SSL/TLS

- SSL = Secure Sockets Layer, TLS = Transport Layer Security. Periaatteessa TLS on uusi versio SSL:stä, käytännössä termiä ”SSL” käytetään geneerisenä molemmista.
- Salakirjoitusprotokolla, perustuu ns. julkisen avaimen salakirjoitukseen (salaukseen käytetään myös symmetristä ts. jaetun salaisen avaimen salakirjoitusta).
- Kaksi funktiota: autentikointi ja salaus.

SSL/TLS

- Käytetään sekä palvelimen aitouden vermistamiseen että (harvemmin) toisinpäin, käyttäjän autentikoimiseen palvelimelle.
- Itse allekirjoitettua sertifikaattia voi käyttää omien koneidensa välillä, ja muutenkin pelkkään salaukseen (ilman autentikointia), mutta nykyisin selaimet valittavat aggressiivisesti sertifikaateista, joita eivät tunnista.

SSL/TLS

- Kaupalliset sertifikaatit maksavat n. 10€/vuosi tai enemmän, edellyttävät omassa hallinnassa olevaa domainia
- "Oikean" sertifikaatin voi saada ilmaiseksikin, ks. esim. https://letsencrypt.org
- Sertifikaatti voi olla yhdelle tai useammalle host-nimelle tai wildcard koko (ali)domainille (*.jyu.fi).

Sertifikaatit

- Sertifikaatti sisältää
  - tunnistetietoja, erityisesti DN (Distinguished Name, tässä yhteydessä domain-nimi kuten ”kone.example.org”), erilaisia osoitetietoja mukaanlukien yhteyssähköpostiosoite allekirjoitettuna sertifikaatin haltijan salaisella avaimella
  - haltijansa *julkisen avaimen*, jota vastaavalla salaisella avaimella haltija voi todistaa sertifikaatin omakseen
  - sertifikaatin myöntäjän salaisella avaimella tehdyn allekirjoituksen, jonka voi tarkistaa vastaavalla julkisella avaimella

Sertifikaatit

- Ottaessaan yhteyden suojattuun palvelimeen asiakas (selain tms) ensin tarkistaa sertifikaatin aitouden hallussaan olevalla myöntäjän julkisella avaimella ja sitten varmistaa palvelimen aitouden sertifikaatissa olevalla haltijan julkisella avaimella.
- Vastaavasti asiakassertifikaattia käytettäessä palvelin tarkistaa sen hallussaan olevalla julkisella avaimella

Sertifikaatit

- Autentikointi siis edellyttää, että käyttäjä (selain, sähköpostiohjelma tms) tuntee entuudestaan julkisen avaimen, jolla palvelimen sertifikaatti (avain) on allekirjoitettu (tai avaimen, jolla on allekirjoitettu avain, jolla sen on allekirjoitettu, joskus ketju voi olla pitkäkin). Sertifikaattien myöntäjien pitää siis saada oma julkinen avaimensa kaikkiin (yleisiin) selaimiin, mikä yleensä maksaa.

Sertifikaatin muodostus

- Sertifikaatti luodaan seuraavasti:

  (1) Hakija luo itselleen avainparin (julkisen ja salaisen avaimen) ja

  (2) sertifikaattipyynnön (Certificate Signing Request), joka sisältää hakijan tunnistetiedot sekä julkisen avaimen allekirjoitettuna salaisella avaimella, ja

  (3) sertifikaatin myöntäjä (tarkistettuaan enemmän tai vähemmän luotettavasti hakijan tiedot) allekirjoittaa CSR:n omalla salaisella avaimellaan ja luo siten siitä sertifikaatin (CRT) ja toimittaa sen hakijalle.

Sertifikaatin muodostus

- Sertifikaatin käyttö siis edellyttää vastaavan salaisen avaimen hallussapitoa. Jos salainen avain kaapataan, kaappari voi esiintyä sen haltijana kunnes sertifikaatti *revokoidaan* eli tehdään mitättömäksi: myöntäjä ylläpitää listaa revokoiduista avaimista, jota käyttäjien (ohjelmien) pitäisi seurata. Käytännössä kaapattua avainta voi usein käyttää kunnes se vanhenee (hyvä syy sertifikaattien määräaikaisuudelle).

Sertifikaatin muodostus

- Palvelimellakin tarvitaan myöntäjän julkinen avain, nämä tulevat yleensä valmiina (Ubuntussa /etc/ssl/certs/*CA.pem) ja mahdollisesti (jos myöntäjä on jälleenmyyjä) ketjutiedosto (olennaisesti jälleenmyyjän avain, jonka on allekirjoittanut ”tukkukauppias”; ketju voi olla pitempikin).

CSR:n luonti

- Avainten ja sertifikaattien hallintaan käytetään komentorivityökalua **openssl**
- Oma avainpari luodaan tähän tapaan:

  openssl genpkey -algorithm RSA -out oma.pem -aes-256-cbc -pkeyopt rsa_keygen_bits:4096

- Syntyneen avainparin salaisen avaimen (molemmat avaimet ovat samassa tiedostossa) käyttö edellyttää aina sen passphrasen syöttämistä. Jos sitä ei haluta syöttää aina kun www-palvelin käynnistyy, luodaan siitä suojaamaton versio:

  openssl rsa -in oma.pem -out oma.key

CSR:n luonti

- Syntynyt tiedosto sisältää sekä salaisen että julkisen avaimen. Sillä voi nyt luoda CSR:n tähän tapaan:

  ```
  openssl req -new -key oma.key -out oma.csr
  ```

- CSR lähetetään sertifikaatin myöntäjälle (certificate authority), joka palauttaa sitä vastaavan sertifikaatin (esim. oma.crt).
- Tiedostojen nimikonventiot ja muukin terminologia vaihtelevat, erityisesti *.pem ja *.key voivat sisältää milloin mitäkin.
- Sertifikaattien myöntäjillä on yleensä myös web-työkaluja sertifikaattien käsittelyyn.

Käärmeöljyä

- Testaamiseen ja omien koneiden välisiin yhteyksiin voi käyttää itse allekirjoitettua sertifikaattia. Se tapahtuu yksinkertaisesti allekirjoittamalla luotu CSR omalla avaimella:

  ```
  openssl x509 -req -days 365 -in oma.csr -signkey oma.key -out oma.crt
  ```

- Allekirjoitusta varten voisi myös luoda eri avaimen (esim. yrityksen oman, jota vastaava julkinen avain sitten talletettaisiin työntekijöiden koneisiin)

Käärmeöljyä

- Kummassakin tapauksessa avain ja sertifikaatti talletetaan; paikka on periaatteessa vapaa, tässä Ubuntun oletus:

  cp oma.crt /etc/ssl/certs

  cp oma.key /etc/ssl/private/

  chown root:ssl-cert /etc/ssl/private/oma.key

  chmod u=rw,g=r,o= /etc/ssl/private/oma.key

- Ubuntun openssl-paketin asennus luo samalla itseallekirjoitetun ”snakeoil” -sertifikaatin tiedostoihin /etc/ssl/private/ssl-cert-snakeoil.key ja /etc/ssl/certs/ssl-cert-snakeoil.pem, joita voi myös käyttää testaukseen.

Letsencrypt

- letsencrypt.org on ilmainen sertifikaattipalvelu, jonka käyttö on yritetty tehdä mahdollisimman helpoksi sertifikaattien automaattista uusimista myöten.
- Käyttö edellyttää joko domainin nimipalveluun tai siellä olevan www-palvelimen juureen pääsyä; jälkimmäinen toimii vain jos nimipalvelu ei estä sitä CAA-tietueella (Certificate Authority Authorization), kuten mm. jyu.fi tekee.

https

- https = http SSL:n yli
- käyttää (oletuksena) porttia 443
- alunperin käytti vain IP:tä ts. virtual hostit eivät toimineet; sitä varten kehitettiin laajennus SNI (Server Name Indication), kaikki melkeinkin uudet selaimet tukevat sitä nykyisin
- asennus erilainen eri palvelinohjelmille, mutta perusaskeleet samat

https käyttöönotto

- hankitaan sertifikaatti ja asennetaan se ja vastaava salainen avain sopiviin paikkoihin
  - lisäksi voidaan tarvita allekirjoitusketjutiedosto
  - ohjelmasta riippuen em. tiedostoja voi joutua yhdistelemään eri tavoin
- konfiguroidaan palvelinohjelma käyttämään https:ää (usein saman tien pakko-ohjataan http-yhteydenotot https:ään)
- avataan asianomaisiin palomuureihin tarvittavat reiät
- testataan että kaikki toimii (mielellään usealla eri selaimella)
- laitetaan kalenteriin muistutus sertifikaatin vanhenemispäivästä (voimassaoloaika vaihtelee, useimmiten yksi vuosi), että muistetaan hakea uusi ajoissa (ellei automatisoitu)

lighttpd & https

- Lighttpd haluaa avaimen ja sertifikaatin samassa tiedostossa.
- Ubuntun mukana tulleella käärmeöljysertifikaatilla:

  lighty-enable-mod ssl

  cat /etc/ssl/private/ssl-cert-snakeoil.key /etc/ssl/certs/ssl-cert-snakeoil.pem >/etc/lighttpd/server.pem

  service lighttpd restart

- Edellä itsetehdyllä sertifikaatilla vastaavasti:

  cat /etc/ssl/private/oma.key /etc/ssl/certs/oma.crt >/etc/lighttpd/server.pem

lighttpd & https

- Haluttaessa https usealle virtual hostille eri sertifikaateilla (/etc/lighttpd/conf-available/... -tiedostoon):

```
$SERVER["socket"] == ":443" {
        ssl.pemfile = "/etc/lighttpd/server.pem"   # oletus
        $HTTP["host"] == ”tt1.student.it.jyu.fi" {
           ssl.pemfile = "/etc/lighttpd/ssl/tt1.student.it.jyu.fi.pem"
        }
        $HTTP["host"] == "s019.vm.it.jyu.fi" {
            ssl.pemfile = "/etc/lighttpd/s019.vm.it.jyu.fi.pem"
        }
    }
```

lighttpd & https

- Jos halutaan pakottaa kaikki liikenne käyttämään SSL:ää (suositeltavaa):

```
$HTTP["scheme"] = "http" {
  $HTTP["host"] =~ ".*" {
      url.redirect = (".*" => "https://%0$0")
  }
}
```

lighttpd & https

- Lisäoptioita tietoturvan parantamiseksi (esimerkki):

```
ssl.use-compression = "disable"
ssl.use-sslv2 = "disable"
ssl.use-sslv3 = "disable"
ssl.cipher-list = "EECDH+AESGCM:EDH+AESGCM:AES128+EECDH:AES128+EDH"
ssl.dh-file = "/etc/ssl/certs/dhparam.pem"
ssl.ec-curve = "secp384r1"
server.modules += ( "mod_setenv" )
$HTTP["scheme"] == "https" {
   setenv.add-response-header  = ( "Strict-Transport-Security" => "max-age=63072000; includeSubdomains; ")
}
```

lighttpd ja letsencrypt

- Oletuskonfiguraatiolla (lighty-enable-mod ssl):

  apt-add-repository ppa:certbot/certbot

  apt update; apt install certbot

  certbot certonly --webroot -d ties478.website

  cd /etc/letsencrypt/live/ties478.website/

  cat {cert,privkey}.pem >server.pem

  ln -sf $PWD/ties478.website/server.pem /etc/lighttpd

  service lighttpd restart

lighttpd ja letsencrypt

- Automaattinen päivitysskripti /etc/cron.d/certbot ei toimi lighttpd:n kanssa, koska sertifikaatit pitää yhdistää samaan tiedostoon. Korjataan esim. tähän tapaan (polut vaihtelevat):

  0 */12 * * * root test -x /usr/bin/certbot -a \! -d /run/systemd/system && perl -e 'sleep int(rand(3600))' && certbot -q renew && cat /etc/letsencrypt/live/ties478.website/{cert,privkey}.pem >/etc/lighttpd/server.pem

lighttpd ja letsencrypt

- Jos sertifikaatteja on (tai voi olla) useita ne voidaan käsitellä kaikki tähän tapaan:

  0 */12 * * * root test -x /usr/bin/certbot -a \! -d /run/systemd/system && perl -e 'sleep int(rand(3600))' && certbot -q renew && for d in /etc/letsencrypt/live/*/cert.pem; do cat $d ${d%cert.pem}privkey.pem >${d%cert.pem}server.pem; done

- Sertifikaattipolkujen pitää olla samat vastaavissa /etc/lighttpd/conf-enabled/* -määrittelytiedostoissa

nginx & https

- Nginx:n https-konfiguraatiossa tarvitaan minimissään seuraavat rivit server{} -blokkiin:

```
listen 443 ssl;
...
ssl on;
server_name example.org;
ssl_certificate /etc/nginx/ssl/example.crt;
ssl_certificate_key /etc/nginx/ssl/example.key;
```

- Siis sertifikaatti ja salainen avain eri tiedostoissa.

nginx & https

- Jos halutaan pakottaa kaikki liikenne käyttämään SSL:ää, laitetaan http-blokkiin (jossa on listen 80) sääntö tyyliin

  return 301 https://$host$request_uri;

  - certbot osaa tehdä tämän automaattisesti
- 80-portin voi myös sulkea kokonaan, mutta yleensä se ei ole hyvä idea

nginx & https

- Nginx:n https-konfiguraatiossa voi myös tehdä tietoturvasäätöjä (ei kattava lista):

```
ssl_session_timeout 5m;
ssl_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers ... ;
ssl_prefer_server_ciphers on;
```

- Erityisesti ssl_protocols -rivi on suositeltava (poistaa käytöstä vanhat, turvattomat protokollat)
- Syytä tarkistaa säännöllisesti uusien tietoturva-aukkojen varalta

nginx ja letsencrypt

apt-add-repository ppa:certbot/certbot

apt update

apt install python-certbot-nginx

certbot --nginx

- osaa yleensä muuttaa konfiguraation automaattisesti (kysyy halutaanko sitä), em. tietoturvasäädöt pitää lisätä käsin
- automaattipäivitys toimii ilman eri säätöä

https ja proxyt

- Proxy-konfiguraatiossa (reverse) proxyn pitää hoitaa SSL, proxyn ja taustapalvelinten liikennettä ei yleensä tarvitse salata. Maailman ja proxyn välinen liikenne siis SSL-suojattua portin 443 kautta, proxyn ja taustapalvelinten välinen suojaamatonta portin 80 kautta eikä taustapalvelinten välttämättä tarvitse tietää SSL:stä mitään.

https ja proxyt

- Jos taustapalvelimessa pyörivän ohjelman pitää tietää käyttävänsä SSL:ää (esim. WordPress), sille pitää välittää tieto siitä lisäheaderilla (miten, riippuu www-palvelinohjelmasta):

  X-Forwarded-Proto: "https" (vanha, standardoimaton)

  Forwarded: proto="https" (RFC 7239, ei kaikkialla tuettu)

  ja kertoa siitä ohjelmalle, esim. wp-config.php (WordPress):

```
if($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https'){
    $_SERVER['HTTPS'] = 'on';
    $_SERVER['SERVER_PORT'] = 443;
}
```

Levyjen salaus

- Voidaan salata koko levy, levyosio, tiedosto, LVM:n alla LV tai PV, RAID-pakka… ehkä yleisintä salata LV tai osio
- Usein salataan kaikki paitsi /boot; VM-alustakoneessa voi jättää käyttöjärjestelmän salaamatta ja salata vain virtuaalikoneiden levyimaget tai niitä sisältävät tiedostojärjestelmät
- Myös swap pitää salata jos sitä käytetään
- Perustyökalu cryptsetup, yleisin käyttötapa LUKS
- Salausavain komentoriviltä tai tiedostosta

cryptsetup

- Tehdään salattu /sala omana LV:nään:

  apt install cryptsetup

  lvcreate -n sala -L 100M vg_tt2

  cryptsetup luksFormat /dev/vg_tt2/sala

  cryptsetup luksOpen /dev/vg_tt2/sala sala_crypt

  mkfs -t ext4 /dev/mapper/sala_crypt

  mkdir /sala

  mount /dev/mapper/sala_crypt /sala

cryptsetup

- Salatun levyn "sulkeminen" käytön jälkeen:

  umount /sala

  cryptsetup luksClose sala_crypt # tai ...close...

- Salausavaimen lisäys, vaihto, poisto:

  cryptsetup luksAddKey [options] /dev/tt_vg/crypt

  cryptsetup luksChangeKey [--keyslot...] /dev/tt_vg/crypt

  cryptsetup luksRemoveKey [--keyslot...] /dev/tt_vg/crypt

  - viimeisen (ainoan) avaimen poisto tekee levystä pysyvästi lukukelvottoman!

cryptsetup

- Käyttöönoton helpottamiseksi jatkossa laitetaan tiedostoon /etc/crypttab rivi

  sala_crypt /dev/tt1_vg/sala none luks

  ja tiedostoon /etc/fstab

  /dev/mapper/sala_crypt /sala ext4 defaults 0 0
- Usein halutaan molempiin optio "noauto", että koneen saa bootattua ilmankin
- Salatun osion suurentaminen:

  lvextend -L+100M /dev/tt_vg1/sala # ei -r:ää!

  cryptsetup resize sala_crypt

  resize2fs /dev/mapper/sala_crypt

RAID

- RAID (Redundant Array of Inexpensive Disks) on mekanismi, jolla useita (halpoja) levyjä yhdistämällä parannetaan turvaa levyvaurioiden varalta ja/tai levyjärjestelmän suorituskykyä
- Virtuaalikoneissa RAIDia ei yleensä käytetä, virtuaalikonealustoissa kylläkin

RAID-tasot

- Useita tasoja, jotka eroavat vaurionsietokyvyn, nopeuden ja kapasiteetin suhteen, mm.

  RAID0: nopea, ei hukkaa levytilaa, ei suojaa mitään (yhden levyn särkyminen hukkaa kaiken)

  RAID1 eli peilaus, kaikki data joka levyllä, paras suoja, syö paljon tilaa, nopeuttaa vähän

  RAID5: nopea lukea mutta hidas kirjoittaa, kohtalainen suoja (kestää yhden levyn särkymisen), vie vähän tilaa (yhden levyn per pakka)

  RAID6: nopea lukea, hidas kirjoittaa, hyvä suoja (kestää kahden levyn särkymisen), vie kohtalaisesti tilaa (kaksi levyä per pakka)

  RAID10 (tai 1+0): nopea sekä lukea että kirjoittaa, hyvä suoja (jopa puolet levyistä saa särkyä), vie paljon tilaa (puolet levyistä)

RAID

- Toteutus voi olla levyohjainkortin firmwaressa (”rautaraid”) tai käyttöjärjestelmätasolla (”softaraid”)
- LVM sisältää softaraid-toiminnallisuutta, mutta rajoitetusti eikä sitä usein Linux-ympäristöissä käytetä
- Linuxin standardisoftaraid nykyisin on md, komentorivityökalu mdadm

RAID

- md:n alla olevat levyt voivat olla myös levypartitioita eivätkä vain kokonaisia levyjä
- md:n käytön voi todeta (pseudo)tiedostosta /proc/mdstat ja sen (bootti)konfiguraation tiedostosta /etc/mdadm/mdadm.conf
- Boottaaminen onnistuu md-pakalta ainakin raid1:n kanssa, edellyttää huolellisuutta grub-konfiguraation kanssa
- md:n alla levyt voi vaihtaa isompiinkin ja ottaa koko tilan käyttöön lennosta (ei yleensä onnistu rautaraidilla)

RAID + LVM + salaus

- LVM:n kanssa yleensä tehdään RAID-pakoista PV:tä. Salaus voidaan tässä kuviossa tehdä monessa eri kohdassa:
  - Salataan fyysiset levyt RAIDin alla
  - Salataan RAID-pakat ja tehdään sen päälle PV:t
  - Salataan PV:t ja niiden päälle VG
  - Salataan LV salaamattoman VG:n sisällä
- Yleensä viimeksimainittu on paras (ja ensinmainittu yleensä järjetön)

identd

- ”identity daemon”: kertoo kuka (käyttäjä) tulee tietystä portista
- Käyttää tcp-porttia 113, avattava palomuurissa, esim.

```
iptables -A INPUT -p tcp --dport 113 -j ACCEPT
```

- Vanha, ei nykyisin kovinkaan hyödyllinen tietoturvamekanismina, käyttäjätietoon ei voi luottaa (saattaa paljastaa osoitekaappauksia joskus, erityisesti jos identd vastaa mutta ei kerro käyttäjää (tcp-wrapperissa UNKNOWN@host) on syytä epäillä IP-spoofausta)

identd

- Nykyisin oikeasti tarpeen lähinnä IRC-käyttäjille, voi olla käyttökelpoinen omassa verkossa tcp wrapperin kanssa muutenkin
- Useita identd-palvelinohjelmia, erityisesti:
  - nullidentd: triviaalitoteutus, ei kerro oikeita käyttäjätunnuksia vaan vakion tai satunnaisen
  - oidentd: yleinen, hyvin ylläpidetty, monipuolinen (käyttäjien voi antaa säätää omaa vastaustaan), konfiguraatiotiedosto /etc/oidentd.conf

inetd

- inetd (”Internet super server daemon”) on ohjelma (demoni), joka kuuntelee useita tcp- ja udp-portteja ja liikennettä havaitessaan käynnistää ko. portille määritellyn ohjelman (demonin).
- Säästää muistia jos ko. palvelua ei tarvita koko aikaa ja yksinkertaistaa niiden koodaamista, kun niiden ei itse tarvitse osata verkkoprotokollia vaan inetd välittää niille yhteyden stdin'in ja stdout'in kautta ja tarjoaa tcp wrapper -toiminnallisuuden myös.

inetd

- Useita toteutuksia, jotka tarjoavat useita saman perustoiminnallisuuden plus vaihtelevasti muuta, Linux-ympäristöissä ehkä yleisin xinetd. (Alkuperäinen ”inetd” on lähes kadonnut, sanaa käytetään yleisterminä sen toiminnallisuudelle.)
- Monet yleiset demonit voi konfiguroida toimimaan joko itsenäisinä tai inetd'n kautta. Useat inetd-toteutukset tarjoavat joitakin palveluita sisäisestikin (tyypillisesti ainakin echo ja discard).
- inetd on käymässä harvinaiseksi muistin halventuessa

tcp wrapper

- TCP Wrapper on yleinen mekanismi (kirjasto), jolla sisääntulevia yhteyksiä voidaan kontrolloida (nimestä huolimatta se voi toimia UDP-palveluidenkin kanssa).
- Toiminnallisuus samankaltainen kuin iptables input filter, mutta toimii ylemmällä tasolla (application layer) ja voi suodattaa sovelluskohtaisesti säännöillä, joita pakettifiltteri ei näe (esim. käyttäjäkohtaisesti, myös salattuja yhteyksiä jne).
- Toiminta edellyttää, että ao. sovellus käyttää inetd:tä tai siinä itsessään on libwrap-kirjasto mukana. Yleisten palveluiden kuten ssh ja ftp kanssa se toimii.

tcp wrapper

- Joskus samat (konekohtaiset) rajoitukset tehdään sekä pakettifiltterillä että tcp wrapperilla: jos yhteys ei toimi, tarkista molemmat.
- Etäkäyttäjän tunnistamiseen käytetään identd:tä, joka ei useinkaan ole käytettävissä, ja silloin käyttäjäkohtaiset rajoitukset eivät toimi.
- Konfiguraatiotiedostojen muuttaminen vaikuttaa heti, niitä ei tarvitse erikseen ladata eikä sovelluksia tarvitse käynnistää uudelleen.
- Ei kannattane enää uusissa asennuksissa käyttää

tcp wrapper

TCP Wrapperin säännöt kirjoitetaan tiedostoihin /etc/hosts.allow ja /etc/hosts.deny. Ensin luetaan hosts.allow, jos sieltä löytyy salliva sääntö yhteys sallitaan saman tien, ellei, luetaan hosts.deny ja jos ei löydy kieltävää sääntöä yhteys taas sallitaan.

- Molempien tiedostojen syntaksi on samanlainen:

  daemon_list : *client_list* [: *shell_command*]

  - *daemon_list* määrää palvelut joita sääntö koskee sitä ohjaavan demoniprosessin nimellä kuten ps sen näyttää, esim. sshd
  - *client_list* tarkoittaa koneita tai verkkoalueita, joita sääntö koskee; käyttäjäkohtainen sääntö voidaan tehdä syntaksilla *user@host*
  - *shell_command* on komento, joka suoritetaan täsmäävälle säännölle

tcp_wrapper

- Jos halutaan sallia kaikki mitä ei ole erikseen kielletty (blacklisting): ei lainkaan hosts.allow -tiedostoa ja kiellot hosts.deny -tiedostoon, esim.

  sshd: 134.170. ALL@UNKNOWN

- Jos halutaan kieltää kaikki mitä ei ole erikseen sallittu (whitelisting), tehdään hosts.deny yksinkertaisesti näin:

  ALL: ALL

  ja luetellaan sallitut hosts.allow -tiedostossa:

  ALL: 127.0.0.1

  vsftpd tftpd: ALL EXCEPT 134.170.

  sshd: 130.234. 172.20.

  sshd: .tapanitarvainen.fi