WHOIS

- Globaali tietokanta, jossa on domainien haltijoiden yhteystiedot, ja protokolla niiden hakemiseen.
- Nimipalvelurekisterien ylläpitämä, luotettavuus vaihtelee, usein julkinen tieto on proxy.
- Komentoriviclient ”whois”, esim:

```
$ whois jyu.fi
domain:   jyu.fi
descr:    Jyväskylän yliopisto
…
```

- Aina whois ei automaattisesti löydä oikeaa palvelinta. Tällöin sitä voi joutua kutsumaan useaan kertaan: kunkin top-level domainin whois-palvelimen pitäisi aina löytyä IANA:n palvelimelta, ja sieltä edelleen ao. TLD:n domainit:

```
$ whois -h whois.iana.org fi
…
whois: whois.fi
$ whois -h whois.fi jyu.fi
```

- Whois-järjestelmän tulevaisuus on epävarma, muuttunee tai ehkä korvataan kokonaan eri järjestelmällä.

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.
- Samaa tekniikkaa voidaan käyttää myös 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.
- Kaupalliset sertifikaatit maksavat n. 10€/vuosi tai enemmän, ei-kaupallisiin tarkoituksiin saattaa saada ilmaiseksikin (esim. startssl.com, edellyttää omaa domainia); ks. myös https://letsencrypt.org
- Sertifikaatti voi olla yhdelle tai useammalle host-nimelle tai wildcard koko 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
- 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.
- 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 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).
- 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 (genpkey'n asemesta käytetään usein yhä vanhempaa genrsa -komentoa, silloin optiot menevät hieman toisin):

  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
- 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
- Syntynyt CSR lähetetään sertifikaatin myöntäjälle (certificate authority), joka palauttaa sitä vastaavan sertifikaatin (esim. oma.crt).
- Huom. 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)
- Kummassakin tapauksessa avain ja sertifikaatti talletetaan (paikka periaatteessa vapaa, tässä Ubuntun oletushakemistot):

  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.

https

- https = http SSL:n yli
- käyttää (oletuksena) porttia 443
- normaalisti käyttää vain IP:tä ts. virtual hostit eivät toimi; sitä varten on kehitetty laajennus SNI (Server Name Indication), mutta se ei toimi ihan kaikilla selaimilla
- asennus erilainen eri palvelinohjelmille, mutta perusaskeleet samat:
  - hankitaan sertifikaatti edelläkuvattuun tapaan 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!

lighttpd & https

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

  ln -s ../conf-available/10-ssl.conf /etc/lighttpd/conf-enabled

  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
- Haluttaessa https usealle virtual hostille (/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"
        }
    }
```

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;
```

- Nyt siis sertifikaatti ja salainen avain pidetään eri tiedostoissa.
- Nginx:n https-konfiguraatiossa on enemmän säätövaraa, mm.

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

- Erityisesti tuo ssl_protocols -rivi on suositeltava (poistaa vanhat, turvattomat protokollat)

inodes

- Jos kone väittää levyn olevan täynnä vaikka siellä df:n mielestä on tilaa, syy voi olla että inodet ovat lopussa. Erityisesti ext* -tiedostojärjestelmissä niiden suhde levytilaan kiinnitetään tiedostojärjestelmää luotaessa eikä sitä voi kasvattaa. Tilanteen voi tarkistaa df -i:llä.
- Inode on olennaisesti osoitin tiedostoon, niiden (käytössä olevien) määrä on suunnilleen sama kuin tiedostojen määrä.
- Inode-määrään voi vaikuttaa luontivaiheessa mkfs:n optiolla -i, bytes-per-inode. Oletus ext4:ssä on noin 16384, jos tiedossa on, että pieniä tiedostoja tulee paljon, voi käyttää pienempää arvoa.
- Etenkin /usr:ään tulee usein paljon pieniä tiedostoja, jolloin inodet saattavat loppua. Koska niitä ei voi lisätä vaikka tiedostojärjestelmää kasvattaa, ratkaisuksi jää joko uuden tiedostojärjestelmän luonti ja tiedostojen siirtäminen sinne, tai tiedostojärjestelmän jakaminen osiin.
- Esimerkki: selvitetään missä /usr:n alihakemistoissa inodeja kuluu eli tiedostoja on paljon:

  ```
  cd /usr; for d in *;do echo -n ”$d: ”; find $d | wc -l; done | sort -nr -k2
  ```

  Suurimmat inode-syöpöt voi sitten siirtää omiksi tiedostojärjestelmikseen (ja varata niihin enemmän inodeja).

tune2fs

- Tiedostojärjestelmien ominaisuuksia voi joskus olla tarpeen tutkia tai muuttaa. Välineet riippuvat tiedostojärjestelmätyypistä, historiallinen komento on tunefs, nykyisille ext* -tiedostojärjestelmille **tune2fs**.
- Kaikki säädettävissä olevat asetukset ja paljon muutakin saa näkyviin komennolla **tune2fs -l** (hyvä tehdä ennen kuin muuttaa mitään).
- Tavu/inode -suhteen saa laskemalla tune2fs -l:n tulostuksesta

  (Blocks per group)/(Inodes per group)*(Block size)
- Muita joskus hyödyllisiä optioita:

  -m *rootille-varattu-prosenttiosuus*

  -L *volume-label* # aseta nimiö

  -U *UUID* # aseta UUID

  -O +large_file # salli yli 2GB tiedostot (kiellä ^large_file)

  -O +huge_file # salli yli 2TB tiedostot (kiellä ^huge_file)

sudo

- sudo -komennolla voi suorittaa komentoja jonkun toisen käyttäjän (yleensä mutta ei aina rootin) oikeuksilla.
- Oikeusmäärittelyt tehdään /etc/sudoers -tiedostossa, nykyisin siihen usein sisällytetään kaikki /etc/sudoers.d -hakemistossa olevat tiedostot (lokaalit muutokset on helppo tehdä sinne).
- /etc/sudoers -tiedoston editointiin on oma komentonsa, visudo (kutsuu EDITOR-muuttujan määrittelemää editoria).
- sudoers-tiedoston perussyntaksi on:

  kuka missä = (kenenä) mitä

  - *kuka* = käyttäjä tai %ryhmä
  - *missä* = kone (usein ALL)
  - *kenenä* = minkä käyttäjän oikeuksilla komento suoritetaan (id:gid, usein ALL tai ALL:ALL)
  - *mitä* = komento jonka suoritus on sallittu (usein ALL)
  - lisämääreellä NOPASSWD: voidaan salasanan kysely jättää pois
- Esim.

  ```
  %sudo   ALL=(ALL:ALL) ALL
  %libvirtd ALL=(root) /usr/bin/vmbuilder
  tt ALL=(ALL) NOPASSWD: ALL
  ```
- Komentojen rajaus vaikea tehdä hyvin, monista komennoista pääsee suorittamaan jopa shellin; usein turvallisempi vaihtoehto on ns. suid wrapper (pieni suid-ohjelma, joka suorittaa halutun komennon eikä muuta).