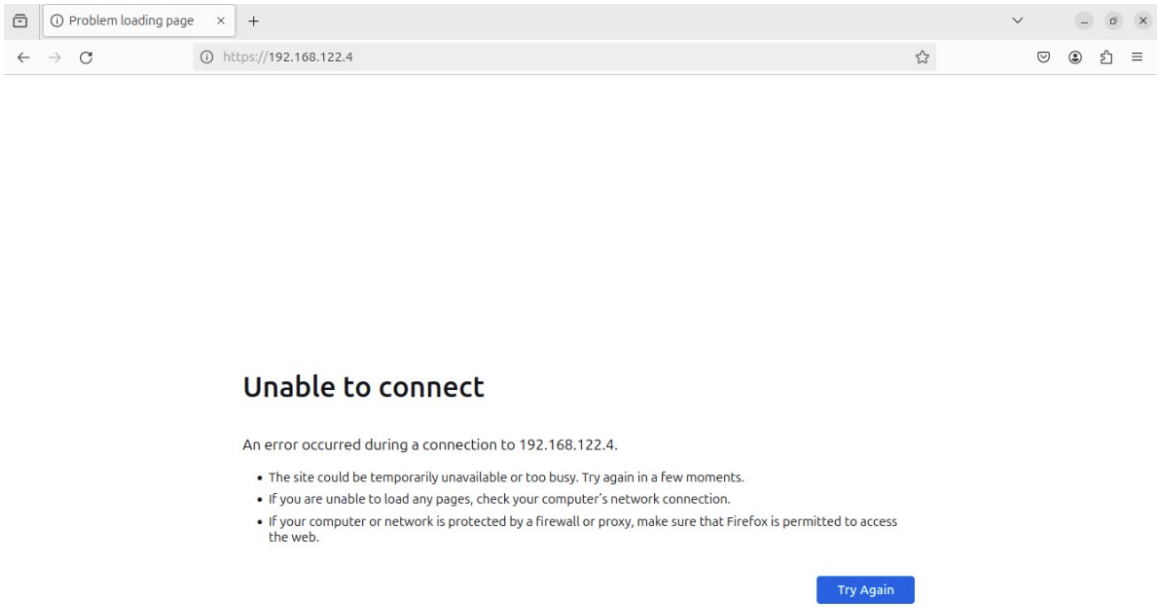
Lopuksi fail2ban ja apache2 voidaan käynnistää uudelleen, jolloin Fail2Banin pitäisi asettua toimintaan:

```
$ fail2ban-client reload
2025-06-10 16:47:58,687 fail2ban.configreader   [6410]: WARNING 'allowipv6' not defined in 'Definition'. Using default one: 'auto'
OK
$ systemctl reload apache2
```

Käytännössä jos Wordpressin Dashboardin kirjautumisikkunassa salasana syötetään tarpeeksi monta kertaa väärin, asettaa Fail2Ban palomuuritasolla IP-kohtaisen eston.

```
$ fail2ban-client banned
[{'sshd': []}, {'wordpress-hard': []}, {'wordpress-soft': ['192.168.122.73']}]
```

Tämän jälkeen sivustolle ei pääse yhdistämään lainkaan, kunnes esto-aika on mennyt umpeen:



Kuva 29: Firefox ei saa yhteyttä sivustoon, kun Fail2Ban estää IP-osoitteen

Lähteet:

https://www.linux.fi/wiki/Fail2ban

https://wiki.archlinux.org/title/Fail2ban

https://github.com/fail2ban/fail2ban/issues/1372

man jail.conf

fail2ban konfigurointitiedostot

https://www.atlantic.net/vps-hosting/how-to-install-fail2ban-with-firewalld-on-rocky-linux-8/

https://unix.stackexchange.com/questions/268357/how-to-configure-fail2ban-with-systemd-journal

Euroopan unionin osarahoittama

Tietokannan web-käyttöliittymä

Wordpressillä verkkosivustoa kehittäessä voi olla hyödyllistä olla perillä siitä mitä sivuston tietokannassa tapahtuu. Tähän voi yksinkertaisesti käyttää tietokannan omaa komentoriviohjelmistoa, mutta tämä vaatii jonkinlaista osaamista SQL-kielestä. Lisäksi pelkkään tietojen tarkasteluun tämä saattaa olla hieman kankeaa.

```
$ mariadb -u wordpress -p
...
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| wordpress          |
+--------------------+
2 rows in set (0.001 sec)

MariaDB [(none)]> USE wordpress;
...
Database changed

MariaDB [wordpress]> SHOW TABLES;
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
...
```

Yleensä Wordpress-sivustojen yhteydestä löytyy jonkinlainen verkkosivustopohjainen käyttöliittymä, josta samoja asioita voi tehdä visuaalisemmin. Yleisesti hostauspalveluissa on valmiiksi asennettuna phpMyAdmin -niminen työkalu, joka toimii MySQL- ja MariaDB-tietokantojen kanssa.

Adminer

Toinen varteenotettava, ja huomattavasti kevyempi, vaihtoehto on Adminer. Adminer on phpMyAdminiin verrattuna vastaava työkalu, jolla voidaan tehdä samoja asioita. Adminer on saatavilla useassa jakelussa pakettirepositoriosta. Viimeisimmät versiot päivittyvät näihin kuitenkin suhteellisen hitaasti, joten saattaa olla tarpeen ladata Adminer tämän virallisesta GitHub-repositoriosta: https://github.com/vrana/adminer.

Käytännössä Adminer on vain yksittäinen .php -tiedosto, joka voidaan asettaa omaksi sivuksi web-palvelimelle. Apache2:n tapauksessa tälle voidaan yksinkertaisesti tehdä oma hakemisto ja Virtual Host -joiden pohjalta Adminer toimii. Jos käytössä on domain, voidaan Adminer asettaa omaksi alidomainikseen.

Tapauksessamme web-palvelin toimii puhtaasti IP-osoitteen yli. Tämän vuoksi alidomain ei onnistu, mutta voimme asettaa Adminerin toimimaan eri porttiin kuin Wordpress-sivuston. Tästä on myös turvallisuusmielessä hyötyä, sillä voimme rajoittaa kyseisen portin saatavuutta niin, että kuka tahansa internetistä ei pääse tietokantakäyttöliittymään käsiksi.

Asennus

Adminerin .php -tiedoston lataamista varten voimme hyödyntää `wget` -pakettia. Jos palvelimella ei ole `wget` -pakettia, voidaan tämä asentaa pakettirepositoriosta:

```
$ apt install wget
```

`Wget`:llä lataaminen on hyvin yksinkertaista. Käytännössä tätä varten siirrytään hakemistoon johon tiedosto ladataan, jonka jälkeen tiedosto ladataan `wget`:llä.

```
$ cd ~/Downloads/
$ wget https://github.com/vrana/adminer/releases/download/v5.3.0/adminer-5.3.0-en.php
```

Kun .php -tiedosto on ladattu, luodaan web-palvelimelle hakemisto jossa Adminer-sivusto pyörii. .php -tiedosto voidaan kopioida luotuun hakemistoon.

```
$ mkdir /var/www/adminer
$ cp Downloads/adminer-5.3.0-en.php /var/www/adminer/
```

Apacheen tulee luoda oma Virtual Host Admineria varten. Tämä voidaan kopioida esimerkiksi olemassa olevasta `/etc/apache2/sites-available/wp-ssl.conf` -tiedostosta, muuttaen muutamia kohtia:

```
$ cp /etc/apache2/sites-available/wp-ssl.conf
/etc/apache2/sites-available/adminer.conf
```


Euroopan unionin osarahoittama

Koska meillä ei tällä hetkellä ole domainia, vaan toimimme pelkästään IP-osoitteen pohjalta, joudumme tekemään Adminer-sivuston saataville normaalista poikkeavaan porttiin. Muutamme tämän portin siis kopioidun tiedoston ensimmäiseltä riviltä. Lisäksi `DocumentRoot` tulee muuttaa vastaamaan Adminerin sijaintia:

```
1 <VirtualHost *:9001>
2         ServerName localhost
3         ServerAlias 192.168.122.4
4
5         ServerAdmin webmaster@localhost
6         DocumentRoot /var/www/adminer
7
8         #   SSL Engine Switch:
9         #   Enable/Disable SSL for this virtual host.
10        SSLEngine on
...
```

Dokumentin lopusta voidaan myös poistaa turhat Wordpressiin liittyvät konfiguraatiot:

```
...
47        Alias /wp-content /var/lib/wordpress/wp-content
48        <Directory /usr/share/wordpress>
49            Options FollowSymLinks
50            AllowOverride Limit Options FileInfo
51            DirectoryIndex index.php
52            Require all granted
53        </Directory>
54        <Directory /var/lib/wordpress/wp-content>
55            Options FollowSymLinks
56            Require all granted
57        </Directory>
...
```

Kokonaisuudessaan konfiguraatio näyttää seuraavalta (ilman isoja kommenttilohkoja):

```
<VirtualHost *:9001>
        ServerName localhost
        ServerAlias 192.168.122.4

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/adminer

        <FilesMatch "\.(?:cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
                SSLOptions +StdEnvVars
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Sivusto tulee myös ottaa käyttöön Apache2:ssa:

```
$ a2ensite adminer
Enabling site adminer.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Koska käytämme tavallisesta poikkeavaa porttia, tulee tämä konfiguroida Apache2:een. Tämä voidaan tehdä tiedostossa `/etc/apache/ports.conf`, johon lisätään kaksi `Listen 9001` -riviä. Nämä lisätään SSL ja TLS-moduulien alle, jotta HTTPS on varmasti käytössä sivustolla.

```
 1 # If you just change the port or add more ports here, you will likely also
 2 # have to change the VirtualHost statement in
 3 # /etc/apache2/sites-enabled/000-default.conf
 4
 5 Listen 80
 6
 7 <IfModule ssl_module>
 8         Listen 443
 9         Listen 9001
10 </IfModule>
11
12 <IfModule mod_gnutls.c>
13         Listen 443
14         Listen 9001
15 </IfModule>
```

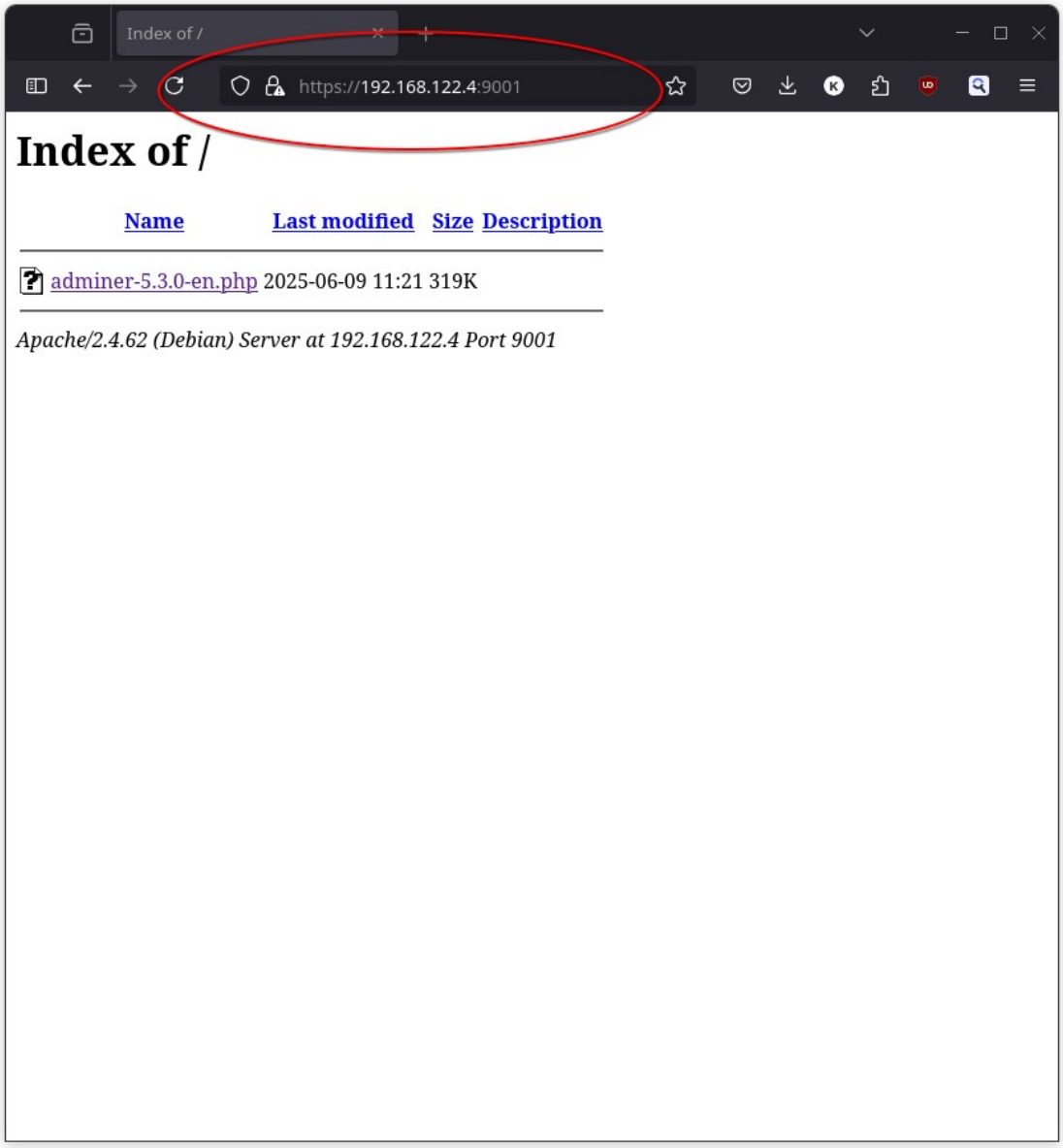
Nyt voimme ladata Apache2 -palvelun uudelleen, jotta kaikki konfiguraatiot tulevat Apache2:n käyttöön:

```
$ systemctl reload apache2
```

Koska asetimme sivuston toimimaan eri portissa kuin HTTPS-sivustot normaalisti, joudumme sallimaan kyseiseen porttiin yhdistämisen palomuurissa. Tässä tapauksessa yhdistäminen sallitaan ainoastaan omasta lähiverkosta.

```
$ firewall-cmd --zone=aimserver --add-rich-rule='rule family="ipv4" source
address="192.168.122.0/24" port port="9001" protocol="tcp" accept' --permanent
success
```

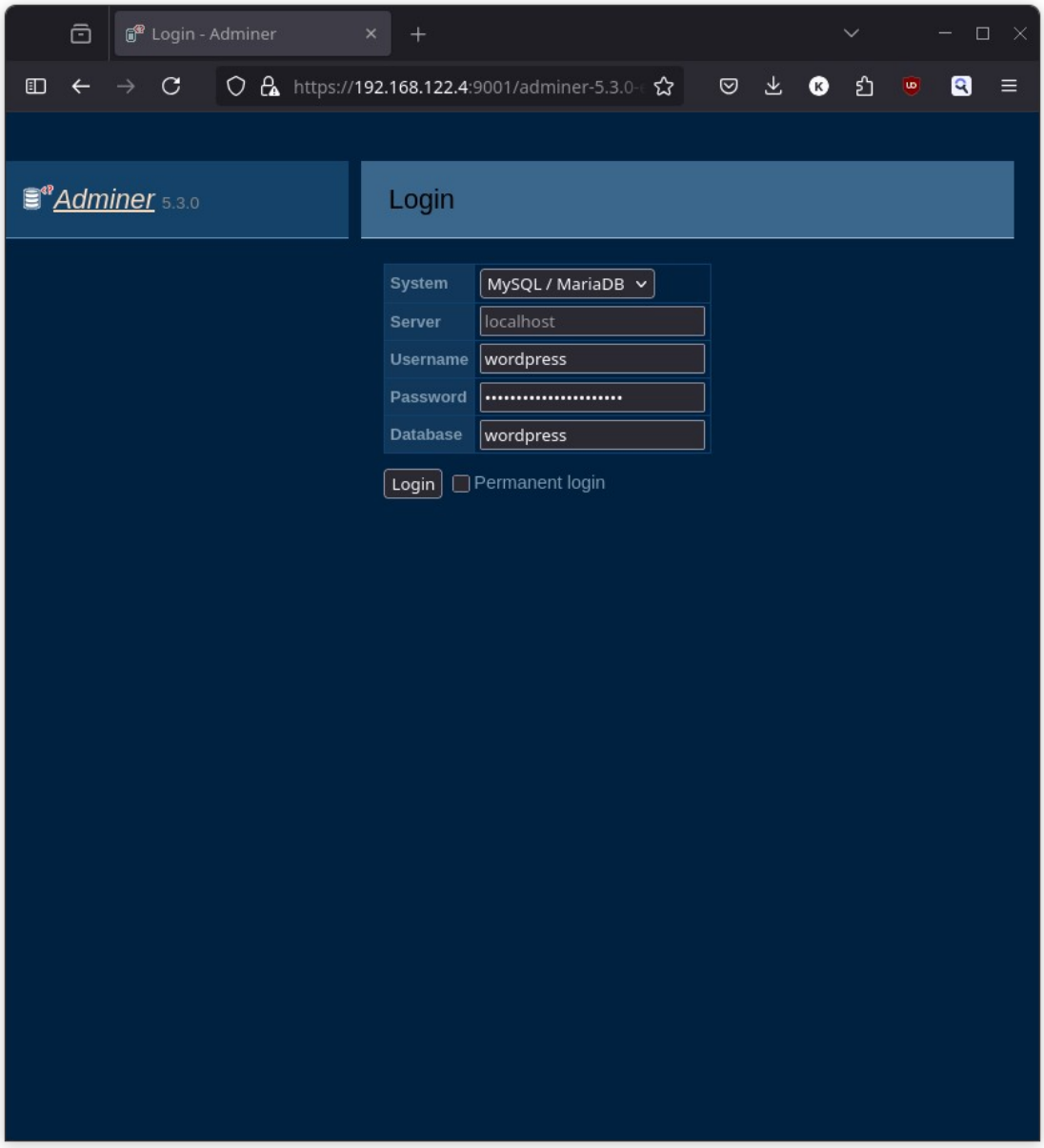
Tässä vaiheessa yhdistämistä web-käyttöliittymään on hyvä kokeilla. Tämä onnistuu vaikkapa oman työaseman selaimella:



Kuva 30: Adminer-käyttöliittymän IP-osoite ja portti

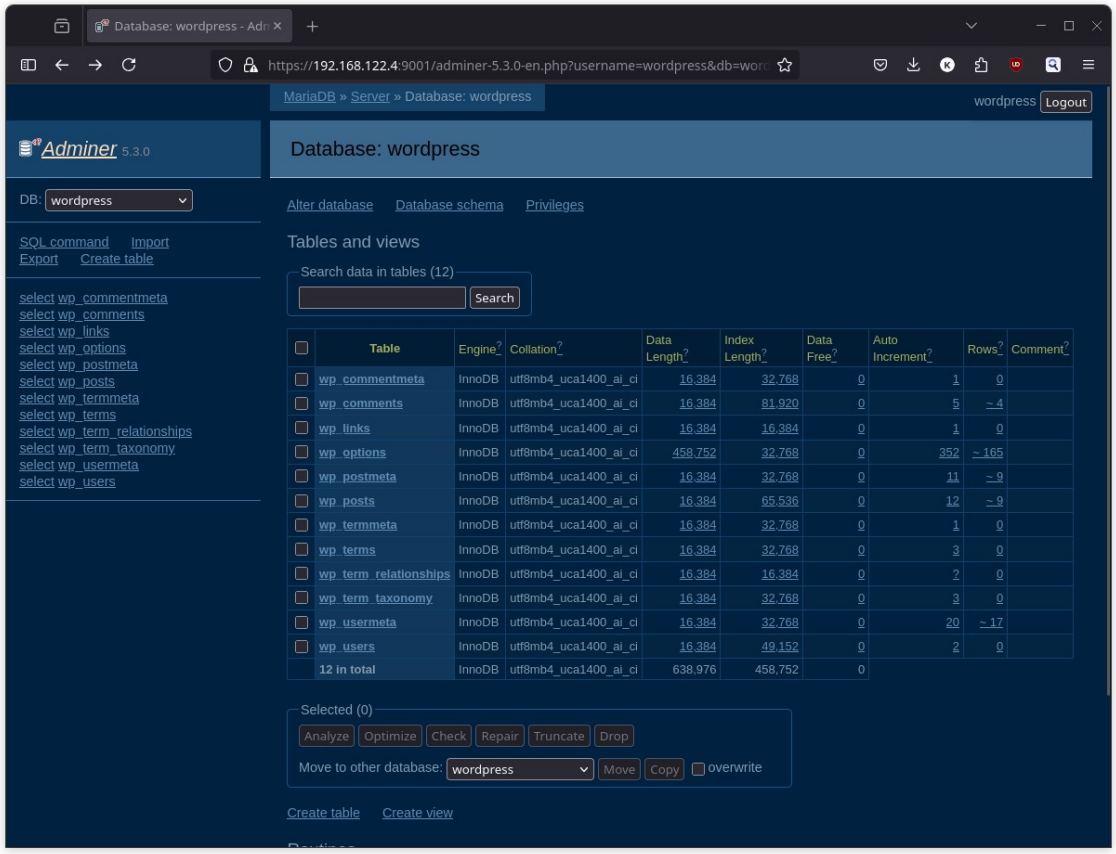
Euroopan unionin osarahoittama

Klikkaamalla .php -tiedoston nimeä, pääsemme itse käyttöliittymään. Käyttöliittymästä pääsee kirjautumaan tietokantaan käyttämällä **Wordpressin tietokannan** käyttäjätunnuksia:



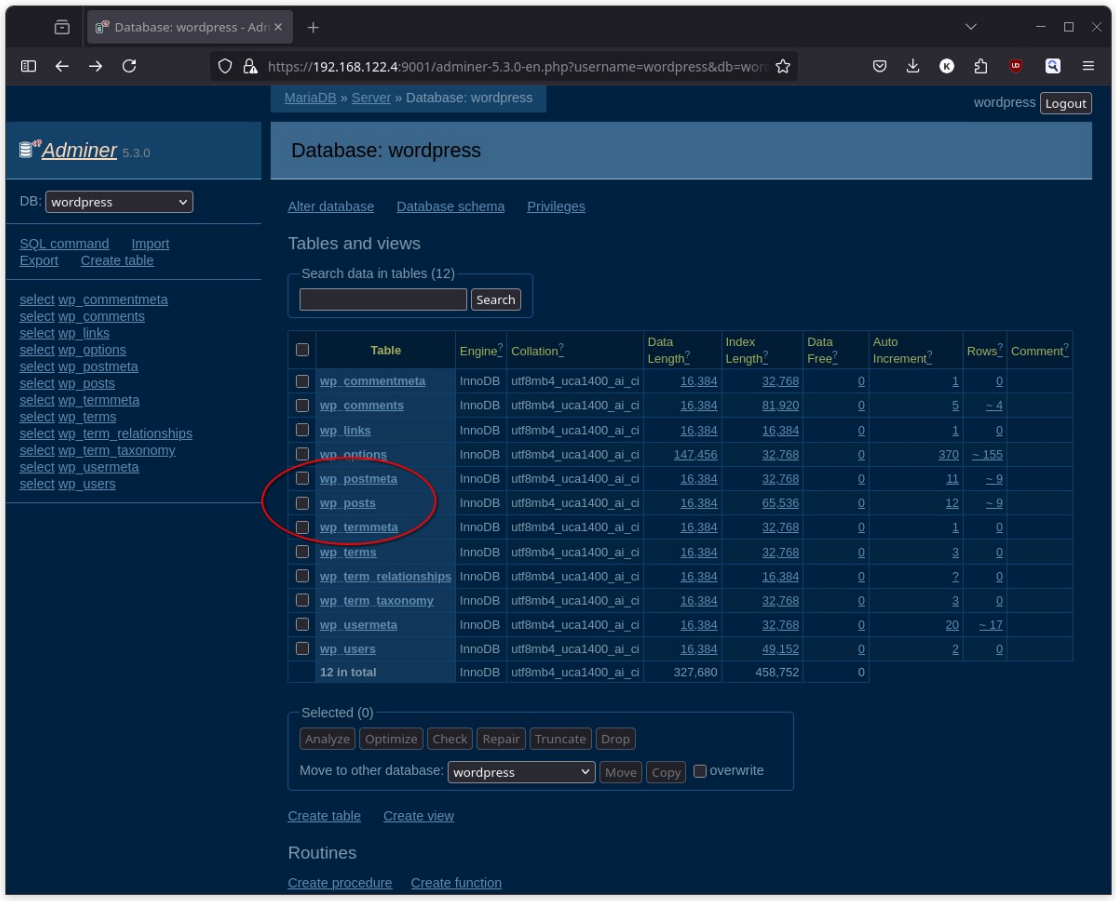
Kuva 31: Adminer-käyttöliittymän kirjautumisnäkymä

Adminerin varsinainen käyttöliittymä näyttää seuraavalta:



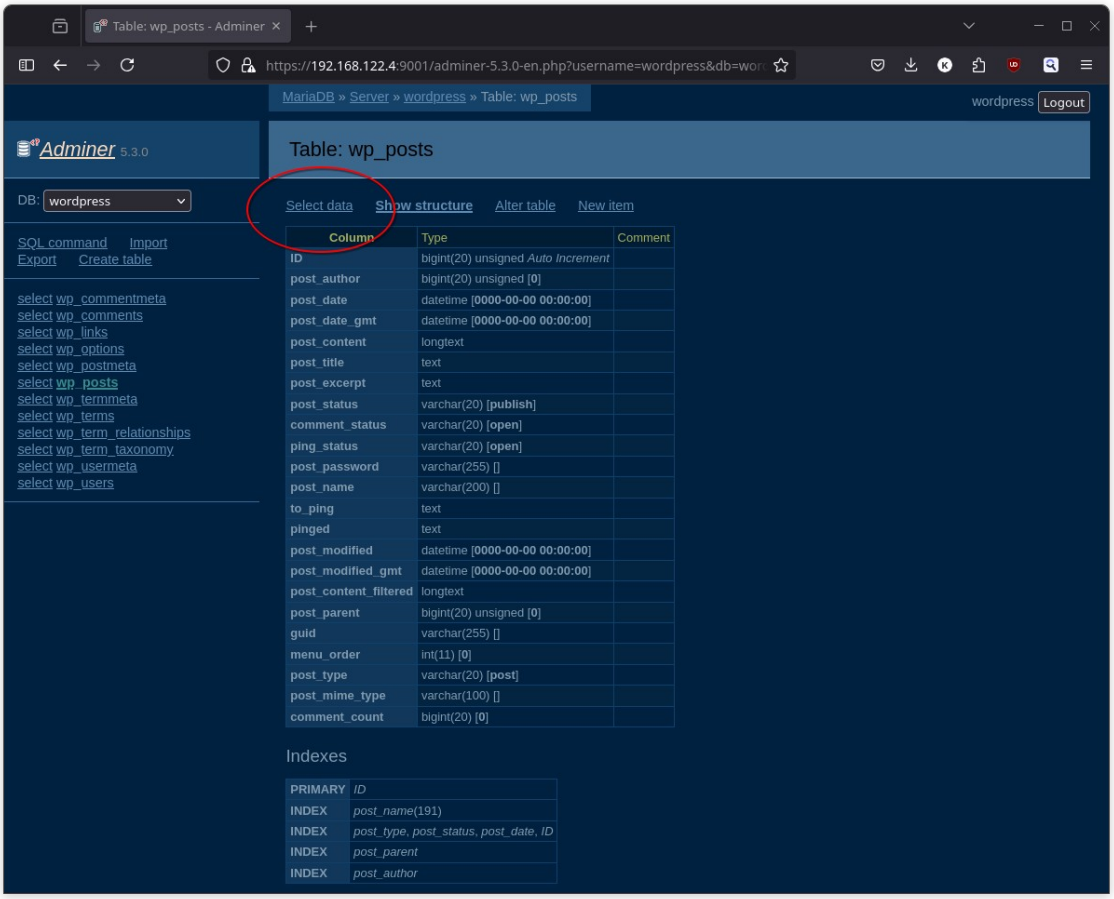
Kuva 32: Tietokannan taulut Adminer-käyttöliittymässä

Käyttöliittymä näyttää tällä hetkellä kaikki wordpress -tietokannan sisältämät taulut. Jos haluamme vaikkapa katsastaa sivustolle tehdyt postaukset, voimme klikata `wp_posts` -linkkiä:



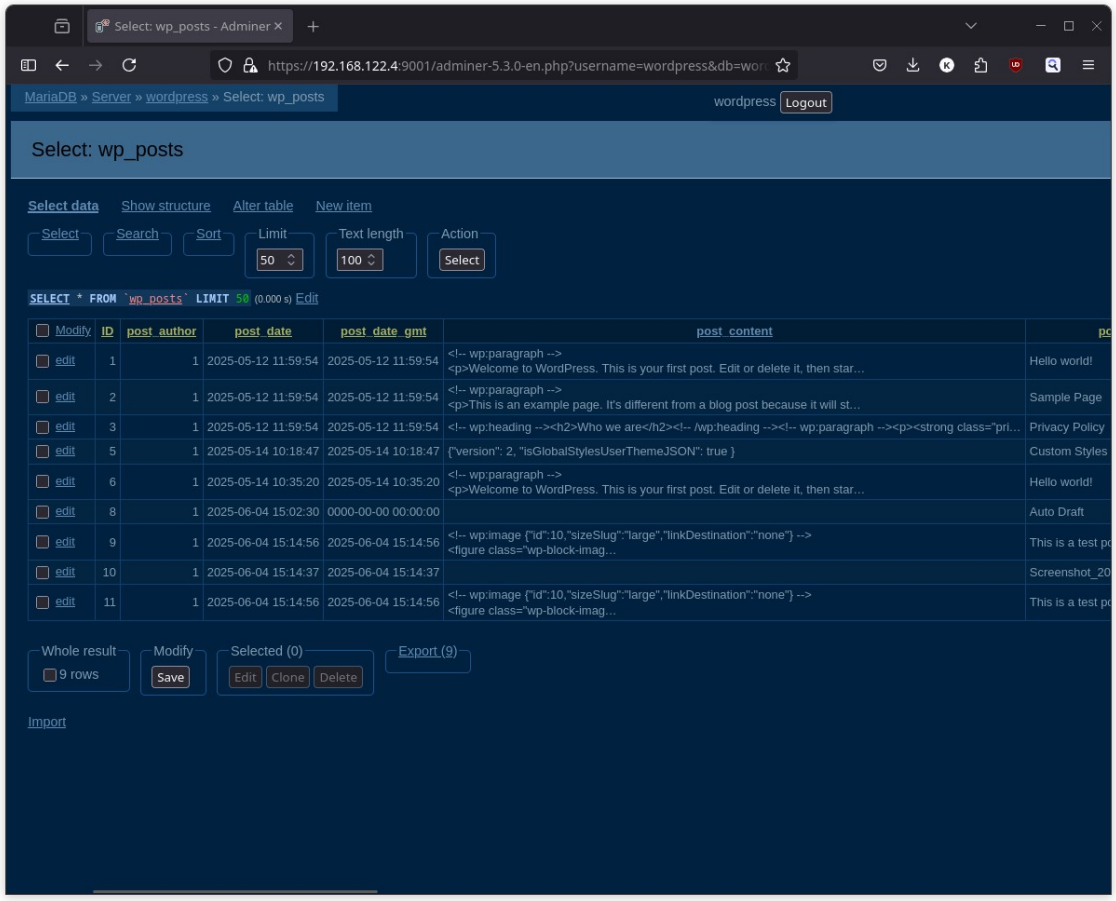
Kuva 33: wp_posts -taulu korostettuna Adminer-käyttöliittymässä

Aukeavassa näkymässä näytetään taululle määritetyt tiedot. Klikkaamalla `Select data` -linkkiä pääsemme näkemään taulun varsinaisen sisällön:



Kuva 34: wp_posts -taulun varsinainen näkymä Adminer-käyttöliittymässä

Taulun riveissä on melko paljon tietoa, mutta voimme kuitenkin selkeästi nähdä mitä postaukset sisältävät. Lisäksi voimme nähdä postauksen tekijän, ajankohdan ja paljon muuta tietoa. Sivustoa kehittäessä nämä saattavat olla hyödyllisiä, esimerkiksi outojen bugien metsästämisessä.



Kuva 35: wp_posts -taulun todellinen sisältö Adminer-käyttöliittymässä

Lähteet:

man firewalld.richlanguage

https://github.com/vrana/adminer

https://www.adminer.org/en/

Virustutka

Linux-maailmassa on hyvin tavallista, että työasemalla tai palvelimella ei ole lainkaan varsinaisia virustorjuntaohjelmistoja. Syitä tähän on monia; Linux-pohjaisia käyttöjärjestelmiä pidetään jo valmiiksi turvallisina, jonka lisäksi käyttäjiä on määrällisesti vähemmän Windowsiin verrattuna. Virustorjuntaohjelmistoja on kyllä markkinoilla, mutta yleensä niiden käyttö on varsin räätälöityä.

Varsinainen turvallisuus kuitenkin syntyy monen tekijän summana, joista virustorjunta on yksi. Tarkemmin sanottuna tässä osiossa asennettava ClamAV ei ole täysimittainen virustorjuntaohjelmisto, vaan virustutka, jolloin sitä tulee myös käsitellä sen mukaisesti.

ClamAV

ClamAV on Cisco Talosin kehittämä avoimen lähdekoodin virustutka, joka toimii monella eri käyttöalustalla. Alunperin ClamAV suunniteltiin toimimaan sähköpostipalvelimilla, suodattaen sähköpostiliikenteestä mahdollisesti haitallista sisältöä. Vaikka toiminta-alue on nykyään pelkkää sähköpostiliikennettä laajempi, ei ClamAV:ta voida kutsua varsinaiseksi antivirus-ohjelmistoksi.

ClamAV skannaa tiedostoja sormenjälkien perusteella. Verkosta noudetaan tietokantoja, joiden sisältämiin haitallisten tiedostojen ja virusten sormenjälkiin skannattavaa tiedostoa verrataan. Jos tiedoston (tai tämän sisällön) sormenjälki vastaa tietokannasta löytynyttä, todetaan tiedosto haitalliseksi.

Toiminnaltaan ClamAV on laajentunut merkittävästi ja osaa suorittaa muun muassa heuristista analyysiä tiedostoille. Joka tapauksessa, verrattuna esimerkiksi Windows Defenderiin, ei ohjelmiston toimintaa ole edes suunniteltu laajamittaiseen järjestelmän tarkasteluun. Parhaimmillaan ClamAV toimii, kun se asetetaan skannaamaan potentiaalisesti vaarallisia tiedostoja sisältäviä hakemistoja. Wordpress-palvelimen tapauksessa tämä voi olla Wordpressin sisältöhakemiston /uploads/ -osio. Vastaavasti työasemakäytössä ClamAV voidaan asettaa skannaamaan käyttäjän Downloads -hakemistoa.

Euroopan unionin osarahoittama

ClamAV:n asentaminen

Debian 12 -jakelulla ClamAV:n, sekä tämän tarvittavat oheispaketit, saa asennettua suoraan pakettirepositoriosta.

```
$ apt install clamav clamav-daemon -y
```

Asennuksen jälkeen tulee tarkistaa clamav-daemon -palvelun tila. Tämän tulisi olla automaattisesti ’enabled’, sekä palvelun tulisi olla aktiivisena.

```
$ systemctl status clamav-daemon
○ clamav-daemon.service - Clam AntiVirus userspace daemon
     Loaded: loaded (/lib/systemd/system/clamav-daemon.service; enabled; preset: enabled)
    Drop-In: /etc/systemd/system/clamav-daemon.service.d
             └─extend.conf
     Active: inactive (dead)
TriggeredBy: ○ clamav-daemon.socket
  Condition: start condition failed at Mon 2025-07-07 12:24:13 EEST; 1min 40s ago
             └─ ConditionPathExistsGlob=/var/lib/clamav/daily.{c[vl]d,inc} was not met
       Docs: man:clamd(8)
             man:clamd.conf(5)
             https://docs.clamav.net/
```

Esimerkkitapauksessa palvelu on inaktiivisena, sillä tiettyä vaadittua tiedostoa ei pystytty käsittelemään oikein. **Varmin tapa tämän korjaamiseen on yksinkertainen uudelleenkäynnistys, joka on muutenkin erittäin suositeltavaa ClamAV:n asennuksen jälkeen.**

ClamAV toimii sormenjälkien perusteella. Käytännössä Cisco ja kumppanit pitävät yllä tietokantaa, johon kerätään haitallisten tiedostojen sormenjälkitunnisteita. Kyseinen tietokanta ladataan paikalliselle laitteelle, jonka avulla paikallinen ClamAV tunnistaa potentiaalisesti haitalliset tiedostot. Tämä tapahtuu `freshclam` -paketin avulla.

Ensimmäisenä kannattaa konfiguroida `freshclam` toimimaan halutulla tavalla. Oletuskonfiguraatioihin ei tarvitse tehdä suuria muutoksia, mutta tämä tiedosto on hyvä käydä läpi jotta tulevaisuudessa tietää mistä on kyse. Konfiguraatiotiedosto sijaitsee polussa `/etc/clamav/freshclam.conf`. Tätä voidaan muokata komentorivitekstieditorilla, kuten `nano`:lla tai `vim`:illä. Oletuksena sisältö on seuraava:

```
1 # Automatically created by the clamav-freshclam postinst
2 # Comments will get lost when you reconfigure the clamav-freshclam package
3
4 DatabaseOwner clamav
5 UpdateLogFile /var/log/clamav/freshclam.log
6 LogVerbose false
7 LogSyslog false
8 LogFacility LOG_LOCAL6
9 LogFileMaxSize 0
10 LogRotate true
11 LogTime true
12 Foreground false
13 Debug false
14 MaxAttempts 5
15 DatabaseDirectory /var/lib/clamav
16 DNSDatabaseInfo current.cvd.clamav.net
17 ConnectTimeout 30
18 ReceiveTimeout 0
19 TestDatabases yes
20 ScriptedUpdates yes
21 CompressLocalDatabase no
22 Bytecode true
23 NotifyClamd /etc/clamav/clamd.conf
24 # Check for new database 24 times a day
25 Checks 24
26 DatabaseMirror db.local.clamav.net
27 DatabaseMirror database.clamav.net
```

Tällä hetkellä tärkeimmät konfiguraatiot ovat riveillä 25 - 27. Näillä voidaan määrittää kuinka monta kertaa automaattiset tietokantapäivitykset ajetaan vuorokaudessa. Oletuksena oleva 24 kertaa vuorokaudessa on varsin hyvä, eikä sitä välttämättä tarvitse muuttaa. `DatabaseMirror` määrittää mistä lähteestä tietokantapäivitykset ladataan.

Euroopan unionin osarahoittama

Näistä ensimmäinen, eli ensisijainen, on hyvä määrittää maantieteellisesti mahdollisimman lähelle. Toinen `DatabaseMirror` määrittää varaosoitteen, jota käytetään jos ensimmäinen ei toimi. Tämä on hyvä jättää oletusasetukseen:

```
...
24 # Check for new database 24 times a day
25 Checks 24
26 DatabaseMirror db.fi.clamav.net
27 DatabaseMirror database.clamav.net
```

Kun freshclam:in konfiguraatioihin ollaan tyytyväisiä, voidaan tiedosto tallentaa. Tämän jälkeen clamav-freshclam -palvelu asetetaan ’enabled’ tilaan. Muutoksien voimaan saattaminen varmistetaan uudelleenkäynnistämällä palvelin / työasema:

```
$ systemctl enable clamav-freshclam
$ systemctl reboot
```

Uudelleenkäynnistyksen jälkeen näemme että clamav-freshclam -palvelu on ’enabled’ sekä aktiivinen. Palvelun logista näemme myös tietokantapäivityksen käynnistyneen.

```
$ systemctl status clamav-freshclam
● clamav-freshclam.service - ClamAV virus database updater
     Loaded: loaded (/lib/systemd/system/clamav-freshclam.service; enabled;
preset: enabled)
     Active: active (running) since Mon 2025-07-07 13:24:30 EEST; 3min 1s ago
       Docs: man:freshclam(1)
             man:freshclam.conf(5)
             https://docs.clamav.net/
   Main PID: 869 (freshclam)
      Tasks: 1 (limit: 9416)
     Memory: 5.4M
        CPU: 15ms
     CGroup: /system.slice/clamav-freshclam.service
             └─869 /usr/bin/freshclam -d --foreground=true
$ journalctl -u clamav-freshclam -f
Jul 07 13:24:30 wordpress-server systemd[1]: Started clamav-freshclam.service -
ClamAV virus database updater.
Jul 07 13:24:30 wordpress-server freshclam[869]: Mon Jul  7 13:24:30 2025 ->
ClamAV update process started at Mon Jul  7 13:24:30 2025
...
```

Toiminnan testaaminen

Nyt kysymys kuuluukin, miten voimme turvallisesti testata ClamAV:n toimintaa? Virustutkien testauskäyttöön on luotu omia testitiedostoja, joista yksi on EICAR-testitiedosto. Käytännössä tämä tiedosto sisältää merkkijonon, jonka modernit virustutkat tunnistavat. Tiedosto itsessään ei ole vaarallinen, mutta virustutka tunnistaa sen saastuneeksi.

Tiedoston voi tehdä itse ja sen sisällöksi asetetaan seuraava merkkijono:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Tämän tiedoston voi luoda esimerkiksi palvelimen käyttäjän Downloads -hakemistoon, josta se on helppo löytää:

```
$ touch ~/Downloads/eicar_testfile
$ echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' > ~/Downloads/eicar_testfile
$ cat ~/Downloads/eicar_testfile
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

`Clamscan`:in suhteen tulee muistaa, ettei sitä tulisi ajaa sudo -komennolla tai root-oikeuksin. Jos kyseessä on oikeasti vaarallinen tiedosto, on mahdollista että virus pääsee suorittamaan itsensä ClamAV:n suojauksista huolimatta. Jos tämä tapahtuu korotetuin oikeuksin, on koko järjestelmä vaarassa. Ohjeistuksena tämä on sinänsä ristiriitainen, sillä clamdscan ajetaan järjestelmäprosessina, vaikkakin `clamav`-käyttäjän alla. Joka tapauksessa, myös sen oikeuksia on rajoitettu root-käyttäjään verrattuna, eikä kyseistä `clamdscan`:ia tulisi edes tarvita tavallisena käyttäjänä.

Downloads -hakemistossa voidaan nyt ajaa ClamAV:n skannaus. Oikein toimiessaan ClamAV ilmoittaa löytäneensä EICAR-testitiedoston, joka merkitään saastuneeksi. ClamAV:n oletusasetuksilla tiedosto löydetään myös .tar -pakatusta arkistosta, joka sisältää saman tiedoston. Vertailun vuoksi, adminer-tiedosto nähdään normaalina, kuten pitääkin.

```
$ clamscan ~/Downloads/
Loading:    22s, ETA:   0s [=========================>]    8.71M/8.71M sigs
Compiling:   4s, ETA:   0s [=========================>]       41/41 tasks

/home/aim/Downloads/adminer-5.3.0-en.php: OK
/home/aim/Downloads/eicar.tar: Eicar-Signature FOUND
/home/aim/Downloads/eicar_testfile: Eicar-Signature FOUND
...
```


Euroopan unionin osarahoittama

Konfigurointi ja automaattinen skannaus

ClamAV toimii kuten pitää; se tunnistaa testitiedostot halutusti. Tämä itsessään ei kuitenkaan tarkoita, että palvelin suojattaisiin haitallisilta tiedostoilta. ClamAV:n voi konfiguroida melko monipuolisesti haitallisten tiedostojen käsittelyn osalta. Wordpress-palvelimen osalta tarkoituksena on myös suorittaa skannaus automaattisesti tietyissä tiedostosijainneissa.

Nämä konfiguraatiot saa tehtyä keskitetysti `/etc/clamav/clamd.conf` -tiedostoon:

```
#Automatically Generated by clamav-daemon postinst
#To reconfigure clamd run #dpkg-reconfigure clamav-daemon
#Please read /usr/share/doc/clamav-daemon/README.Debian.gz for details
LocalSocket /var/run/clamav/clamd.ctl
FixStaleSocket true
LocalSocketGroup clamav
LocalSocketMode 666
# TemporaryDirectory is not set to its default /tmp here to make overriding
# the default with environment variables TMPDIR/TMP/TEMP possible
User clamav
ScanMail true
ScanArchive true
ArchiveBlockEncrypted false
MaxDirectoryRecursion 15
FollowDirectorySymlinks false
FollowFileSymlinks false
ReadTimeout 180
MaxThreads 12
MaxConnectionQueueLength 15
...
```

Konfiguraatioasetuksia on melkoisesti. Tässä tapauksessa on käytännössä mahdotonta käydä niitä kaikkia läpi, mutta jos asioita haluaa hienosäätää haluamaansa suuntaan, kannattaa vilkaista `man clamd.conf` -ohjekomento.

Ensimmäisenä konfiguraatiotiedostossa kannattaa kiinnittää huomiota ClamAV:n käyttäjänimeen. Tämä käyttäjänimi on tärkeä automaattisen skannauksen suhteen, sillä loputtomien loopien estämiseksi emme salli automaattista skannausta jos tämä käyttäjä koskettaa tiedostoja.

```
...
 8 # TemporaryDirectory is not set to its default /tmp here to make overriding
 9 # the default with environment variables TMPDIR/TMP/TEMP possible
10 User clamav
11 ScanMail true
12 ScanArchive true
...
```

Tarvittavat lisäkonfiguraatiot, joilla määritellään automaattisesti skannattavat tiedostosijainnit, voidaan lisätä konfiguraatiotiedoston loppuun. Wordpress-palvelimen tapauksessa yksi `OnAccessIncludePath` -asetuksella määriteltävä tietostosijainti on `/var/lib/wordpress/wp-content/uploads` -hakemisto. Tähän hakemistoon tallennetaan kaikki käyttäjien sivustolle lataamat tiedostot, kuten kuvat ja videot.

Palvelimen turvaamiseksi voimme asettaa ClamAV:n estämään saastuneeksi luokitellun tiedoston käsittelyn kaikilta käyttäjiltä. `OnAccessPrevention` -asetuksella voidaan varmistaa ettei mikään käyttäjä, kuten `www-data`, pääse avaamaan tiedostoa. `OnAccessExtraScanning` puolestaan ajaa skannaukset jo silloin kun tiedosto luodaan tai jos sitä pyritään siirtämään.

```
...
88 # OnAccess confs
89 OnAccessIncludePath /var/lib/wordpress/wp-content/uploads
90 OnAccessPrevention yes
91 OnAccessExtraScanning yes
92 OnAccessExcludeUname clamav
```

Konfiguraatiossa on huomattava, että aiemmin mainittu `clamav` -käyttäjä tulee lisätä `OnAccessExcludeUname` -asetuksella poikkeukseksi automaattisien skannauksien laukaisijaksi. Tämä johtuu automaattisen skannauksen toimintaperiaatteesta.

Käytännössä `OnAccess` -skannaus toimii niin, että aina kun mikä tahansa käyttäjä koskee tiedostoon määritellyissä tiedostosijainneissa, suorittaa ClamAV tiedostolle skannauksen. Jos `clamav` -käyttäjää ei olisi asetettu poikkeukseksi, syntyisi loputon skannaussilmukka.

Euroopan unionin osarahoittama

`Clamd.conf` -tiedoston konfiguroinnin jälkeen on syytä ladata `clamav-daemon` -palvelu uudelleen. Tämän lisäksi tulee ajaa `clamonacc` -komento korotetuin oikeuksin (`sudo`:a käyttäen).

```
$ systemctl reload clamav-daemon
$ clamonacc
```

Jotta automaattinen skannaus olisi käytössä aina palvelimen uudelleenkäynnistyksen jälkeen, tulee tätä vastaava palvelu asettaa ’enabled’ -tilaan:

```
$ systemctl enable --now clamav-clamonacc
$ systemctl status clamav-clamonacc
● clamav-clamonacc.service - ClamAV On-Access Scanner
     Loaded: loaded (/lib/systemd/system/clamav-clamonacc.service; enabled;
preset: enabled)
     Active: active (running) since Thu 2025-07-10 12:19:57 EEST; 2s ago
...
```

Käytössä olevasta jakelusta riippuen saattaa käyttöjärjestelmässä olla käytössä Mandatory Access Control (MAC) -järjestelmä. Nämä järjestelmät voivat olla erittäin kriittisiä palvelimen turvallisuuden suhteen. Yksinkertaistetusti selitettynä, ne ikään kuin jakavat tiedostojärjestelmän osat lohkoihin. Näille lohkoille säädetään käyttäjäkohtaisia oikeuksia, joiden avulla voidaan säätää miten kukin käyttäjä voi käsitellä lohkon sisäistä informaatiota.

Yleisimmät MAC-järjestelmät ovat AppArmor sekä Security-Enhanced Linux (SELinux). Näistä ensin mainittu on oletuksena käytössä esimerkkipalvelimen Debian 12 -jakelussa. Jotta ClamAV pääsee automaattisesti skannaamaan Wordpressin `/var/lib/wordpress/wp-content/uploads/` -hakemistoa, tulee AppArmor:in tapauksessa tälle asettaa omat erityisoikeudet.

`clamscan` ja `clamdscan` ovat kaksi eri ohjelmaa. Clamscan ajetaan komennon suorittavan käyttäjän omassa ympäristössä, jolloin suoritus- ja oikeusvaatimukset ovat samat kuin käyttäjällä. Clamdscan puolestaan toimii täysin `clamav` -käyttäjän pohjalta, jolloin tälle käyttäjälle tulee määritellä oikeudet erikseen. Automaattinen skannaus toimii clamdscan:in kautta.

Ennen AppArmor -konfiguraatioita saadaan virheilmoitus manuaalinen clamdscan ajettaessa:

```
$ clamdscan /var/lib/wordpress/wp-content/uploads/
/var/lib/wordpress/wp-content/uploads: File path check failure: Permission
denied. ERROR


----------- SCAN SUMMARY -----------
Infected files: 0
Total errors: 1
Time: 0.001 sec (0 m 0 s)
Start Date: 2025:07:09 15:33:33
End Date:   2025:07:09 15:33:33
```

Jos kyseistä virheilmoitusta ei tule ja skannaus suoritetaan normaalisti, MAC-järjestelmään ei todennäköisesti kannata koskea. ClamAV:n toiminnan voi varmistaa asettamalla EICAR-testitiedoston Wordpressin tiedostoihin, jolloin ClamAV:n tulisi löytää tämä.

ClamAV:n tarvittavat AppArmor -konfiguraatiot voidaan tehdä `/etc/apparmor.d/usr.sbin.clamd` -tiedostoon. Tähän tiedostoon määritellään Wordpressin hakemistopolku, johon ClamAV:lle annetaan erilliset lukuoikeudet. Tiedoston alun tulisi näyttää vastaavalta:

```
 1 # vim:syntax=apparmor
 2 # Author: Jamie Strandboge <jamie@ubuntu.com>
 3 # Last Modified: Sun Aug  3 09:39:03 2008
 4
 5 #include <tunables/global>
 6
 7 /usr/sbin/clamd {
 8   #include <abstractions/base>
 9   #include <abstractions/nameservice>
10   #include <abstractions/openssl>
11
12   # LP: #433764:
13   capability dac_override,
14
15   # needed, when using systemd
16   capability setgid,
17   capability setuid,
18   capability chown,
19
...
```

Käytännössä tiedoston loppuun lisätään ClamAV:n automaattiseen skannaukseen konfiguroitu tiedostopolku, lukuoikeuden kera:

```
...
52   # For use with exim
53   /var/spool/exim4/** r,
54
55   # Allow home dir to be scanned
56   @{HOME}/ r,
57   @{HOME}/** r,
58
59   # wp-content/uploads access
60   /var/lib/wordpress/wp-content/uploads/** r,
61
62   # Site-specific additions and overrides. See local/README for details.
63   #include <local/usr.sbin.clamd>
64 }
```

Tämän jälkeen sekä AppArmor että ClamAV tulee ladata uudelleen, jolloin muutokset tulevat varmasti käyttöön:

```
$ systemctl reload apparmor
$ systemctl reload clamav-daemon
$ clamonacc
```

Nyt kun suoritamme manuaalisen clamdscanin kyseisessä hakemistossa sijaitseviin tiedostoihin, ilmoittaa ClamAV löytäneensä EICAR-testitiedoston:

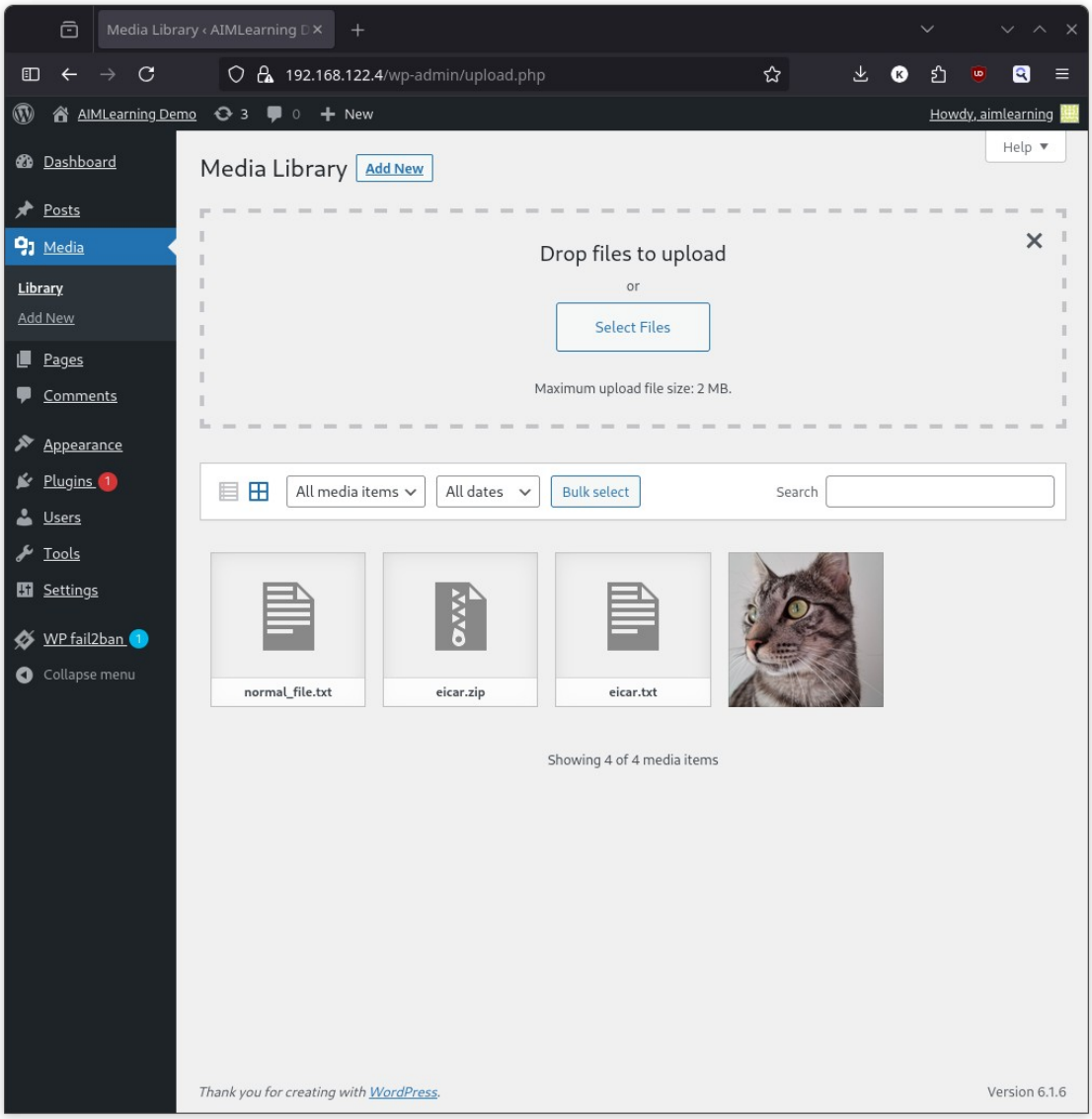
```
$ clamdscan /var/lib/wordpress/wp-content/uploads/2025/
/var/lib/wordpress/wp-content/uploads/2025/07/eicar: Eicar-Signature FOUND
...
```

Tehty konfiguraatio antaa ClamAV:lle oikeuden `/var/lib/wordpress/wp-content/uploads/` -hakemiston sisällä oleviin hakemistoihin. Itse `/uploads/` -hakemistossa `clamdscan` -komento ajettaessa, saamme edelleen virheilmoituksen. Wordpressin tapauksessa ladatut tiedostot jaotellaan kaikki päiväyksen mukaan omiin hakemistoihinsa, jolloin tämä konfiguraatio on riittävän tarkka.

Automaattista skannauksen toiminnan voi testata koittamalla tulostaa tiedoston sisällön. ClamAV estää tiedoston käsittelyn kokonaan:

```
$ cat /var/lib/wordpress/wp-content/uploads/2025/07/eicar
cat: /var/lib/wordpress/wp-content/uploads/2025/07/eicar: Operation not permitted
```

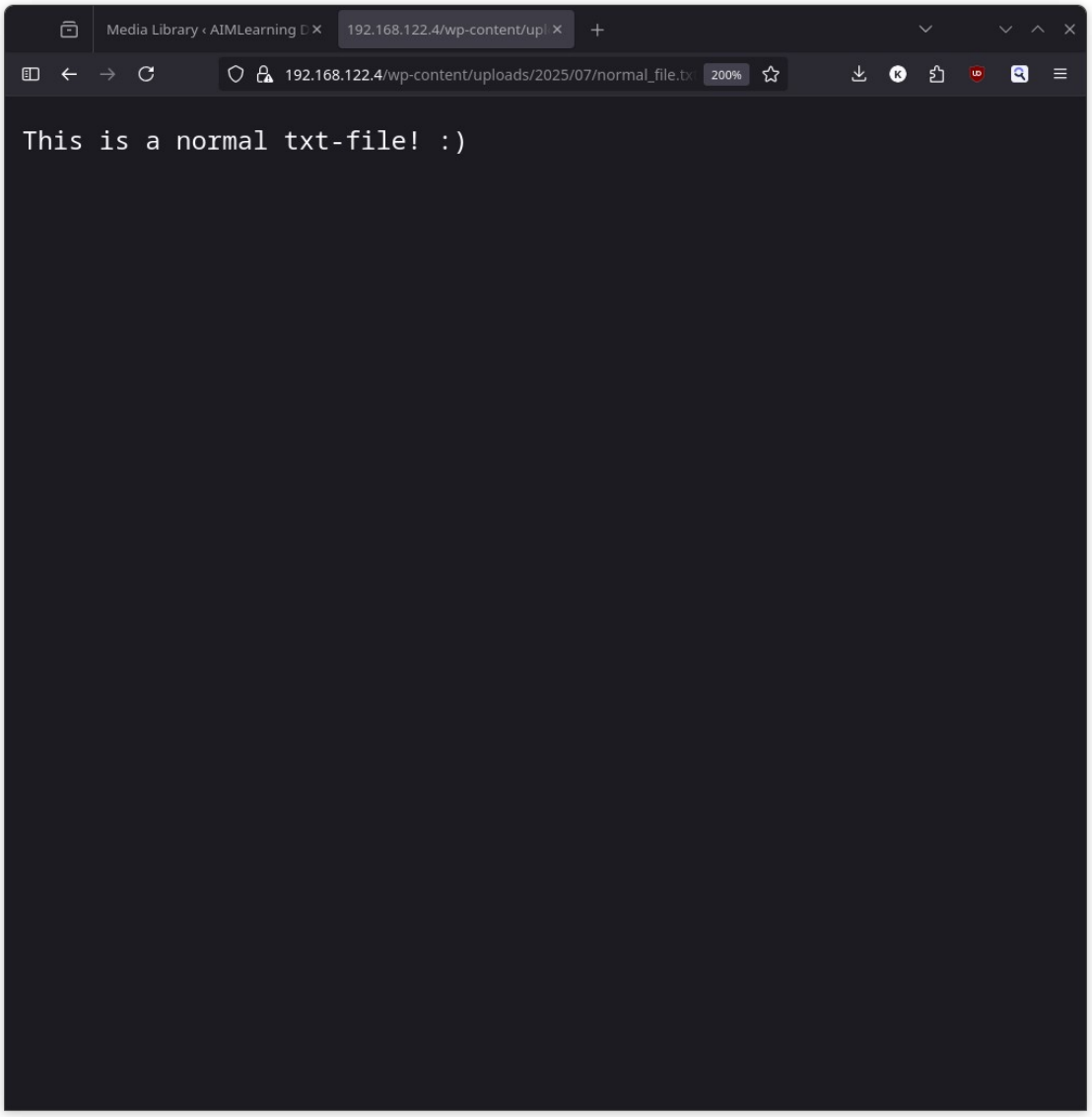
Samaisen tiedoston voi myös koittaa ladata Wordpressin tiedostoihin Wordpress Dashboardin kautta. Testitapauksessa EICAR-tekstitiedosto ladattiin palvelimelle normaalina tekstitiedostona, sekä .zip -arkistoon paketoituna. Toiminnallisuuden todistamiseksi, ladattiin palvelimelle myös tavallinen tekstitiedosto.



Kuva 36: EICAR-testitiedostot Wordpress Dashboardin mediakirjastossa

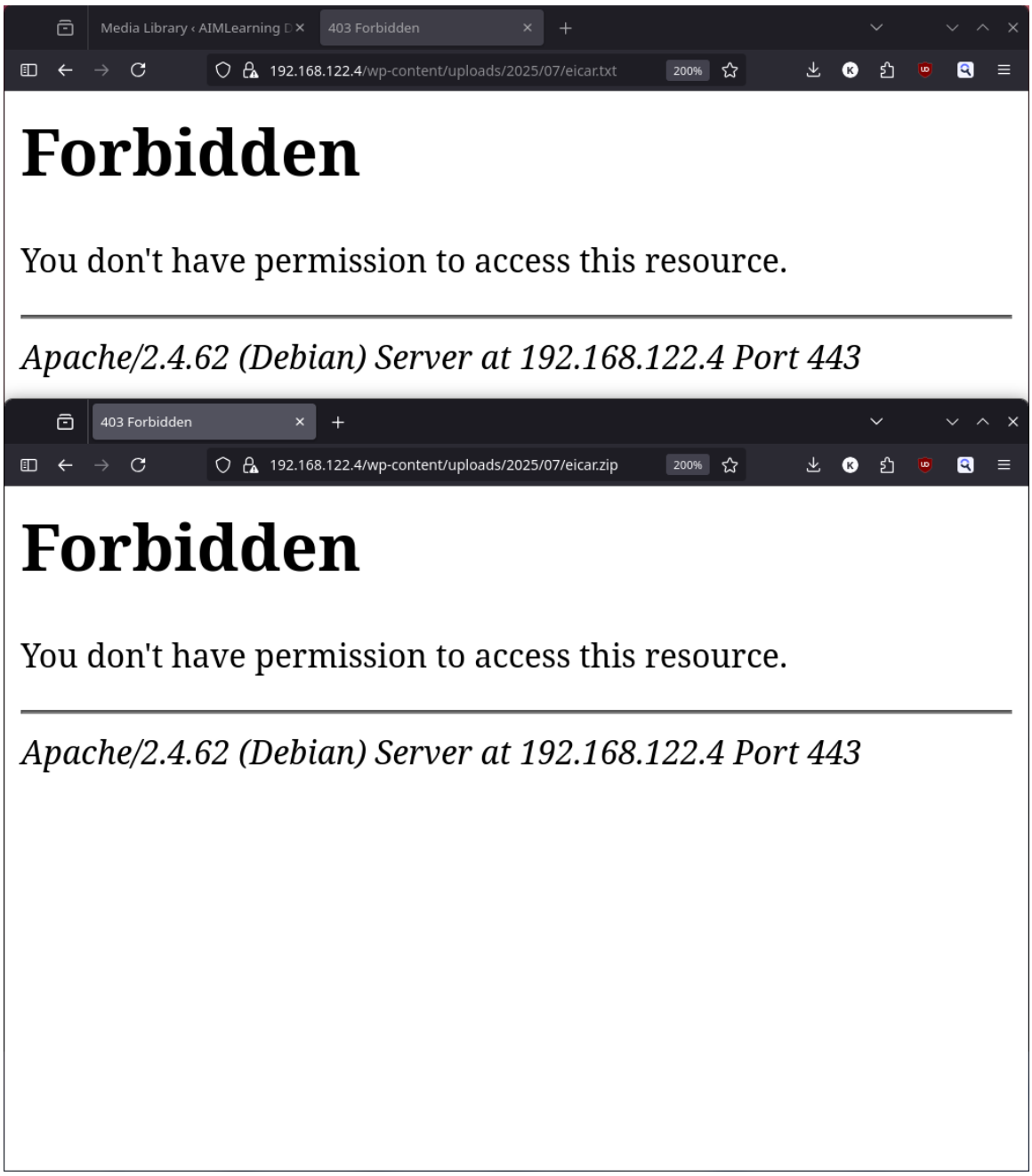
Euroopan unionin osarahoittama

Normaali tekstitiedosto voidaan avata selaimessa. Se ei sisällä mitään haitallista ClamAV:n tulkitsemana, jolloin kaikki toimii normaalisti.



Kuva 37: Normaali tekstitiedosto tulostettuna selaimeen Wordpress-palvelimelta

Jos taas vastaavasti koetamme avata EICAR-tekstitiedostoa selaimen kautta, saamme **`403 Forbidden`** virheen. Vastaava virhe tapahtuu kun EICAR.zip -tiedosto koitetaan ladata.



Kuva 38: ClamAV estää pääsyn EICAR-testitiedostoihin selaimen kautta

Tämä johtuu ClamAV:n automaattiskannauksen toimintalogiikasta. Palvelimella Wordpressin käyttäjä on `www-data`. ClamAV on konfiguroitu niin, että kun mikä tahansa käyttäjä koskettaa tiedostoa, estetään sen käsittely täysin jos tiedosto todetaan haitalliseksi. Tästä syystä Wordpress ei pysty tarjoamaan pyydettyä tiedostoa web-sivuston vierailijalle. Tiedoston pystyy poistamaan Dashboardin kautta, mutta muuta toiminnallisuutta niiden osalta ei käytännössä ole.

Halutessamme voimme varmistaa ClamAV:n tapahtumat tämän logista, josta näemme ClamAV:n todella havainneen haitalliset tiedostot:

```
$ journalctl -u clamav-daemon -f
...
SelfCheck: Database status OK.
Jul 09 16:21:07 wordpress-server clamd[54939]: Wed Jul  9 16:21:07 2025 -> /var/lib/wordpress/wp-content/uploads/2025/07/eicar.zip: Eicar-Signature(f531db596ac401be79bcfaddd20162f5:185) FOUND
Jul 09 16:22:01 wordpress-server clamd[54939]: Wed Jul  9 16:22:01 2025 -> /var/lib/wordpress/wp-content/uploads/2025/07/eicar.txt: Eicar-Signature(69630e4574ec6798239b091cda43dca0:69) FOUND
...
```

Ei kuitenkaan kannata unohtaa, että ilman lisäkonfiguraatiota haitalliset tiedostot jäävät palvelimelle.

Haitallisten tiedostojen automaattinen poistaminen

ClamAV:n pystyy konfiguroimaan poistamaan haitalliseksi todetut tiedostot muutamalla eri tavalla. Vaihtoehtoisesti tiedostot voidaan siirtää ”karanteeniin”, eli turvalliseksi tehtyyn sijaintiin, jossa näitä voidaan myöhemmin tarkastella ja analysoida.

`/etc/clamav/clamd.conf` -tiedostoon voidaan asettaa `VirusEvent` -asetus, jonka avulla voidaan suorittaa jokin tietty komento tai skripti kun haitallinen tiedosto havaitaan. Tämän käyttö ei kuitenkaan ole varsin mutkatonta. Komentojen suoraa käyttöä ei varsinaisesti suositella; myös skriptien kanssa tulee olla varovainen. Konfiguraatiotiedostoon suoraan tehtävä asetus saattaa myös vaikuttaa manuaalisen `clamscan`:in ajamiseen, mikä ei välttämättä ole toivottavaa.

Vaihtoehtoisesti `clamonacc` -ohjelmalle voidaan asettaa `--remove` -flagi, jonka avulla ohjelma pyrkii itsenäisesti poistamaan haitalliseksi todetun tiedoston. Käytännössä tämä toiminnallisuus on se mitä tällä hetkellä haemme. `clamonacc` skannaa tiedostoja sitä mukaa kun niitä luodaan tai jos niihin kosketaan. Jos ohjelma tunnistaa tiedoston haitalliseksi, haluamme poistaa sen välittömästi. Lisäksi `clamonacc` skannaa ainoastaan tiettyjä hakemistoja, joten emme vahingossa poista mitään kriittistä.

Tällä hetkellä `clamonacc` on asetettu käynnistymään automaattisesti. Tämän palvelun käynnistävän binäärin näemme tutusta `systemctl status` -näkymästä:

```
$ systemctl status clamav-clamonacc
● clamav-clamonacc.service - ClamAV On-Access Scanner
...
     CGroup: /system.slice/clamav-clamonacc.service
             └─1481 /usr/sbin/clamonacc -F --log=/var/log/clamav/clamonacc.log --move=/root/quarantine
```

Kuten huomaamme, **korostettu** rivi näyttää tavalliselta konsolikomennolta. Ainut ero on viittaus täydelliseen sijaintiin jossa tämä sijaitsee. Komennossa on valmiina `--move=/root/quarantine` -flag, joka pyrkii siirtämään haitalliseksi todetut tiedostot karanteeniin. Tätä sijaintia ei ole kuitenkaan otettu käyttöön, mutta halutessaan näin voi tehdä. Siinä tapauksessa automaattista poistoa ei välttämättä tarvitse ottaa käyttöön. **Karanteenisijainnin kanssa tulee olla äärimmäisen huolellinen, jos tämä otetaan käyttöön!**

Tarkoituksena on kuitenkin muuttaa `clamonacc` poistamaan havaitut tiedostot suoraan. Koska palvelu toimii käynnistämällä automaattisesti tavallisen ohjelman, voimme muuttaa ohjelmalle annettavia flageja palvelun tiedostosta. Kyseinen tiedosto sijaitsee polussa `/usr/lib/systemd/system/clamav-clamonacc.service`; **korostettu rivi** näyttää käynnistyksen yhteydessä suoritettavan komennon.

```
 1 # clamonacc systemd service file primarily the work of ChadDevOps & Aaron Brighton
 2 # See: https://medium.com/@aaronbrighton/installation-configuration-of-clamav-antivirus-on-ubuntu-18-04-a6416bab3b41#a340
 3
 4 [Unit]
 5 Description=ClamAV On-Access Scanner
 6 Documentation=man:clamonacc(8) man:clamd.conf(5) https://docs.clamav.net/
 7 Requires=clamav-daemon.service
 8 After=clamav-daemon.service syslog.target network.target
 9
10 [Service]
11 Type=simple
12 User=root
13 ExecStartPre=/bin/bash -c "while [ ! -S /run/clamav/clamd.ctl ]; do sleep 1; done"
14 ExecStart=/usr/sbin/clamonacc -F --log=/var/log/clamav/clamonacc.log --move=/root/quarantine
```

Yksinkertaisesti komennon `--move` vaihdetaan `--remove`:en, aivan kuten tavallisen konsolikomennon kanssa:

```
...
13 ExecStartPre=/bin/bash -c "while [ ! -S /run/clamav/clamd.ctl ]; do sleep 1; done"
14 ExecStart=/usr/sbin/clamonacc -F --log=/var/log/clamav/clamonacc.log --remove
```


Euroopan unionin osarahoittama

Muita muutoksia tiedostoon ei tarvita. Tämän jälkeen tiedosto tallennetaan, `systemd`:n palvelut ladataan uudelleen ja `clamav-clamonacc` käynnistetään uudelleen:

```
$ systemctl daemon-reload
$ systemctl restart clamav-clamonacc
$ systemctl status clamav-clamonacc
● clamav-clamonacc.service - ClamAV On-Access Scanner
     Loaded: loaded (/lib/systemd/system/clamav-clamonacc.service; enabled;
preset: enabled)
     Active: active (running) since Mon 2025-07-21 12:25:50 EEST; 3s ago
       Docs: man:clamonacc(8)
             man:clamd.conf(5)
             https://docs.clamav.net/
    Process: 1547 ExecStartPre=/bin/bash -c while [ ! -S
/run/clamav/clamd.ctl ]; do sleep 1; done (code=exited, status=0/SUCCESS)
   Main PID: 1548 (clamonacc)
      Tasks: 8 (limit: 9416)
     Memory: 6.6M
        CPU: 23ms
     CGroup: /system.slice/clamav-clamonacc.service
             └─1548 /usr/sbin/clamonacc -F --log=/var/log/clamav/clamonacc.log
--remove
```

Nyt clamonacc pyrkii poistamaan haitalliseksi todetut tiedostot välittömästi. Toimintaa voidaan testata esimerkiksi palvelimelta löytyvillä EICAR-testitiedostoilla:

```
$ ls /var/lib/wordpress/wp-content/uploads/2025/07
eicar1.zip  normal_file.txt
$ cat /var/lib/wordpress/wp-content/uploads/2025/07/eicar1.zip
cat: /var/lib/wordpress/wp-content/uploads/2025/07/eicar1.zip: Operation not
permitted
$ ls /var/lib/wordpress/wp-content/uploads/2025/07
normal_file.txt
$ journalctl -u clamav-daemon -f
...
Jul 21 13:06:22 wordpress-server clamd[604]: Mon Jul 21 13:06:22 2025 ->
/var/lib/wordpress/wp-content/uploads/2025/07/eicar1.zip: Eicar-
Signature(f531db596ac401be79bcfaddd20162f5:185) FOUND
```

Vastaavasti toiminnallisuuden voi varmistaa lataamalla Wordpress Dashboardin kautta EICAR-testitiedoston palvelimelle. Clamonacc tunnistaa ja poistaa tiedoston välittömästi.

```
$ journalctl -u clamav-daemon -f
...
Jul 21 13:11:46 wordpress-server clamd[604]: Mon Jul 21 13:11:46 2025 ->
/var/lib/wordpress/wp-content/uploads/2025/07/eicar.txt: Eicar-
Signature(69630e4574ec6798239b091cda43dca0:69) FOUND
Jul 21 13:13:12 wordpress-server clamd[604]: Mon Jul 21 13:13:12 2025 ->
/var/lib/wordpress/wp-content/uploads/2025/07/eicar1.zip: Eicar-
Signature(f531db596ac401be79bcfaddd20162f5:185) FOUND
```

Lähteet:

https://wiki.debian.org/ClamAV

https://docs.clamav.net/manual/Usage/Configuration.html

man freshclam.conf

https://docs.clamav.net/manual/OnAccess.html

https://docs.trendmicro.com/all/ent/de/v1.5/en-us/de_1.5_olh/ctm_ag/ctm1_ag_ch8/t_test_eicar_file.htm

https://www.eicar.org/download-anti-malware-testfile/

man clamd.conf

Euroopan unionin osarahoittama

Loppusanat

Onneksi olkoon, palvelimen konfigurointi on tämän oppaan osalta tehty! Olemme onnistuneesti kasvattaneet palvelimen tietoturvaominaisuuksia ja toivottavasti kaikki tarvittava toiminnallisuus on myös tallella.

Tämä ei kuitenkaan tarkoita että työ ja palvelin ovat täysin valmiita. Nykypäivän haasteena on tietoympäristön jatkuva muuttuminen ja eläminen. Käytännössä kaikista ohjelmistoista löytyy loputtomiin erilaisia bugeja ja haavoittuvuuksia, jonka vuoksi on tärkeää pysyä niistä kärryillä. Ainakin palvelimen ohjelmistojen päivittäminen ajan tasalle (ja niiden pitäminen päivitettynä) on ensisijaisen tärkeää tietoturvan ylläpitämisen kannalta.

Kuten oppaan alussa mainittiin; on täysin mahdotonta kattaa kaikkia muuttujia joita oppaan käsittelemät aiheet sisältävät. Tämän vuoksi suosittelen erittäin vahvasti tutustumaan eri ohjelmistojen toimintaan ja konfigurointiin syvällisemmin. Oppaassa käydyt esimerkit on luotu tiettyä mallia silmällä pitäen, joka todennäköisesti ei ole sama kuin muissa palvelimissa.

Riippuen palvelimen käyttötarkoituksesta, käyttäjistä sekä loogisesta ja fyysisestä sijainnista, on uhkakuva aina uniikki. Tämän takia suojaus joudutaan suunnittelemaan ja rakentamaan tilannekohtaisesti. Tarvittaessa myös muutoksia joudutaan tekemään. Uhkakuvan mukaisten konfiguraatioiden luomista helpottaa valtavasti ajatusmalli: sallitaan ainoastaan se toiminta joka on välttämätöntä.

Tämän oppaan tarkoituksena ei ole toimia kaikkitietävänä kirjana Linux-palvelinten turvaamiseen. Kuitenkin, jos oppaan mukana on tehnyt palvelinkonfiguraatioita kannesta kanteen, voi melko turvallisin mielin lähteä soveltamaan omaa osaamistaan. Jos aihepiiri kiinnostaa, suosittelen suuresti jatkamaan sen parissa.

Liite
Hyvät käytännöt

Salasanojen hallinta

Salasanat ovat todennäköisesti kaikista yleisin tapa tunnistaa käyttäjä, kun tämä kirjautuu johonkin palveluun, käyttöjärjestelmään tai muuhun suojattavaan kohteeseen. Verkkosivustoille tunnistautuessa salasana on yleensä jonkinlainen merkkijono, älypuhelin käynnistettäessä PIN-koodi tai kuvio.

Salasanoja pidetään monesti itsestäänselvyytenä, mutta niihin liittyy myös vanhentuneita tai kokonaan vääriä uskomuksia. Hyvin monella on yksi ja sama salasana jokaisessa käyttämässään palvelussa. Lisäksi kansainvälisesti yleisimmät salasanat ovat yksinkertaisesti tolkuttoman huonoja:

TOP 10 kansainvälisesti yleisimmät salasanat Nordpass 2024		
Sija	**Salasana**	**Esiintyvyys**
1.	*123456*	3 018 050
2.	*123456789*	1 625 135
3.	*12345678*	884 740
4.	*password*	692 151
5.	*qwerty123*	642 638
6.	*qwerty1*	583 630
7.	*111111*	459 730
8.	*12345*	395 573
9.	*secret*	363 491
10.	*123123*	351 576

Taulukko 3: 2024 Top 10 kansainvälisesti yleisintä salasanaa.
Lähde: https://nordpass.com/

Nykyään salasanan sijaan voidaan käyttää erilaisia menetelmiä. Biometrinen tunnistautuminen, avainparitunnistautuminen ja lohkoketjupohjainen tunnistautuminen ovat kaikki mahdollisia toteuttaa. Kuitenkin pitkälti myös näissä salasanat, muodossa tai toisessa, ovat edelleen läsnä.

Euroopan unionin osarahoittama

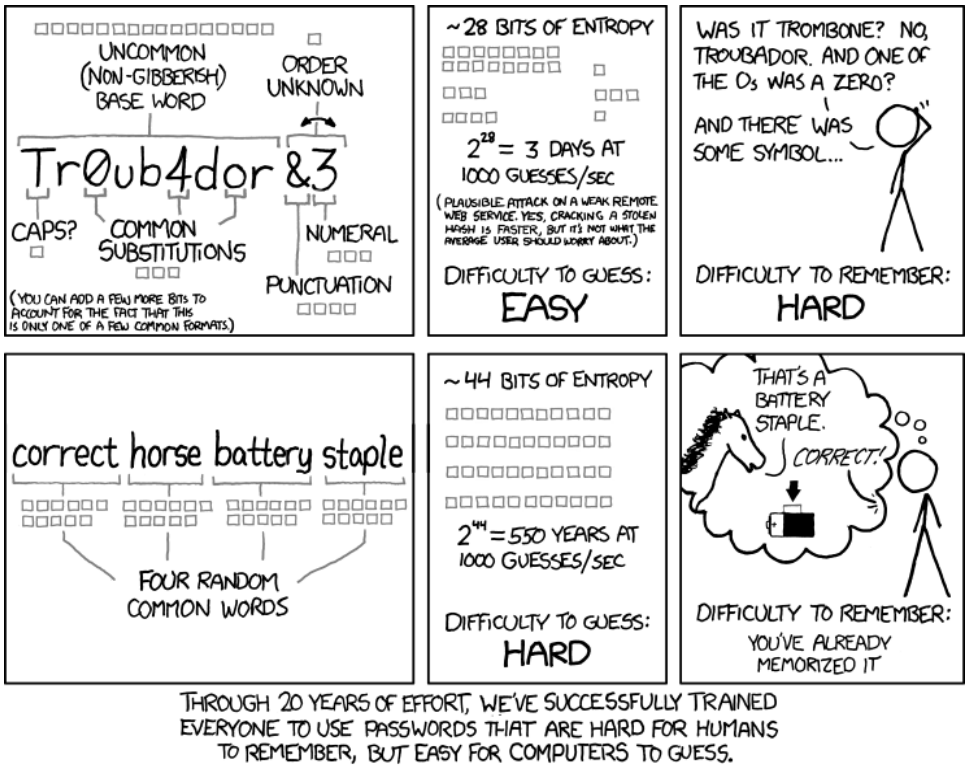
Tämän vuoksi salasanoista ja niiden turvallisuudesta kannattaa pitää huolta. Jokaisessa palvelussa kannattaa olla erillinen salasana, jota helpottamaan salasanapankit ja -generaattorit ovat hyödyllisiä.

Millainen on hyvä salasana

Hyvin lyhyesti sanottuna hyvä salasana on sellainen, jossa on mahdollisimman suuri entropia. Selkosuomeksi käännettynä, salasanan tulee olla mahdollisimman monimutkainen ja pitkä, jotta sen arvaaminen koneellisesti on hyvin työlästä ja aikaavievää.

Teoriassa voidaan ajatella, että mikä tahansa salasana voidaan murtaa. Vaikka salasana olisi 1000 merkkiä pitkä, sisältäen erikoismerkkejä, numeroita ja satunnaisia kirjaimia, on tämäkin salasana mahdollista murtaa. Murtaminen todennäköisesti kuitenkin veisi miltä tahansa nykyaikaiselta laitteistolta enemmän aikaa, mitä maailmankaikkeuden lämpökuolemaan on jäljellä.

Meidän ei siis kannata tavoitella kyseisenkaltaista salasanaa, vaan sellaista joka on riittävä nykyaikaiset vaatimukset huomioiden. Tässä tulee myös ottaa huomioon käytännöllisyys. Joitakin salasanoja joutuu kirjoittamaan toistuvasti käsin, jolloin ne tulee opetella ulkoa. Toisaalta monia salasanoja ei tarvitse itse edes tietää, sillä ne voidaan kopioida salasanapankista. Tällöin ne voivat olla yli 128 merkkiä täyttä siansaksaa. Helposti muistettavan salasanan ei kuitenkaan tarvitse olla yhtään turvattomampi.



Kuva 39: xkcd 936 -sarjakuva.
Lähde: https://xkcd.com/ (CC BY-NC 2.5)

KYBERTURVALLISUUSKESKUS SUOSITTELEE, ETTÄ SALASANA OLISI VÄHINTÄÄN 15 MERKKIÄ PITKÄ, SISÄLTÄÄ ISOJA JA PIENIÄ KIRJAIMIA SEKÄ ERIKOISMERKKEJÄ. SALASANASSA EI TULISI OLLA HENKILÖKOHTAISIA TIETOJA TAI NÄPPÄIMISTÖN ASETTELUN MUKAISIA KUVIOITA. MYÖS YLEISESTI KÄYTÖSSÄ OLEVIA SALASANOJA ON SYYTÄ VÄLTTÄÄ.

Entropian laskeminen

Salasanojen entropiaa mitataan bitteinä. Mitä suurempi tämä bittimäärä on, sen turvallisempana salasanaa voidaan pitää, ainakin teoriassa. 64 a-kirjainta peräkkäin tuottaa suuren entropian, mutta on murrettavissa sekunneissa. Käytännössä entropian ollessa 128 bittiä tai suurempi, voimme puhua jo hyvin vahvasta salasanasta. Eräs yhtälö, jolla salasanan entropia voidaan laskea, näyttää seuraavalta:

$$E = L \times \log_2(R)$$

Yhtälössä L on salasanan pituus, R on kokonaismäärä kaikista mahdollisista (englannin kielen) merkeistä joita salasana saattaisi sisältää. Entropialuku saadaan, kun R lasketaan 2-kantaisella logaritmilla, jonka tulos kerrotaan salasanan pituudella L.

Salasanan luonti

Salasanan **`salasana123`** pituus (L) on 11 merkkiä. Se sisältää pieniä kirjaimia ja numeroita, joten kaikkia mahdollisia merkkejä (R) on 26 + 10 = 36. Kyseisen salasanan entropia bitteinä on 56,9. Kuten olettaa saattaa, kyseinen salasana ei ole kovinkaan vahva, sen pystyy myös arvaamaan melko helposti.

Vastaavasti **`kp^Re5*mM4Ycc!6duy2EF!t$b!J^&d^J`** näyttää jo silmällä hyvin vaikeasti arvattavalta salasanalta. Salasana on 32 merkkiä pitkä, koostuu isoista ja pienistä kirjaimista, numeroista ja erikoismerkeistä. Yhteenlaskettuna salasana voisi sisältää 10 + 26 + 26 + 32 = 94 erilaista merkkiä. Bitteinä laskettu entropia on hyvin korkea; 209,7. Kyseinen salasana on hyvä tapauksissa, joissa sitä ei tarvitse kirjoittaa käsin. Jos kuitenkin kyseessä on esimerkiksi oman salasanapankin pääsalasana, tulee se muistaa ulkoa.

Salalauseen luonti

Perinteisimmän salasanan sijaan voi käyttää salalausetta. Salalause koostuu useammasta tavallisesta sanasta, erotettuna erikoismerkeillä ja mahdollisesti numeroista. Erityisesti suomen kielessä etuna on, että salalauseesta voi muodostaa vaikkapa lyhyen tarinan, joka on kansainvälisesti ajateltuna hyvin vaikea murtaa. Salalausetta muodostaessa tulee

Euroopan unionin osarahoittama

kuitenkin muistaa, että siihen ei kannata sisällyttää mitään henkilökohtaista tai salattavaan kohteeseen liittyvää tietoa.

Salalauseella **`Suunnittelimme&Retkea&Puistoon37`** saavutetaan sama entropia kuin kuin edellisen salasanan kanssa. Salalauseesta voidaan tehdä helposti myös paljon pidempi. **`Suunnittelimme&Retkea&Puistoon37&Sitten&Ikavasti&Satoikin:(`** on edeltävää kolme sanaa pidempi ja nyt sen lopussa on suruhymiö lisäämässä erikoismerkkejä. Salasanan entropia on jo kasvanut 386,7 bittiin ja on silti hyvin helppo opetella ulkoa.

Sanakirjahyökkäys voi ensisilmäyksellä vaikuttaa tekevän salalauseista oleellisesti huonompia perinteisiin salasanoihin verrattuna. Kuitenkin kuusi suomenkielistä, eri tavoilla taivutettua sanaa, erikoismerkkien ja numeroiden kera, on käytännössä hyvin vaikea yhtälö mille tahansa sanakirjahyökkäykselle. Kun vastaava salalause yhdistetään hyvin vahvaan hash-algoritmiin sekä ohjelmistopohjaiseen kaksivaiheiseen tunnistautumiseen, on hyökkääjän erittäin hankalaa murtaa pääsyä tilille.

Vahvojen salasanojen ja salalauseiden luontiin kannattaa hyödyntää erilaisia salasanageneraattoreita. Jos käytössä on salasanapankki, on erittäin todennäköistä että se sisältää mahdollisuuden generoida todella vahvoja salasanoja.

Salasanapankit

Oletko koskaan miettinyt kuinka moneen verkkopalveluun olet vuosien saatossa rekisteröitynyt? Todennäköisesti vastausta tuohon kysymykseen saa miettiä hetken, jonka jälkeen pääsee lopputulokseen; en tiedä. Pelkästään sosiaalisen median palveluita on olemassa kymmeniä, verkkopelaamiseen jälleen kymmeniä ja satunnaisia verkkosivustoja loputtomiin.

Todennäköisesti moni on tämän takia sortunut, ainakin jossain vaiheessa, käyttämään jokaisessa palvelussa yhteisesti vain muutamaa salasanaa. Tämä saattaa kuitenkin helposti kostautua. Tällä käytännöllä riittää, että käyttäjätunnukset murretaan vain yhdestä palvelusta, jolloin hyökkääjillä on pääsy jokaiseen tiliin samalla salasanalla.

Nykyään salasananhallinta on tehty hyvin helpoksi, sillä selaimet ja käyttöjärjestelmät ehdottavat kaikki omia salasanapankkejaan. Näiden turvallisuudesta voi kuitenkin olla montaa mieltä, eikä siitä välttämättä voi mennä lainkaan takuuseen.

Varsinaisia salasanapankkiohjelmistoja ja -palveluita on kuitenkin saatavilla moneen lähtöön, eikä niistä tarvitse kuluttajana välttämättä edes maksaa. Ominaisuudet eroavat, mutta lähtökohtaisesti parhaissa palveluissa on vahvasti salatut järjestelmät, zero-knowledge -ominaisuudet ja monivaiheiset tunnistautumiset.

Muutamia varteenotettavia palveluita ja ohjelmistoja ovat muun muassa:

	Bitwarden	KeePassXC	1Password	Proton Pass
Ilmainen palvelu saatavilla	Kyllä	Kyllä	Ei	Kyllä
Avoin lähdekoodi	Kyllä	Kyllä	Ei	Kyllä
E2E-salaus, zero-knowledge	Kyllä	Ei verkkotoiminnallisuutta	Kyllä	Kyllä
Palvelimet	Cloud (USA & EU), Self-hosted	Self-hosted	Cloud (USA, Kanada, EU)	Cloud (EU, Sveitsi)
Kolmansien osapuolten turvallisuuskatsaukset	Kyllä, Saatavilla	Kyllä, Saatavilla	Kyllä, Saatavilla	Kyllä, Saatavilla
Täyttää GDPR-vaatimukset	Kyllä	Ei verkkotoiminnallisuutta	Kyllä	Kyllä
Tuetut käyttöjärjestelmät	Windows, Linux, MacOS, Android, iOS	Windows, Linux, MacOS	Windows, Linux, MacOS, Android, iOS	Windows, Linux, MacOS, Android, iOS

Taulukko 4: Eri salasanapankkien ominaisuuksia

Lähdekoodi - Osa palveluista on suljettuja, pitäen tiedot järjestelmäkoodista täysin itsellään, kun taas osa palveluista on täysin avoimeen lähdekoodiin perustuvia. Turvallisuusmielessä molemmissa vaihtoehdoissa on hyvät ja huonot puolensa, mutta lähtökohtaisesti avoin lähdekoodi takaa ainakin yksityisyyteen liittyvien ominaisuuksien läpinäkyvyyden.

E2E-salaus, zero-knowledge - E2E-salaus (End-to-End -encryption) salaa käyttäjien dataliikenteen kauttaaltaan. Tiedot salataan päätelaitteessa, eikä niitä pääse purkamaan missään vaiheessa reittiä. Salasanapankin tapauksessa tiedot voidaan purkaa ainoastaan käyttäjän toimesta. Zero-knowledge tarkoittaa, että palvelun tarjoajalla ei ole minkäänlaista pääsyä käyttäjien salausavaimiin, salasanapankkeihin tai näiden sisältöön.

Palvelinten sijainti - Palveluita on saatavana muun muassa pilvipohjaisina ja self-hosted -versioina. Pilvipohjaisissa palveluissa salatut tiedot tallennetaan palveluntarjoajan palvelinten tietokantoihin. Pilvipohjainen palvelu valitessa kannattaa kiinnittää huomiota palvelinten sijaintiin, sillä paikallinen lainsäädäntö voi vaikuttaa oleellisesti datan käsittelyyn. Self-hosted -palvelut voidaan ottaa käyttöön omille palvelimille tai laitteille. Mikään tieto ei kulje eteenpäin, mutta myös kokonaisturvallisuudesta ollaan itse vastuussa.

Euroopan unionin osarahoittama

Kolmansien osapuolten turvallisuuskatsaukset - Hyvät palvelut ovat avoimia turvallisuutensa suhteen ja julkaisevat kolmansien osapuolten tekemiä turvallisuuskatsauksia. Turvallisuuskatsauksissa selvitetään mahdollisia tietoturva-aukkoja ja -ongelmia, joista ei välttämättä olla muuten tietoisia.

GDPR-vaatimukset - GDPR (General Data Protection Regulation), on Euroopan Unionin yleinen tietosuoja-asetus. Tietosuoja-asetus on hyvin laaja mekanismi loppukäyttäjien henkilötietojen suojaamiseen. GDPR-asetusta joutuvat noudattamaan kaikki Euroopan Unionin sisällä toimivat yritykset, sekä kaikki yritykset jotka käsittelevät EU:n kansalaisten henkilötietoja.

Kaksivaiheinen tunnistautuminen / 2FA

Kaksivaiheinen tunnistautuminen tarkoittaa kirjaimellisesti sitä, että kun tiettyyn palveluun kirjaudutaan, tunnistautumisessa on kaksi vaihetta. Yleensä nämä ovat salasana ja turvakoodi, jotka vaaditaan käyttäjän kirjautuessa.

Tämä tuo lisäturvaa, varsinkin kriittisissä palveluissa. Vaikka jostain syystä salasana olisi hyökkääjän hallussa, ei tämä pääse kirjautumaan palveluun ilman toisen vaiheen hyväksyntää. Epäonnistuneista kirjautumisyrityksistä yleensä ilmoitetaan käyttäjälle, jolloin tämä tietää vaihtaa salasanansa uuteen. Tämän takia kaksivaiheista tunnistautumista suositellaan kaikkialle, jossa se on mahdollista ottaa käyttöön.

Metodeja kaksivaiheiseen tunnistautumiseen on monia. Sähköpostivarmennuksessa käyttäjän sähköpostiin lähetetään koodi, jonka käyttäjä syöttää sivuston sitä pyytäessä. Vastaavalla tavalla toimii myös tekstiviesti tai puheluvarmistus. Nämä eivät kuitenkaan ole verrattain kovin turvallisia vaihtoehtoja. Erityisesti tekstiviestipohjaista varmennusta ei suositella, sillä niiden väärentäminen tai päätyminen vääriin käsiin on hyvin riskialtista.

Yleisesti käytännön turvallisena vaihtoehtona on pidetty ohjelmallista varmistusta. Autentikaatiosovellus voi olla vaikkapa käyttäjän puhelimessa ja siihen sidotaan yhteys jokaiseen palveluun, johon kaksivaiheinen tunnistautuminen liitetään. Kaikki liikenne voidaan suorittaa päästä-päähän salatun kanavan ylitse, jolloin ulkopuoliset eivät pysty selvittämään käytössä olevia vahvistuskoodeja. Sovelluksen turvakoodit myös päivittyvät tiheään tahtiin, jolloin vanhasta koodista ei ole enää hyötyä hyökkääjälle.

Fyysiset 2FA-tunnistautumislaitteet ovat myös maininnan arvoinen ryhmä. Käytännössä nämä voivat olla USB-tikun näköisiä laitteita, jotka käyttäjä asettaa laitteeseensa vahvistaakseen autentikoinnin.

Lähteet:

https://proton.me/blog/what-is-password-entropy

https://xkcd.com/936/

https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/file/Salasanat_haltuun.pdf

https://oulurepo.oulu.fi/bitstream/handle/10024/14101/nbnfioulu-201909212922.pdf;jsessionid=94A43C23E62C1BCB28E507E927888180?sequence=1

https://bitwarden.com/password-generator/

https://www.canada.ca/en/government/system/digital-government/online-security-privacy/password-guidance.html

https://nordpass.com/most-common-passwords-list/

https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_fi.htm

https://bitwarden.com/help/

https://keepassxc.org/

https://1password.com/

https://support.1password.com/

https://proton.me/pass/

https://www.techtarget.com/searchsecurity/definition/two-factor-authentication

https://www.hypr.com/security-encyclopedia/blockchain-authentication

Euroopan unionin osarahoittama

Yleiset käsitteet

Tämän osion käsitteiden selitykset on generoitu tekoälyä hyödyntäen (Github Copilot - GPT-4.1).

AppArmor
AppArmor on Linuxin tietoturvamoduuli, joka rajoittaa ohjelmien oikeuksia profiilien avulla. Profiilit määrittelevät, mitä tiedostoja ja järjestelmäresursseja ohjelmat saavat käyttää. AppArmor auttaa estämään haitallista toimintaa ja rajoittamaan vahinkoja, jos sovellus kaapataan. (ks. SELinux)

Apache2
Apache2 on yksi maailman käytetyimmistä avoimen lähdekoodin HTTP-palvelinohjelmistoista. Sitä käytetään verkkosivujen ja -palveluiden tarjoamiseen käyttäjille. Apache2 on erittäin muokattavissa ja tukee laajasti erilaisia laajennuksia ja konfiguraatioita.

apt
apt (Advanced Package Tool) on Debian-pohjaisten Linux-jakeluiden paketinhallintajärjestelmä. Sen avulla käyttäjät voivat asentaa, päivittää ja poistaa ohjelmistoja sekä hallita ohjelmistoriippuvuuksia helposti. apt hakee ohjelmistot ja päivitykset määritellyistä ohjelmistovarastoista (ks. paketinhallinta, repositorio).

arptables
arptables on työkalu, jolla hallitaan ja suodatetaan ARP-liikennettä Linux-järjestelmissä. Sitä käytetään erityisesti verkon tietoturvan ja liikenteen hallintaan. ARP (Address Resolution Protocol) yhdistää IP-osoitteet MAC-osoitteisiin lähiverkossa.

backend
Backend viittaa ohjelmiston palvelinpuoleen, joka käsittelee tietoja, liikennettä ja liiketoimintalogiikkaa käyttäjän näkymättömissä. Wordpress-kehityspalvelimella backend sisältää esimerkiksi tietokannan, PHP-tulkin ja palvelinohjelmiston. (ks. frontend)

bash
Bash (Bourne Again SHell) on suosittu komentotulkki Linux- ja Unix-järjestelmissä. Sitä käytetään komentojen suorittamiseen, skriptien ajamiseen ja järjestelmän hallintaan komentoriviltä. Bash tukee monipuolisia ohjelmointiominaisuuksia, kuten ehtolauseita ja silmukoita. (ks. shell, skripti)

BSD
BSD (Berkeley Software Distribution) on Unix-pohjaisten käyttöjärjestelmien perhe, johon kuuluvat esimerkiksi FreeBSD, OpenBSD ja NetBSD. BSD-järjestelmät ovat tunnettuja vakaudestaan ja tietoturvastaan, ja niitä käytetään usein palvelimissa.

Certificate Authority (CA)
Certificate Authority (CA) on luotettu kolmas osapuoli, joka myöntää ja allekirjoittaa digitaalisia varmenteita (sertifikaatteja). CA varmistaa, että verkkosivuston tai palvelimen identiteetti on aito, mikä mahdollistaa turvallisen TLS/SSL-salatun yhteyden. (ks. sertifikaatti, TLS/SSL)

chroot
chroot on Linuxin ja Unixin ominaisuus, jolla ohjelman juurihakemisto vaihdetaan eristettyyn ympäristöön. Tätä käytetään esimerkiksi ohjelmien testaamiseen turvallisesti tai palveluiden eristämiseen muusta järjestelmästä.

ClamAV
ClamAV on avoimen lähdekoodin virustorjuntaohjelmisto, jota käytetään erityisesti Linux-palvelimilla. Se tunnistaa ja poistaa haittaohjelmia, kuten viruksia, troijalaisia ja matoja. ClamAV:ta voidaan käyttää automaattisesti tarkistamaan esimerkiksi Wordpressin latauskansioita.

CMS
CMS (Content Management System) on sisällönhallintajärjestelmä, jonka avulla käyttäjät voivat luoda, muokata ja hallita verkkosivujen sisältöä ilman ohjelmointitaitoja. Wordpress on maailman suosituin CMS, ja sitä käytetään laajasti blogeissa ja yrityssivuilla.

DHCP
DHCP (Dynamic Host Configuration Protocol) on protokolla, joka jakaa automaattisesti IP-osoitteita ja muita verkkoparametreja laitteille verkossa. Tämä helpottaa verkon hallintaa, koska käyttäjien ei tarvitse määrittää osoitteita käsin.

Debian 12
Debian 12 on Debian Linux -jakelun kahdestoista pääversio, joka tunnetaan vakaudestaan ja laajasta ohjelmistotuesta. Debian 12 soveltuu hyvin palvelinkäyttöön, kuten Wordpress-kehityspalvelimen alustaksi.

EICAR
EICAR (European Institute for Computer Antivirus Research) on eurooppalainen tietoturva-alan yhteistyöjärjestö, joka keskittyy haittaohjelmien torjuntaan ja tietoturvatutkimukseen. EICAR kehittää standardeja ja testausmenetelmiä tietoturvaohjelmistoille.

EICAR-testitiedosto
EICAR-testitiedosto on EICARin kehittämä vaaraton tekstitiedosto, jota käytetään virustorjuntaohjelmistojen toiminnan testaamiseen. Tiedosto ei sisällä haitallista koodia, mutta yleisesti virustorjuntaohjelmat tunnistavat sen testivirukseksi ja reagoivat siihen kuin oikeaan uhkaan.

ebtables
ebtables on työkalu, jolla hallitaan ja suodatetaan Ethernet-kehysten liikennettä Linux-järjestelmissä. Sitä käytetään erityisesti sillattujen (bridged) verkkojen tietoturvan hallintaan.

Euroopan unionin osarahoittama

Fail2Ban
Fail2Ban on ohjelma, joka suojaa palvelinta automaattisesti estämällä IP-osoitteet, joista havaitaan toistuvia epäonnistuneita kirjautumisyrityksiä. Se auttaa ehkäisemään brute force -hyökkäyksiä esimerkiksi SSH- ja Wordpress-kirjautumissivuilla.

firewalld
firewalld on dynaaminen palomuuripalvelu, joka hallitsee iptables- ja nftables-sääntöjä Linux-järjestelmissä. Se mahdollistaa palomuurisääntöjen muuttamisen lennossa ilman palvelun uudelleenkäynnistystä.

frontend
Frontend tarkoittaa ohjelmiston osaa, jonka käyttäjä näkee ja jonka kanssa hän on vuorovaikutuksessa, kuten verkkosivun käyttöliittymä. Wordpressissä frontend on esimerkiksi sivuston teema ja ulkoasu. (ks. backend)

FTP
FTP (File Transfer Protocol) on vanha tiedostonsiirtoprotokolla, jota käytetään tiedostojen siirtoon palvelimen ja asiakkaan välillä. FTP ei ole salattu, joten sitä ei suositella arkaluontoiseen tiedonsiirtoon. (ks. SFTP, SCP)

GNU
GNU on avoimen lähdekoodin ohjelmistoprojekti, joka loi monia Unix-yhteensopivia työkaluja ja ohjelmia. GNU-projektin ohjelmat muodostavat suuren osan Linux-jakeluiden perusohjelmistoista.

GNU/LINUX
GNU/Linux tarkoittaa käyttöjärjestelmää, joka koostuu GNU-ohjelmista ja Linux-ytimestä (kernel). Useimmat Linux-jakelut, kuten Debian, perustuvat tähän yhdistelmään. (ks. Linux, kernel)

GUFW
GUFW on graafinen käyttöliittymä UFW-palomuuriin, joka tekee palomuurin hallinnasta helppoa myös aloittelijoille. (ks. UFW)

GZIP
GZIP on tiedostojen pakkausohjelma, jota käytetään usein yhdessä TAR:n kanssa suurten tiedostojen ja kansioiden pakkaamiseen ja siirtämiseen.

HTML
HTML (HyperText Markup Language) on verkkosivujen rakenteen ja sisällön kuvaamiseen käytetty merkintäkieli. Kaikki verkkosivut rakentuvat HTML:n varaan.

HTTP
HTTP (HyperText Transfer Protocol) on protokolla, jolla verkkosivut ja niiden sisältö siirretään selaimelle palvelimelta.

HTTPS
HTTPS on HTTP-protokollan salattu versio, joka käyttää TLS/SSL-salausta suojaamaan tiedonsiirtoa ja varmistamaan palvelimen aitouden.

iptables
iptables on työkalu, jolla hallitaan IPv4-verkkoliikenteen suodatussääntöjä Linux-järjestelmissä. Sitä käytetään palomuurin ja NAT-toimintojen toteuttamiseen.

ip6tables
ip6tables on vastaava työkalu kuin iptables, mutta se hallitsee IPv6-liikenteen suodatussääntöjä.

Intrusion Prevention System (IPS)
IPS on järjestelmä, joka tunnistaa ja estää haitallista verkkoliikennettä reaaliaikaisesti. Se voi automaattisesti torjua hyökkäyksiä ja ilmoittaa ylläpidolle.

Internet Protocol (IP)
IP on verkkoprotokolla, joka mahdollistaa tietoliikenteen laitteiden välillä verkossa. IP-osoitteet yksilöivät laitteet verkossa.

IP-osoite
IP-osoite on yksilöllinen numerosarja, jolla laite tunnistetaan tietoverkossa. Osoite voi olla IPv4- tai IPv6-muotoinen.

IPv4
IPv4 on yleisin IP-protokollan versio, jossa osoitteet ovat 32-bittisiä ja esitetään neljänä pisteellä erotettuna lukuna (esim. 192.168.1.1).

IPv6
IPv6 on uudempi IP-protokollan versio, jossa osoitteet ovat 128-bittisiä ja mahdollistavat huomattavasti enemmän osoitteita kuin IPv4.

journald
journald on systemd:n lokipalvelu, joka kerää ja tallentaa järjestelmän ja palveluiden lokitietoja keskitetysti.

JavaScript
JavaScript on ohjelmointikieli, jota käytetään verkkosivujen dynaamisten toimintojen ja vuorovaikutteisuuden toteuttamiseen.

kernel
Kernel eli ydin on käyttöjärjestelmän keskeisin osa, joka hallitsee laitteistoa, muistia ja prosesseja. Linux-ytimen ympärille rakennetaan koko Linux-järjestelmä. (ks. Linux)

Kaksivaiheinen tunnistautuminen (2FA)
2FA on tunnistautumismenetelmä, jossa käyttäjän on annettava kaksi erillistä varmennetta, kuten salasana ja kertakäyttökoodi, kirjautuakseen palveluun.

Linux
Linux on avoimen lähdekoodin käyttöjärjestelmän ydin, jota käytetään monissa eri jakeluissa. Linux-jakelut ovat suosittuja palvelimissa, työpöydillä ja sulautetuissa järjestelmissä. (ks. GNU/LINUX)

Euroopan unionin osarahoittama

localhost
localhost viittaa paikalliseen tietokoneeseen, yleensä IP-osoitteella 127.0.0.1. Sitä käytetään testaamaan palveluita ilman ulkoista verkkoyhteyttä.

MAC-osoite
MAC-osoite (Media Access Control) on verkkolaitteen yksilöllinen fyysinen osoite, jota käytetään lähiverkoissa laitteiden tunnistamiseen.

MariaDB
MariaDB on MySQL-yhteensopiva avoimen lähdekoodin tietokantapalvelin, joka tarjoaa tehokkaan ja turvallisen tietokantaratkaisun esimerkiksi Wordpressille.

Monivaiheinen tunnistautuminen (MFA)
MFA laajentaa kaksivaiheista tunnistautumista vaatimalla useampia kuin kaksi varmennetta, kuten salasana, kertakäyttökoodi ja biometrinen tunniste.

MySQL
MySQL on suosittu avoimen lähdekoodin relaatiotietokantapalvelin, jota käytetään laajasti verkkosovelluksissa, kuten Wordpressissä.

nftables
nftables on uudempi ja monipuolisempi työkalu palomuurisääntöjen hallintaan Linuxissa, ja se korvaa vähitellen iptablesin.

Nginx
Nginx on kevyt ja tehokas HTTP- ja käänteisproxy-palvelin, jota käytetään usein korkean suorituskyvyn verkkopalvelimena.

OpenSSH
OpenSSH on avoimen lähdekoodin toteutus SSH-protokollasta, joka mahdollistaa turvallisen etäyhteyden ja tiedonsiirron palvelimelle.

paketinhallinta
Paketinhallinta on järjestelmä, jolla ohjelmistot asennetaan, päivitetään ja poistetaan keskitetysti. Debianissa käytetään apt-työkaluja paketinhallintaan.

Palomuuri
Palomuuri on ohjelmisto tai laite, joka rajoittaa ja valvoo verkkoliikennettä määriteltyjen sääntöjen perusteella. Palomuuri suojaa palvelinta luvattomalta liikenteeltä.

PHP
PHP on palvelinpuolen ohjelmointikieli, jota käytetään erityisesti verkkosovelluksissa ja dynaamisten sivujen tuottamiseen, kuten Wordpressissä.

PID
PID (Process ID) on prosessin yksilöllinen tunnistenumero käyttöjärjestelmässä, jonka avulla prosesseja voidaan hallita ja seurata.

portti
Portti on verkkoliikenteen päätepiste, jonka kautta palvelut ovat saavutettavissa. Esimerkiksi HTTP käyttää porttia 80 ja HTTPS porttia 443.

PuTTY
PuTTY on Windowsille tarkoitettu SSH- ja telnet-asiakasohjelma, jolla voi muodostaa etäyhteyksiä palvelimiin.

Red Hat
Red Hat on yrityskäyttöön suunnattu Linux-jakelu ja siihen liittyvä yritys, joka tarjoaa kaupallista tukea ja palveluita.

repositorio
Repositorio eli ohjelmistovarasto on palvelin, josta paketinhallinta hakee ohjelmia ja päivityksiä. Debianin repositoriot sisältävät laajan valikoiman ohjelmistoja.

root-käyttäjä
Root-käyttäjä on järjestelmänvalvojan tili, jolla on täydet oikeudet järjestelmässä. Root-oikeuksia tarvitaan esimerkiksi järjestelmän asetusten muuttamiseen.

RSA
RSA on yleinen julkisen avaimen salausalgoritmi, jota käytetään esimerkiksi SSH-avaimissa ja TLS/SSL-sertifikaateissa.

RDP
RDP (Remote Desktop Protocol) on Microsoftin kehittämä protokolla etätyöpöytäyhteyksien muodostamiseen.

SCP
SCP (Secure Copy Protocol) on tiedostonsiirtoprotokolla, joka käyttää SSH-yhteyttä turvalliseen tiedostojen siirtoon.

SELinux
SELinux (Security-Enhanced Linux) on Linuxin tietoturvamoduuli, joka rajoittaa ohjelmien oikeuksia sääntöjen avulla ja tarjoaa hienojakoista käyttöoikeuksien hallintaa. (ks. AppArmor)

sertifikaatti (verkkoliikenne)
Sertifikaatti on digitaalinen varmenne, jolla todennetaan palvelimen tai käyttäjän identiteetti verkossa. Sertifikaatit mahdollistavat salatun ja luotettavan yhteyden (ks. TLS/SSL, CA).

shell
Shell on komentotulkki, jonka avulla käyttäjä voi antaa komentoja käyttöjärjestelmälle ja ajaa skriptejä. Bash on yleisin shell Linuxissa.

skripti
Skripti on komentorivillä suoritettava tiedosto, joka automatisoi tehtäviä ja toimenpiteitä järjestelmässä.

SQL
SQL (Structured Query Language) on kyselykieli, jolla hallitaan ja käsitellään tietokantojen tietoja.

Euroopan unionin osarahoittama

SSH
SSH (Secure Shell) on salattu yhteysprotokolla, jota käytetään etähallintaan ja tiedonsiirtoon palvelimille.

SFTP
SFTP (SSH File Transfer Protocol) on SSH-yhteyden yli toimiva tiedostonsiirtoprotokolla, joka tarjoaa turvallisen tavan siirtää tiedostoja.

sudo (komento)
sudo on komento, jolla tavallinen käyttäjä voi suorittaa ohjelmia järjestelmänvalvojan oikeuksilla. Sudo vaatii yleensä käyttäjän salasanan.

systemd
systemd on Linuxin käynnistyspalvelu ja prosessinhallintaohjelma, joka vastaa palveluiden käynnistämisestä ja hallinnasta.

TAR
TAR on tiedostojen pakkaus- ja arkistointiohjelma, jota käytetään usein yhdessä GZIP:n kanssa suurten tiedostojen ja kansioiden siirtämiseen. (ks. ZIP)

TCP
TCP (Transmission Control Protocol) on yhteyspohjainen tiedonsiirtoprotokolla, joka varmistaa luotettavan ja järjestyksessä olevan tiedonsiirron verkossa.

TLS/SSL
TLS (Transport Layer Security) ja SSL (Secure Sockets Layer) ovat salausprotokollia, joita käytetään suojaamaan verkkoliikennettä ja varmistamaan osapuolten identiteetti.

UDP
UDP (User Datagram Protocol) on yhteydetön tiedonsiirtoprotokolla, jota käytetään nopeaan mutta epäluotettavaan tiedonsiirtoon, kuten suoratoistossa.

Ubuntu
Ubuntu on suosittu Debian-pohjainen Linux-jakelu, joka tunnetaan helppokäyttöisyydestään ja laajasta yhteisötuesta.

UFW
UFW (Uncomplicated Firewall) on helppokäyttöinen palomuurin hallintatyökalu Ubuntu- ja Debian-järjestelmissä, joka yksinkertaistaa iptablesin käyttöä. (ks. GUFW)

UUID
UUID (Universally Unique Identifier) on 128-bittinen tunniste, jota käytetään yksilöimään laitteita, levyjä ja muita resursseja järjestelmässä.

wget (komento)
wget on komentorivityökalu, jolla voi ladata tiedostoja verkosta HTTP-, HTTPS- ja FTP-protokollilla.

Wordpress
Wordpress on avoimen lähdekoodin sisällönhallintajärjestelmä (CMS), jolla voi rakentaa ja ylläpitää verkkosivuja ja blogeja helposti.

Wordpress-plugin
Wordpress-plugin on laajennusosa, jolla Wordpressin toiminnallisuutta voidaan laajentaa esimerkiksi uusilla ominaisuuksilla tai integraatioilla.

XML
XML (eXtensible Markup Language) on merkintäkieli, jota käytetään tietojen jäsentämiseen, siirtämiseen ja tallentamiseen rakenteisessa muodossa.

ZIP
ZIP on suosittu tiedostojen pakkausmuoto ja -ohjelma, jolla voidaan pakata ja purkaa useita tiedostoja yhteen arkistoon. (ks. TAR)

Euroopan unionin osarahoittama