E-közigazgatási Szabad
Szoftver Kompetencia Központ

Mátó Péter, Rózsár Gábor, Őry Máté,
Varga Csaba Sándor, Zahemszky Gábor

20/80

Hálózati szolgáltatások szabad szoftverekkel

ÚJ SZÉCHENYI TERV

20/80 – Hálózati szolgáltatások szabad szoftverekkel

Módszertan és tartalmi tervek: Mátó Péter, Varga Csaba Sándor, Zahemszky Gábor

Írták: Mátó Péter, Rózsár Gábor, Őry Máté, Varga Csaba Sándor, Zahemszky Gábor:

0.9-es változat, 2014. április 11., elektronikus kiadás

Kiadó: E-közigazgatási Szabad Szoftver Kompetencia Központ

Honlap, javított kiadások: http://szabadszoftver.kormany.hu/sajat-oktatasi-anyagok/.

ISBN 978-963-08-8300-9

Szerzői jog

Tartalomjegyzék

Előszócska

Ezzel a könyvvel és testvéreivel az a célunk, hogy viszonylag tömören összefoglaljuk azokat az információkat, amiket egy szabad szoftvereket használó szakembereknek tudnia illik.

20/80. Mit akar ez jelenteni? Tapasztalatunk szerint a létező eszközöknek és információknak csak egy kis része szükséges a mindennapok tipikus feladatainál. Igyekeztünk kiválogatni nektek a tudásnak azt a 20%-át, ami az általában előforduló feladatok 80%-ánál elegendő lesz. Célunk ezen elv alapján összeszedni, rendszerezni és átadni a leghasznosabb dolgokat. Hiába próbálnánk mindent elmondani – nekünk nincs időnk mindent leírni, nektek meg nincs időtök elolvasni. Ezért sok minden kimarad. Ha úgy gondolod, hogy fontos, kimaradt vagy bővebben kellene beszélni róla, szólj! Ha valami hibás, szólj! E-mail címünk: esz2k2@gmail.com. De ha írsz, légy türelemmel, valószínűleg 200 másik levél is vár még megválaszolásra. A továbbfejlesztés során minden konstruktív javaslatot igyekszünk majd az anyagba építeni.

A tárgyalt megoldások és szabad szoftverek legtöbbször több operációs rendszer alatt is használhatóak. Amikor viszont operációs rendszer szintről esik szó (telepítés, csomagkezelés vagy finomhangolás), akkor ez most – népszerűsége miatt – nálunk Linuxot jelent.

A könyvben időnként kérdéseket teszünk fel, de néha nyitva hagyjuk a választ. A cél: gondolkozz, olvass utána, használd az agyadat! Ha egy témát alaposabban meg akarsz ismerni, akkor nincs mese, alaposabban utána kell olvasnod. Minden területnek megvannak a maga – tőlünk sokkal mélyebb ismereteket tárgyaló – szakkönyvei, előttük azért érdemes a mi összefoglalónkat elolvasni, mert ezekben – reményeink szerint – az adott terület esszenciája található. Ez alapján már könnyebben eligazodsz majd a 6-700 oldalas, lényegesen kimerítőbb anyagokban is.

Vállalati levelezés, spamszűrés és vírusszűrés

Történeti áttekintés

Az első generációs levelezőszerverek a nagygépes UNIX és már a linuxos idők kezdetén a Sendmail rendszert preferálták legtöbbször. A Sendmail a maga tekintetében lerakta az RFC szerinti alapokat az 1980-as évek elején, és gyakorlatilag iparági standarddá vált. A maga nemében hihetetlen rugalmassággal rendelkezett, amelyet M4 makrónyelven lehetett programozni. A Sendmailt ma is fejlesztik, és sokan használják, mégis egyeduralkodása a 2000-es évek elején megtört, és számos úgynevezett Sendmail kompatibilis levelezőrendszer készült. Jelenleg széles körben használt rendszerek:

– Postfix[1]
– Exim4[2]
– qmail[3]
– Sendmail[4]

A kis- és középvállalatok körében leginkább elterjedtnek tekinthető Linux terjesztések, mint pl. a Debian[5], Ubuntu[6] (vagy akár a kereskedelmi RHEL[7] és SLES[8]) is ezt a négy levelezőrendszert csomagolják könnyen elérhetően a kiadásaikba. Nagyon fontos tény, hogy a biztonsági frissítések is hamarabb, illetve rendszeresebben érkeznek az olyan szoftverekhez, amelyek használata szélesebb körű, hiszen ezeket gyakran nagyobb programozói csapat és figyelem kíséri.

1. http://www.postfix.org/
2. http://www.exim.org/
3. http://www.qmail.org/ – az eredeti fejlesztője következetesen kisbetűvel írja a nevet, ezért használjuk mi is így
4. http://www.sendmail.com
5. http://www.debian.org/
6. http://www.ubuntu.com/
7. http://www.redhat.com/
8. http://www.suse.com/

Kiválasztás

A megfelelő MTA[9] kiválasztásánál a következő szempontokat vegyük figyelembe. A jelenleg legnagyobb felhasználói bázissal rendelkező, és leginkább támogatott (alapítványok és közösségek által egyaránt), terjesztésekben melyek integráltak, és a rájuk épített járulékos szolgáltatások melyekkel a legkönnyebben integrálhatóak. Azaz a későbbiekben taglalt vírusvédelem, spamszűrés, illetve levelezőlista-kezelés melyikkel a legkönnyebben kivitelezhető – és természetesen a biztonság is jelentős szempont.

Sendmail

Összetett rendszer, amely igen robusztus felépítésű. Ellenérvként hozható fel, hogy az átlagostól eltérő igényekhez az M4 makrók ismerete szükséges, melynek elsajátítása komoly programozói ambíciót igényel. Valószínűleg mindent meg lehet csinálni vele, ami levelezésben egyáltalán előfordulhat, de egyre kevesebben szánják rá az időt a megtanulására. Mindenesetre akinek esetleg nem csak Linuxban, hanem zárt Unix rendszerekben is kell gondolkodnia, annak lehet fontos ezt (is) megismerni.

Exim4

Gyors, összetett, komplex rendszer. Minden igényre könnyedén alkalmazható, remekül bírja a nagy forgalmat. Igazi előnytelen tulajdonsága nincs. Nem véletlen volt alapértelmezett levelezőszerver a Debian rendszerű terjesztésekben.

qmail

Gyors és kis program, alacsony hardverigényű, még akkor is jól teljesít, ha nagy az átmenő forgalom, rendkívül biztonságos, a konfigurálhatósága jobb a Sendmailnél. Ellenérvként hozható fel vele szemben, hogy a bonyolultabb igényeket nehezebb megoldani qmail alatt, mint Exim vagy Postfix alatt. További ellenérv lehet a licencelése, amely közkincs (public domain)[10], valamint hogy eredeti fejlesztője a qmail aktív fejlesztésével leállt.

Postfix

Készítője Wietse Venema[11], aki főként biztonsági szemléletéről vált ismertté a Postfixet megelőző időkben. Ez rögtön szembetűnik, amint áttekintjük a Postfix felépítését, amely szemben a hagyományos MTA-kkal, nem egy programon belül látja el az összes funkciót, hanem egy piramis felépítésére hasonlító moduláris programrendszer valósítja meg a funkciókat. A beérkező leveleket az első réteg fogadja és adja át a megfelelő funkciót ellátó következő komponensnek, a beállított szabályok szerint. Ez a felépítés az, amely nagyban hozzájárul a magas fokú biztonsághoz. A hibajavítás és a biztonsági frissítések elkészítése is jóval egyszerűbb és áttekinthetőbb. További hatalmas előnye, hogy mivel az élmezőnyben szerepel, azaz több millió felhasználóval rendelkezik,

9. http://en.wikipedia.org/wiki/Message_transfer_agent
10. http://en.wikipedia.org/wiki/Qmail
11. http://www.porcupine.org/wietse/

ezért dokumentációs szempontból uralja az internet fórumait és számtalan karbantartott „hogyan” található hozzá.[12]

Natív támogatással rendelkezik a következő operációs rendszerekhez: AIX, a különböző BSD-k, HP-UX, IRIX, GNU/Linux, Mac OS X, Solaris, Tru64 UNIX, de minden olyan UNIX-jellegű rendszeren futtatható, amelyen C fordítóval forrásból előállítható és ahol POSIX futtatási környezet létrehozható. A legtöbb Linux-terjesztés alá natív módon a csomagkezelővel telepíthető. Ha az nem jó, a Postfix forrásból való installálását a dokumentáció mutatja be.[13] Ez azonban csak abban az esetben javasolt, ha nem érhető el csomagban, vagy a használni kívánt kombináció nem érhető el a terjesztő által elkészített változatban, pl. olyan komponenseket akarunk használni, amelyek csak forrásból való fordítás után érhetők el.

Tudása: nem csak biztonsági szempontból figyelemre méltó moduláris felépítése, hanem ennek köszönhetően könnyen illeszthető adatbázishoz. Lehetőség van a felhasználók azonosítására a legtöbb SQL-szerver használata esetén (de akár a levelek is tárolhatók SQL-adatbázisban, ha valakinek ilyen extremitásra van igénye). Ugyanilyen könnyen kapcsolható az Amavis programcsomag segítségével a különböző spam- illetve vírusvédelmi megoldásokhoz is. A nagyon részletes funkciólista az interneten tekinthető meg.[14] Fontos további előnyt jelent, hogy a Postfix ugyanolyan jól megállja a helyét a kisvállalati környezetben, napi 100-200 levél forgalmazása esetén, mint nagyvállalati környezetben, ahol a vele szemben támasztott igény akár 80-150e átmenő e-mailtől indul. A Postfix rétegződéséről és felépítéséről itt[15] találhatunk információt. Két főbb konfigurációs fájllal rendelkezik: a master.cf határozza meg, hogy hogyan kapcsolódjanak egymáshoz a levelek küldéséhez, fogadásához, postaládába helyezéséhez vagy a különböző ellenőrzésekhez használt komponensei, a main.cf-ben pedig a Postfix paraméterezését találjuk. A későbbiekben vegyesen hol a main.cf-ben, hol pedig szükség esetén a master.cf-ben fogjuk állítani a szükséges dolgokat.

A Postfix fontosabb beállításai

Nézzük tehát, hogyan is lehet egy átlagos kis- és középvállalkozás számára használhatóvá tenni a Postfixet. A kiindulási alapunk, hogy már akár 50 felhasználó felett praktikus lehet összekapcsolni adatbázissal, de a konfigurációs példák tartalmazni fogják a normál felhasználókkal és adatbázis-kapcsolattal felépített használatot is (az általános rész után található meg), pontosan azért, hogy mindenki a saját ízlése és igényei szerint tudja használni.

Eredendően az MTA-k egy e-mail tartományba tartozó felhasználók levelezését kezelték, ekkor logikus volt, hogy a címzettek valódi, létező felhasználók legyenek a levelezőszerverként működő gépen. Ma persze már majdnem minden levelezőszerver nem csak egy, hanem több (több száz, ezer) domain levelezéséért felel, ezért logikusan a különböző virtuális domainekhez tartozó felhasználókat nem lehet az operációs rendszer felhasználóiként beállítani. (Hogyan kezeljek könnyedén két Kovács János nevű felhasználót?) Ezért ma a virtuális domainek virtuális felhasználóira vonatkozó információkat különböző adatbázisokban szokták tárolni, és a levelezőszerverek leggyakrabban onnan veszik elő az információkat.

A Postfix telepítése (amely Ubuntu és Debian rendszerekben az `apt-get install postfix` parancs kiadását jelenti) és egy párbeszédalapú minimális konfigurálás után az MTA üzemkész. Az alap-

12. http://www.postfix.org/docs.html
13. http://www.postfix.org/INSTALL.html
14. http://www.postfix.org/features.html
15. http://www.postfix.org/OVERVIEW.html

konfiguráció egy átlagos szerver, vagy hálózati átjáró esetén teljesen jól használható, természetesen csak az alapvető funkciókkal.

Alap telepítés esetén a következő beállításokat érdemes megadni. Kezdjük a beállításokat a /etc/postfix/main.cf állománnyal. Gyári beállítása Ubuntu 12.04 LTS esetén tartalmazza a következő sort:

```
smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
```

Ezzel mondhatjuk meg, hogy az SMTP banner mi legyen, azaz hogy a külvilág felé minek mutassuk magunkat. Célszerű ezt megváltoztatni, legalább az operációs rendszer típusát levenni. Bár sok esetben az nmap parancs OS finger funkciója is meg tudja mondani (tippelni), hogy egy (Ubuntu) Linuxot használunk, de a „Security by Obscurity" gondolkodásmód mentén haladva, elég ha mindenki csak annyit tud rólunk, amennyit feltétlen szükséges. Így ha ezt a változót $myhostname ESMTP $mail_name értéken hagyjuk, az éppen elégséges. (Az ESMTP szövegrész viszont szükséges, figyeljünk rá.)

```
myhostname = mail.cegnev.hu
```

Ide írjuk a tényleges nevét a gépnek. Illik figyelni, hogy – mint majdnem minden hálózatot használó szoftvernél – a használt hálózati nevek és IP-címek összerendelése korrekten (és konzisztensen) legyen megoldva.

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

A fenti paraméterek mondják meg, hogy milyen címzetteket fogad el a gép – azaz itt az elfogadható e-mail címeket sorolhatjuk fel. Tipikusan ilyenek az elterjedten használt webmaster, postmaster, hostmaster, stb. címek, és aliasból irányítják konkrét felhasználóhoz a leveleket. Az alias_maps lehet lokális állomány, vagy hálózaton keresztül elérhető (pl. SQL-lel, LDAP-protokollal elérhető adatbázis) is. Ha lokális állományt (is) használunk, akkor a helyi fájl (amiben felsoroljuk az aliasainkat) módosítása után adjuk ki a sudo /usr/bin/newaliases parancsot, ezzel készítünk belőle egy (a példa szerint) hash formátumú adatbázist, amely indexszel ellátott, ezzel gyorsítja a Postfix számára a keresést. (A newaliases parancs egyike a Sendmail-kompatibilis parancsoknak, használható helyette a Postfix saját postalias parancsa.) Lényeges különbség, hogy az alias_mapsben megadott helyeken keresi a Postfix az aliasok feloldását, míg az alias_database-ben adjuk meg, hogy melyek azok az adatbázisok, amelyeket módosítás után újra kell építeni a newaliases paranccsal. Ezek értelemszerűen lokális (hash, tree, dbm tipusú) fájlok, míg távoli – pl: NIS, SQL vagy LDAP elérésű – adatbázisnál erre nincs szükség – sőt lehetőség sem.[16],[17]

```
virtual_maps = hash:/etc/postfix/virtual
```

amennyiben nem csak egy, hanem több domainnek akarunk levelezést biztosítani, akkor a virtual állományban sorolhatjuk fel, hogy a Postfix milyen domain névre hallgasson még és a virtualban felsorolt e-mail címek mely felhasználóknak, vagy e-mail címeknek legyenek továbbítva. Egy ilyen virtuális tábla felépítése valahogy így néz ki:

```
felhasznalo1@masodikdomain.hu        username
postmaster@masodikdomain.hu          postmaster@sajatceg.hu
abuse@masodikdomain.hu               abuse@sajatceg.hu
felhasznalo2@masodikdomain.hu        felhasznalo1@cegnev.hu
```

16. http://readlist.com/lists/postfix.org/postfix-users/1/5058.html
17. http://readlist.com/lists/postfix.org/postfix-users/1/5060.html

```
@masodikdomain.hu                    summ-username[18]
```

A felhasznalo1 e-mail cím egy létező, a felhasználói adatbázisban szereplő személyhez tartozik – neki a bejelentkezési neve szerepel a jobb oldalon; felhasznalo2 pedig igazából egy másik domainbe tartozik, és erre a címre lesznek továbbítva a bejövő levelei. Ahogyan az alias esetében, a virtual esetében is szükség van az indexek elkészítésére és változás esetén a frissítésükre is, amelyet a következőképpen tudunk végrehajtani:

```
postmap /etc/postfix/virtual
```

fontos, hogy a virtual-ban elhelyezett új felhasználó csak a DB frissítése után lesz érvényes.

```
mail_spool_directory = /var/spool/mail
```

Ezzel a beállítással rögzítjük, hogy a felhasználók Inbox állományai hol tárolódjanak. Alapesetben érdemes így hagyni, hacsak lemeztelítettség, vagy I/O teljesítmény miatt nem akarjuk máshova helyezni.

```
mailbox_command = /usr/bin/procmail
```

A bejövő levelek postafiókba tételét a procmail programra bízzuk. Mivel a procmail nem része a Postfixnek, így ha ezt az opciót használjuk, akkor előtte telepíteni is kell a rendszerre. Utána viszont, mielőtt a felhasználó postfiókjába kerülne a levél, a procmail segítségével különböző feltételek alapján a bejövő leveleket különböző mappákba szortírozhatjuk, automatikusan törölhetjük, másik postafiókba továbbíthatjuk és még egy sereg egyéb trükköt megtehetünk.

```
mydestination =mail.cegnev.hu, localhost.localdomain, localhost
```

megmondhatjuk az SMTPD-nek (ennek a programnak a feladata a hálózaton keresztül, SMTP-protokollon keresztüli levélfogadás), hogy milyen domain neveket tekinthet saját magának, azaz mely címek esetén legyen helyi kézbesítés (local delivery).

```
relayhost = relayserver.cegnev.hu
```

Ezt az opciót csak akkor használjuk, ha rendelkezünk relay-szerverrel[19], és a használata célszerű vagy kötelező. Elsősorban kis cégek esetén javasolt, akik az internetszolgáltatójuk levelezőszerverén keresztül küldik a leveleket – és itt ezt a szervert kell megadni. Jellemzően egy átlagos céges felépítés esetén, ahol vagy co-locationben vagy DMZ-ben, vagy esetleg több hálózati csatoló esetén belső-külső lábon mi szolgáltatunk levelezést, ott mi vagyunk mások számra a relayhost, így ott ezt az opciót töröljük.

```
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128,!172.20.1.58,
  172.20.1.0/24
```

Meghatározhatjuk, hogy honnan fogadja el a küldendő leveleket. Azaz az itt rögzített hálózatok azonosítás (autentikáció) nélkül küldhetnek rajtunk keresztül leveleket. Éppen ezért érdemes kihasználni a Postfix rugalmas lehetőségeit, és csak akkor adjuk meg az egész IP-címtartományt, ha valóban minden cím a felügyeletünk alatt áll. Egyéb esetekben felsorolás jelleggel adjuk meg az IP-címeket. (A példából kikövetkeztethető, hogy balról jobbra értelmeződnek a bejegyzések, és a felkiáltójellel kizárhatok a – későbbi – listából elemeket.) Ha sok címet akarnánk megadni, az át-

18. Ide fog minden olyan levél megérkezni, amelyhez előtte nem definiáltunk címet, tehát az elcímzettek is mint pl: felhasznalo1@masodikdomain.hu helyett ha valaki a elhasznalo1@masodikdomain.hu -ra ír, akkor az a summ-username accountba fog megérkezni.
19. a relay-szerver olyan gép, amelyik mások helyett a levéltovábbítást végzi (jellemzően az internetszolgáltatók nyújtanak ilyen szolgáltatót a saját ügyfeleiknek); technikai megközelítéssel: relay az, aki átvesz olyan levelet, ami nem neki szól, és továbbítja a valódi címzettnek

láthatóság érdekében használhatunk itt is (az aliases-hez hasonló) MAP[20] állományt és felsorolhatjuk abban a címeket.

```
mailbox_size_limit = 51200000
message_size_limit = 10240000
```

Rögzíthetjük a rendszerben levő postafiókfájl méretét (byte-ban), illetve megmondhatjuk az egyes üzenetekre, hogy mekkora maximális méretű átmenő levelet fogadunk el. Érdemes ezeket az értékeket a végfelhasználók felé is jelezni, hogy a levelezőszolgáltatást ne tekintsék FTP-nek, valamint készüljenek fel arra, hogy a postafiókok mérete sem végtelen, azaz rendszeresen takarítani kell. (Nem túl régen találkoztunk felhasználóval, aki 70 MB-nál kicsit nagyobb levelet próbált küldeni, de a kapott hibaüzenet hatására sem tett le erről.)

```
inet_interfaces = all
```

Megmondhatjuk, hogy a Postfix mely hálózati csatolókon látszódjon, pl. több interface megléte esetén (tűzfal, DMZ,stb)

```
body_checks = regexp:/etc/postfix/body
```

Megadhatjuk, hogy a Postfix a levél tartalmában szabályos kifejezéseket[21] használva keressen, és ezek segítségével döntsön a levél elutasítása vagy beengedése mellett, illetve meghatározhatjuk, hogy milyen üzenettel utasítsa el pl. a nem megengedett kiterjesztésű fájlokat tartalmazó leveleket.

Egy példa a /etc/postfix/body tartalmára:

```
/^Content-(Disposition|Type).*name.*=.*"?(.*\.(bat|c[ho]m|cmd|exe|pif|scr|hta|
  jse?|vb[esx]|wav|mov|avi|mpe?g|mp3))"?$/ REJECT Some file extension in the
  attachment is not allowed
/^.*?name.*=.*"?(.*\.(bat|c[ho]m|cmd|exe|pif|scr|hta|jse?|vb[esx]|wav|mov|avi|
  mpe?g|mp3))"?$/ REJECT Some file extension in the attachment is not allowed
/^begin \d\d\d .*\.(vbe|vbs)/ REJECT
/kozossegiweboldalak.com/ REJECT Message content rejected.
```

A példák a feltételben megadott regexp-illeszkedés esetén utasítják el a levelet, a REJECT szó mögé írt indoklással.

Az első példa esetén ha a levélben valahol szerepel egy sor, ami a Content-Disposition vagy Content-Type szöveggel kezdődik, valahol egy „name” szócska követi (nagy eséllyel tehát valamely fájlnév hivatkozás), majd egy egyenlőségjel, ami után némi egyéb (meg nem határozott karakterek) után a .bat, .com, .chm, .cmd, .exe, (és így tovább) szövegek valamelyike következik, akkor a fájlkiterjesztés tiltására vonatkozó hibaüzenettel el kell utasítani a levelet. (Ilyen Content-Type jellegű sorokat jellemzően csatolmányok küldése esetén lehet találni a levélben.)

A második példa tulajdonképp ugyanez, csak még a Content-Type részt sem igényli.

A harmadik példa a ma már nem túl elterjedten használt UUEncode-kódolású fájlátküldés esetén is észreveszi ha gyanús (.vbe és .vbs) kiterjesztésű fájl van a levélben, és elutasítja azt.

20. http://www.postfix.org/postconf.5.html#mynetworks
21. A Postfix fordításakor beállítható, hogy a szabványos, POSIX-kompatibilis (az egrep parancs által használtnak megfelelő) regexp implementáció mellett lehessen-e a Perl-ből ismert un. PCRE-verziót is használni. Ha az rendelkezésre áll, akkor a regexp: helyett a pcre: írandó, és annak szintaxisa használható, ám ehhez külön támogatás telepítése szükséges Debian/Ubuntu alatt a Postfixhez.

Az utolsó példa pedig minden olyan levelet el fog utasítani, melyben bárhol szerepel a „kozos-segiportal.hu” szöveg. Ezzel könnyedén elérhető, hogy minden, az adott oldalra hivatkozó linket tartalmazó leveleket visszautasítsuk. (De még ha csak megemlítik a levélben, már az is elég.)

További reguláris kifejezéseket használó szűrési lehetőségek itt[22].

Postfix nagyobb helyen

Ahogyan a tűzfalaknál is elmondható, a levelező szervereknél is vannak közel egyenrangú megoldások (azaz ugyanaz a projekt megvalósítható Exim vagy Postfix segítségével is hasonlóképpen), de alapvetően az aktuális szituáció, az aktuális és a tervezhető jövő határozza meg, hogy hogyan érdemes felépíteni egy MTA-t. A fenti konfigurációt könnyen átalakíthatjuk, ha igény mutatkozik arra, hogy a felhasználókat valamilyen adatbázisból keressük ki. Ez lehet valamilyen hagyományos, SQL-nyelven kezelhető lokális vagy távoli adatbázis (kezdve a MariaDB-vel vagy MySQL-lel, folytatva a PostgreSQL-en át akár zárt adatbázis-kezelőkig), de komolyabb cég esetén akár LDAP-alapon elérhető címtár is. Címtár alatt nem csak az ismertebb AD-t kell érteni, hanem az alapvetően ugyanazt a funkciót nyújtó szabad szoftveres címtárakat (OpenLDAP, OpenDJ), illetve bármilyen egyéb akár zárt termékként elérhető címtárat. Ebben a dokumentumban a Postfix és a címtár használatát nem tárgyaljuk, akit érdekel, kezdetnek a Postfix dokumentációban[23] olvasson utána.

A MariaDB/MySQL és a Postfix kapcsolata

Mire is jó ez? Tipikusan az olyan esetekben, amikor tudható, hogy több száz, vagy ezer felhasználó fogja használni azonnal vagy akár csak belátható időn belül a rendszerünket, akkor érdemes átgondolni, és egy rugalmasabban alakítható alap rendszert létrehozni. Azaz később sokkal kevesebbet kell majd a skálázhatósággal foglalkozni, ha az elején olyan rendszert tervezünk, amely működik 1, de akár 10 000 vagy több felhasználóval is. Hiszen később már „csak” a hardverkörnyezetet kell hozzáigazítani. A Postfix támogatja az SQL adatbázisban tárolt felhasználókat, így most kifejezetten csak azokat a paramétereket taglaljuk ebben a részben, amelyeket az SQL-lel és kifejezetten a MySQL-lel/MariaDB-vel való összekötés kapcsán kell a telepítés utáni felálláshoz képest megváltoztatni:

Az alaptelepítést egészítsük ki a következőkkel:

```
apt-get install postfix-mysql mysql-client mysql-server
```

Ha MySQL helyett MariaDB-t használnánk, az adatbázis-fejezetben leírtak szerint telepítsük a MariaDB-t (apt forrás hozzáadásával) de ugyanúgy a postfix-mysql csomagot rakjuk fel (apt-get install postfix-mysql). (Jelenleg a MySQL és a MariaDB egymással olyan szinten kompatibilisek, hogy nincs is külön csomag a Postfix–MariaDB-kapcsolathoz.) Az SQL szerver beállításai között adjuk meg az adatbázis-adminisztrátor (hagyományosan: root) felhasználó választott jelszavát, amelyet a telepítő szkript kérni is fog. A jelszó legyen minimum 12-20 karakter hosszú, lehetőleg erre szolgáló eszközzel generált (pl: pwgen[24]).

22. http://www.jeffborders.com/techdocs/postfix/body_checks.html
23. http://www.postfix.org/LDAP_README.html
24. Mivel a legtöbb disztribúcióban elérhető parancsnak gyakorlatilag nincs saját honlapja, ezért az elvi

Ezt követően ki kell alakítani a felhasználók és a virtuális felhasználók számára az SQL adatbázist. Hozzunk létre egy adatbázist, amely stílszerűen mail névre hallgasson. (Persze bármi lehet, de ebben a példában ezt használjuk, ettől eltérő adatbázisnév esetén értelemszerűen a parancsokban a nevet megfelelően módosítsuk!) Példánkban a MySQL binárisát fogjuk hívni, de természetesen minden ugyanúgy működik MariaDB esetében is.

```
mysql -u root -p
```

Majd a MySQL shell-ben:

```
CREATE DATABASE mail;
  USE mail;
GRANT SELECT, INSERT, UPDATE, DELETE ON mail.* TO 'mail_admin'@'localhost'
  IDENTIFIED BY 'mail_admin_password';
GRANT SELECT, INSERT, UPDATE, DELETE ON mail.* TO
  'mail_admin'@'localhost.localdomain' IDENTIFIED BY 'mail_admin_password';
FLUSH PRIVILEGES;
```

Ezekkel létrehoztuk az adatbázist, majd jogosultságot és jelszót állítunk be a mail_admin felhasználó számára, szigorúan lokális kapcsolódás esetére. Természetesen ha másik gépen van az adatbázis, akkor a GRANT parancsot megfelelően paraméterezve akár IP-cím alapján is állíthatunk be az adatbázishoz kapcsolódási jogot.

```
CREATE TABLE domains (domain varchar(50) NOT NULL, PRIMARY KEY (domain) );
CREATE TABLE forwardings (source varchar(80) NOT NULL, destination TEXT NOT
  NULL, PRIMARY KEY (source) );
CREATE TABLE users (email varchar(80) NOT NULL, password varchar(20) NOT NULL,
  PRIMARY KEY (email) );
CREATE TABLE transport (domain varchar(128) NOT NULL default '', transport
  varchar(128) NOT NULL default '', UNIQUE KEY domain (domain) );
quit
```

A fenti utasításokkal létrehozzuk a virtuális domainek és a továbbítandó felhasználók adattábláját. Az SQL részével készen vagyunk. (Kiegészítés: fenti adatbázisfelépítés kis, közepes méretű cég esetén alkalmas. Nagyobb felhasználószám esetén érdemes lehet ezt az adatbázisstruktúra felépítésekor érvényre juttatni. Azaz nem egyetlen **users** táblában tárolni az összes e-mail címet, hanem az egyes virtuális domain-ek virtuális felhasználóit külön-külön táblában. Természetesen ezt nem csak a fenti adatbázis létrehozó parancsoknál, hanem a később szereplő adatlekérdező parancsoknál is figyelembe kell venni, és azokat is megfelelően módosítani kell.)

A fenti paramétereket (admin, jelszó, adatbázis neve) rögzítenünk kell a Postfix számára is. Hozzuk létre a /etc/postfix/sql-virtual_domains.cf állományt, majd írjuk bele a következőket:

```
user = mail_admin
  password = mail_admin_password
  dbname = mail
  query = SELECT domain  FROM domains WHERE domain='%s'
  hosts = 127.0.0.1
```

Értelemszerűen a jelszót és ha más felhasználónevet és/vagy adatbázisnevet adtunk meg az adatbázishoz, akkor azokat is cseréljük ki megfelelően. A hosts esetében pedig abból a feltételezésből indultunk ki, hogy az SQL-motor a lokális gépen a localhoston elérhetően (alap beállítás szerint) működik. Természetesen ha az adatbázis külön virtuális vagy fizikai gépen fut, akkor a

információkért nézzük meg a kézikönyvet (man pwgen), esetleg a windows-os verzió oldalát http://pwgen-win.sourceforge.net

megfelelő biztonsági alapelveket betartva adjuk meg a hozzáférést, azaz például az adatbázis- és a Postfix-gép nyílt hálózaton ne forgalmazzon egymás között, csak (SSL-lel) titkosított formában, amelyet a Mysql és a MariaDB is natívan támogat.[25]

Ugyanígy hozzuk létre a többi táblához tartozó konfig állományokat:

```
/etc/postfix/sql-virtual_forwardings.cf

  user = mail_admin
  password = mail_admin_password
  dbname = mail
  query = SELECT destination FROM forwardings WHERE source='%s'
  hosts = 127.0.0.1
/etc/postfix/mysql-virtual_mailboxes.cf

  user = mail_admin
  password = mail_admin_password
  dbname = mail
  query = SELECT CONCAT( SUBSTRING_INDEX( email, '@', -1 ), '/',
  SUBSTRING_INDEX( email, '@', 1 ), '/' ) FROM users WHERE email='%s'
  hosts = 127.0.0.1
/etc/postfix/mysql-virtual_email2email.cf

  user = mail_admin
  password = mail_admin_password
  dbname = mail
  query = SELECT email FROM users WHERE email='%s'
  hosts = 127.0.0.1
```

A létrehozott állományok fájljogosultságait állítsuk be 640-ra és a root:postfix user:csoport birtokában legyenek (A postfix nevű felhasználói csoportot Ubuntu alatt a csomag telepítése automatikusan létrehozza, ha az általunk választott terjesztés nem tenné, akkor hozzuk létre kézzel.)

Utána hozzunk létre egy felhasználót, amelynek a könyvtárában fognak tárolódni a virtuális felhasználók HOME könyvtárai. Ezt megtehetjük a

```
groupadd -g 5000 vmail
useradd -g vmail -u 5000 vmail -d /home/vmail -m
```

parancsok segítségével, illetve érdemes átgondolni, hogy a /home alkalmas-e arra, hogy ott több száz vagy ezer felhasználó könyvtára (adatokkal) tárolódjon. Ha LVM-et használunk, vagy úgy alakítottuk ki a rendszer, hogy a /home várható telítődése megfelelő, akkor nincs más dolgunk a felhasználóval.

Adjuk meg a Postfix számára a már beállított plusz konfigurációkat, értelemszerűen a dokumentum elejében beállítottakhoz képest pluszban, az előzőekhez képest eltérés ezen kívül, hogy a virtusers állomány helyett SQL-adatbázist fogunk használni. Ezen konfigok rögzítése történhet az eddig megszokott módon (kézzel, szövegszerkesztővel), vagy pedig a **postconf** parancs segítségével, amely hozzá fogja fűzni a main.cf-hez:

```
postconf -e 'virtual_alias_domains ='
postconf -e 'virtual_alias_maps = proxy:mysql:/etc/postfix/sql-
  virtual_forwardings.cf, mysql:/etc/postfix/sql-virtual_email2email.cf'
postconf -e 'virtual_mailbox_domains = proxy:mysql:/etc/postfix/sql-
```

25. https://mifosforge.jira.com/wiki/display/MIFOS/How+to+enable+MySQL+SSL+on+Ubuntu

```
  virtual_domains.cf'
postconf -e 'virtual_mailbox_maps = proxy:mysql:/etc/postfix/sql-
  virtual_mailboxes.cf'
```

Megmondjuk a Postfixnek a domain és a mailbox map SQL konfigjának helyét.

```
postconf -e 'virtual_mailbox_base = /home/vmail'
postconf -e 'virtual_uid_maps = static:5000'
postconf -e 'virtual_gid_maps = static:5000'
```

A felhasználók HOME könyvtárának helyét és UID és GID beállításait rögzítjük.

```
postconf -e 'virtual_create_maildirsize = yes'
postconf -e 'virtual_maildir_extended = yes'
postconf -e 'proxy_read_maps = $local_recipient_maps $mydestination
  $virtual_alias_maps $virtual_alias_domains $virtual_mailbox_maps
  $virtual_mailbox_domains $relay_recipient_maps $relay_domains $canonical_maps
  $sender_canonical_maps $recipient_canonical_maps $relocated_maps
  $transport_maps $mynetworks $virtual_mailbox_limit_maps'
```

Ha ezzel megvagyunk, akkor a Postfixet indítsuk újra:

```
service postfix restart
```

A rendszernaplóban (Ubuntu 12.04 LTS: /var/log/mail.log) meg tudjuk nézni, hogy minden rendben ment-e, illetve érdemes a telnet localhost 25 parancs segítségével ellenőrizni, hogy a Postfix életjelet ad-e a 25-ös porton.

Ha minden rendben ment, akkor létre kell hozni a felhasználókat.

Belépünk az SQL-motorba, akár parancssor, akár phpMyAdmin segítségével:

```
mysql -u mail_admin -p
USE mail;
INSERT INTO domains (domain) VALUES ('sajatceg.hu');
INSERT INTO users (email, password) VALUES (ugyfelszolgalat@sajatceg.hu',
  ENCRYPT('password'));
quit
```

Természetesen ezt is érdemes vagy szkriptelni, vagy pedig egy webes adminisztrációs felületet készíteni hozzá, amelyet a helyi HR rendszerhez hozzáépítve delegálható a megfelelő szintre a felhasználó felvétele, törlése, felfüggesztése, stb. (Legrosszabb esetben pedig egy megfelelő jártassággal rendelkező személyre lehet bízni a phpMyAdminon belüli adminisztrációját)

(Mindezt egészítsük ki azzal, hogy hosszú távon kifizetődő megoldás a levelező felhasználók adatainak adminisztratív kezelését hozzáigazítani az adott szervezet humánerőforrás-menedzsment környezetéhez, és az arra használt eszközbe beilleszteni. Azaz akár a dolgozó munkába állásával egyidejűleg megtörténhet az e-mail adminisztráció – természetesen automatikusan.)

Miután a fenti parancsokkal felvettük az ügyfélszolgálat levelezési címét, még egy teendőnk marad. A Postfix a felhasználó könyvtárát akkor fogja létrehozni, ha megérkezik az első levél. Így mindenképpen célszerű a felhasználó felvétele után egy üdvözlőlevelet küldeni a felhasználónak, akár a mailx/mutt parancsok használatával (mailx -s Udv ugyfelszolgalat@sajatceg.hu < /dev/null, vagy mutt -s Udvozlet ugyfelszolgalat@sajatceg.hu < /dev/null) vagy bármilyen egyéb e-mail klienssel, mint pl. a Thunderbird. A naplóban utána nyomon követhetjük, célba ért-e a levél, majd megnézhetjük a postafiókhoz tartozó könyvtárak létrejöttét. A könyvtár létrehozásáig a felhasználó hiába rendelkezik érvényes jelszóval, a POP3 és IMAP4 kliens sem fog tudni a nevében intézkedni, ezért is

fontos, hogy a próbalevelet kézbesítsük részére. Az ugyfelszolgalat felhasználó postafiókjának könyvtára pedig a következő néven lesz létrehozva: /home/vmail/sajatceg.hu/ugyfelszolgalat/.

SASL

A cél: ugyanazzal a felhasználónévvel és jelszóval, SSL hitelesítés segítségével lehessen levelet küldeni, mint a POP/IMAP-hoz szükséges felhasználói azonosítók. Ehhez a Cyrus SASLAuthd nevű rendszert használjuk. A korrekt működéshez első lépésben beállítjuk a Cyrus SASLAuthd-t úgy, hogy a hitelesítést a PAM-on keresztül az SQL motorból végezze el. Majd pedig a már működő SASLAuthd után a Postfixet[26], hogy a felhasználók azonosítását a Cyrus-féle SASLAuthd-n keresztül intézze.

A következőket kell telepíteni:

```
apt-get install libsasl2-2 libsasl2-modules libsasl2-modules-sql sasl2-bin
  libpam-mysql
```

Mivel a Postfix saját chroot könyvtárában dolgozik (/var/spool/postfix) , ezért létre kell hozni az SASLAuthd-nek a szokásostól eltérő környezetben a könyvtárat:

```
mkdir /var/spool/postfix/var/run/saslauthd
```

A /etc/default/saslauthd konfigurációs fájlba pedig beleírjuk a saját paramétereinket:

```
START=yes
DESC="SASL Authentication Daemon"
NAME="saslauthd"
MECHANISMS="pam"
MECH_OPTIONS=""
THREADS=5
OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd -r"
```

(Engedélyeztük gépindításkor a SASLAuthd automatikus elindítását, a fentebb létrehozott könyvtár nevét adtuk meg a SASLAuthd számára és jeleztük, hogy PAM-on keresztül történjen az azonosítás.)

Mivel a SASLAuthd opciói között azonosításra a PAM-modult neveztük meg, ezért rá kell beszélni a PAM-ot, hogy az SMTP szoftver számára MySQL-ből/MariaDB-ből azonosítson:

Hozzuk létre a /etc/pam.d/smtp állományt a következő tartalommal (ez 2 igen hosszú sor):

```
auth required pam_mysql.so user=mail_admin passwd=mail_admin_password
  host=127.0.0.1 db=mail table=users usercolumn=email passwdcolumn=password
  crypt=1
account sufficient pam_mysql.so user=mail_admin passwd=mail_admin_password
  host=127.0.0.1 db=mail table=users usercolumn=email passwdcolumn=password
  crypt=1
```

Értelemszerűen az SQL paramétereket állítsuk be a korábban már megadott értékekre (adatbázis neve, táblák neve, felhasználónév az adatbázishoz, jelszó).

26. A Postfix a Cyrus és a Dovecot-féle SASL-megvalósítást támogatja. A postconf -a (SMTP-szerver üzemmód) illetve postconf -A (SMTP-kliens üzemmód) opciókkal ellenőrizhető, hogy a telepített verzióban melyik működik.

Hozzuk létre a következő könyvtárat: /etc/postfix/sasl/[27] és abban egy állományt: /etc/postfix/sasl/smtpd.conf[28], melynek tartalma:

```
pwcheck_method: saslauthd
mech_list: plain login
```

legyen.

Majd miután elmentjük a SASLAuthd-nek szóló beállításokat, csak az van hátra, hogy a Postfix számára is nyilvánvalóvá tegyük, hogy az SASLAuthd-t fogjuk használni azonosításra. A Postfix-hez tartozó main.cf-be rögzítsük a beállításokat a következő parancsok segítségével:

```
postconf -e 'smtpd_sasl_auth_enable = yes'
postconf -e ‘smtpd_sasl_type = cyrus’
  postconf -e ‘smtpd_sasl_path = smtpd’
postconf -e 'smtpd_sasl_authenticated_header = yes'
postconf -e ‘smtpd_sasl_security_options = noanonymous’
postconf -e 'broken_sasl_auth_clients = yes'
postconf -e 'smtpd_recipient_restrictions = permit_mynetworks,
  permit_sasl_authenticated, reject_unauth_destination'
postconf -e 'proxy_read_maps = $local_recipient_maps $mydestination
  $virtual_alias_maps $virtual_alias_domains $virtual_mailbox_maps
  $virtual_mailbox_domains $relay_recipient_maps $relay_domains $canonical_maps
  $sender_canonical_maps $recipient_canonical_maps $relocated_maps
  $transport_maps $mynetworks $virtual_mailbox_limit_maps'
```

Az adatbázisból történő autentikációval végeztünk, de a levelezőszerver éles használata előtt még meg kell tanítanunk a Postfixet arra, hogy titkosított kommunikációt végezzen, hiszen most már nagy érték a felhasználó jelszava és e-mail címe, mivel segítségével majdnem korlátlan mennyiségben lehet levelet küldeni, helytől független módon. A legjobb megoldás tehát, ha SSL-kulcsot készítünk a titkosított kommunikációhoz.

(Az SSL-lel és tanúsítványkezeléssel kapcsolatos dokumentum része a keretrendszernek, abban a minimálisan szükséges háttér-információk mellett pár alternatív eszköz használatát mutatjuk be: az igen ismertnek számító, de fapadosan kényelmetlen OpenSSL parancssori kezelése, és néhány újabb, akár grafikus eszköz. Ezért a kulcsgenerálás ebben a fejezetben meglehetősen tömör, pár lépéses használata szerepel a dokumentumban.)

Ha még nem volt telepítve, telepítsük a gépre az OpenSSL csomagot:

```
apt-get install openssl
```

Készítsük el a szerver tanúsítványát:

```
openssl req -new -outform PEM -out smtpd.cert -newkey rsa:2048 -nodes -keyout
  smtpd.key -keyform PEM -days 730 -x509
```

A fenti parancs kiadása után az OpenSSL feltesz pár kérdést, amelyre a cég neve, címe stb. alapján adjuk meg a megfelelő válaszokat. Az elkészült smtpd.key és smtpd.cert állományt másoljuk a /etc/postfix/ könyvtárba, majd állítsuk be a Postfix számára is láthatóan az SSL-lel kapcsolatos paramétereket:

```
postconf -e 'smtpd_use_tls = yes'
postconf -e 'smtpd_tls_auth_only = yes'
```

27. Ez az elérési útvonal Ubuntu esetén érvényes, más terjesztések esetén más lehet (pl. /usr/lib/sasl2)
28. Noha ez a Cyrus-hoz tartozó konfigfájl, a Postfix main.cf fájljában szereplő smtpd_sasl_path=**smtpd** határozza meg, hogy ennek a fájlnak smtpd.conf lesz a neve

```
postconf -e 'smtpd_tls_cert_file = /etc/postfix/smtpd.cert'
postconf -e 'smtpd_tls_key_file = /etc/postfix/smtpd.key'
postconf -e 'tls_random_source = dev:/dev/urandom'
```

Végezetül pedig indítsuk újra a Postfixet:

```
service postfix restart
```

Hasznos vállalati funkciók

Hasznos vállalati plusz funkciók amelyek könnyen kivitelezhetőek Postfix segítségével:

Always BCC: a teljes átmenő levélforgalom nem csak könnyedén naplózható, hanem tárolható is, küldött és fogadott levelek napi/heti/havi bontásban akár maildir vagy mailbox formátumban a felhasználó által nem észlelhető módon. Ez a funkció a hatályos magyar törvények szerint magáncímek esetében (ilyenek a sok helyen használt vezetéknév.keresztnév@e-mail.cim) csak akkor kapcsolható be, ha az érintettek erről írásban nyilatkoznak (pl. a munkaszerződésük része). Ezen funkció segítségével bármikor visszaállítható a felhasználó által véletlenül vagy szándékosan letörölt levél – használata természetesen dupla akkora lemezigénnyel jár, mint alapesetben. Ezt az opciót is a main.cf-ben kell elhelyezni:

```
  always_bcc = bcc
```

az = jel után a bcc egy felhasználó neve, akinek postafiókjába minden egyes küldött és fogadott levelet tárol majd a rendszer. Fontos figyelni arra, hogy a bcc használata esetén, az egész átmenő levelezés miatt a megfelelő lemezméretet szükséges biztosítani. Igazán nagy átmenő forgalom esetén célszerű külön lemezre (lehetőleg raid1+0 legyen) tenni a BCC-t, hogy a plusz I/O-műveletek jobban el legyenek osztva. További fontos teendő, hogy a BCC felhasználónak feltétlen Maildir formátumot állítsunk be, mivel hatalmasra nőhet ez a tároló és nem túl praktikus 1 hatalmas állományban tárolni mindent, valamint később mentés és a visszakeresés szempontjából is nehézkes ez a megoldás. A BCC megoldás egy dedikált lemez segítségével kiegészítheti a hagyományos mentésünket is. Hiszen itt minden küldött és fogadott levél tárolva van (bármi ami átmegy az SMTPD-n, sajnos a nem megfogott spamek is), ezért ha egy felhasználó pont két snapshot közötti időpontban törölt ki egy nagyon fontos levelet, azt később itt meg lehet keresni és továbbítani számára. A Maildir tárolás esetén reguláris kifejezésekkel, szkriptek segítségével, vagy akár a Midnight Commander használatával (keresés – reguláris kifejezések alapján egy adott könyvtár vagy könyvtár szerkezetben) igen eredményesen lehet a levelek fejléce vagy törzse alapján is keresni. Azaz egy szerény változó mögé lehet építeni egy egyedi, akár minősített követelményeknek is megfelelő hosszútávú levéltárolást. Mivel itt minden céges információ tárolva lesz, ami csak áthalad a levelezőszerveren, ezért bizonyos esetekben érdemes megfontolni a titkosított fájlrendszer (pl: Linux loop-aes, FreeBSD Geli, stb.) használatát[29], ami viszont jelentős mértékű plusz CPU, I/O és memória terhelést okoz.

PFLogsumm[30]: egy Perl-program amely napi bontásban kiértékeli az MTA forgalmát. Remek statisztikákat lehet a HR vagy az ügyvitel számára készíteni vele. Valamint a rendszermérnököknek is hasznos, ugyanis bontásban mutatja a sosem kézbesített és egyéb okokból elakadt levelek számát is. Használata igen egyszerű, a telepítés után a Perl szkriptnek paraméterben meg kell adni a kiértékelendő naplóállomány helyét. A kimenetet pedig irányíthatjuk állományba, vagy akár

29. Mindenképpen tartsuk be a helyi adatvédelmi előírásokat.
30. http://jimsun.linxnet.com/postfix_contrib.html

crontab-ból futtatva, levélben kaphatjuk meg. A végeredmény egy domainekre és felhasználókra lebontott, részletes és emészthető statisztika.

Munin integráció[31]: a Munin gyári bővítmény segítségével grafikusan is könnyedén nyomon követhetők a csúcsidők, a hírlevél-kiküldések szerverre gyakorolt hatása, illetve ez alapján lehet a működést elemezni, monitorozni.

Disclaimer: Postfixben is van lehetőség minden levél végére központilag megjegyzéseket fűzni. Általában véve ezt Disclaimer-nek hívjuk. A Postfix remekül együttműködik az Altermime[32]-mal, amelynek beüzemelése viszonylag egyszerű.[33]

Submission port[34]: Postfixben lehetőség van az 587-es port megnyitására, amelyet a klienseknek adhatunk meg levél küldésre a 25-ös és a 465-ös port mellé (vagy helyett). Jelentősége akkor van, ha olyan hálózatból kell kliensként leveleket küldenünk, ahol alapesetben a 25-ös és a 465-ös port elérése tiltva van (spam- és vírusvédelmi okokra való hivatkozással). Ebben az esetben a master.cf-ben a sor elején álló megjegyzésjel (#) törlésével engedélyeznünk kell a

```
#submission inet n       -       -       -       -       smtpd
```

sort és a tűzfalon is utat nyitni számára.

Illesztőprogram, amely összeköti a levelezőszervert, a víruskeresőt és a levélszemétkeresőt *(a tényleges konfigurációt egyben lehet megtalálni ennek a fejezetnek a végén, mivel ez a három eszköz: Amavis+spamszűrő+vírusszűrő motor együttműködve értelmezhető a leginkább)*

Amavis[35]: egy interfész az MTA és a különböző levélellenőrző-programok közé ágyazva, amely biztosítja közöttük a gyors kommunikációt. Ha gyors és könnyen kezelhető átjárót szeretnénk az MTA és a levélfeldolgozó programok között, akkor igazi alternatívája nincs. Átláthatóan konfigurálható, nagy teherbírású program, amelyet Perlben írtak. Beépített védelemmel rendelkezik az esetleges diszktelítettség, vagy diszk I/O-hibából eredő eseményekre. Dkim aláírás támogatással rendelkezik és természetesen GPL licencelésű.

Vírus és spam elleni védelem: azok az idők vélhetően visszavonhatatlanul elmúltak, amikor egy szervezet levelezése vírus- és levélszemét-védelem nélkül elképzelhető volt. A vírusvédelem még a szabad operációs rendszerek alatt is tanácsos, hiszen a szerverekhez kapcsolódó kliensek lehetnek védtelenek is, valamint a szerverre beérkező leveleket első vonali védelemmel ellátva, központilag szabályozva azok szűrését sokkal védettebbé tehetjük hálózatunkat. Az első vonali védekezés pedig az esetek túlnyomó többségében egyszerűbb és hatékonyabb is. Több megoldás is található:

– Greylisting
– Dspam[36]
– Anti-Spam SMTP Proxy Server[37]
– SpamBayes[38]

31. http://munin-monitoring.org/
32. http://www.pldaniels.com/altermime/
33. http://www.howtoforge.com/add-disclaimers-to-outgoing-emails-with-altermime-postfix-debian-etch
34. http://en.wikipedia.org/wiki/Mail_submission_agent
35. http://www.ijs.si/software/amavisd/
36. http://dspam.nuclearelephant.com/
37. http://sourceforge.net/projects/assp/
38. http://spambayes.sourceforge.net/

– Bogofilter[39]

– Clapf[40]

– SpamAssasin[41]

Greylisting: a szürkelista névre is hallgató mechanizmus lényege, hogy egy adott címről érkező levelet a levelezőszerver azonnal visszautasít egy ideiglenes hibára utaló jelzéssel. A szabványos működésű SMTP-szerverek az ilyen hibakóddal visszautasított levelet némi késlekedés után újra-és-újra megpróbálják kézbesíteni. A fogadó oldal pedig egy meghatározott idő eltelte után elfogadja a levelet. Mivel általában tárolódik, hogy kitől fogadtunk el levelet, ezért nem az összes, csak az első (pár) levél késleltetése szembeötlő. Viszont a spamek jó része nem szabványosan viselkedő szerverektől jön akik a nagyobb hatékonyság érdekében általában nem túl szabványosak (ezért pl. meg sem kísérelnek ilyenkor újraküldeni), így ezzel a technológiával nagy részüktől meg lehet szabadulni. Komoly hátulütője a technológiának, hogy a felhasználók hite szerint a levelezés azonnali üzenettovábbítást jelent, így mondjuk már egy 30 perces visszautasítást nem tolerálnak. Postfixhez használható a Postgrey nevű eszköz, amely alapbeállításban 5 perces visszautasítási idővel dolgozik. Ismert trükk, hogy az elején a szokásos levelezőpartnerek címének megtanulásához rövid – mondjuk 1 perces – időintervallumot választunk, és csak a későbbiekben, a kezdő adatbázis elkészülte után emeljük meg a várakozási értéket, de akkor se nagyon legyen 10 percnél több.

Dspam: Készítője Jonathan A. Zdziarski, a program 99,5–99,95%-os találati pontossággal működik. Gyors reagálás és nagy teherbírás jellemzi, azonban a jó eredmény eléréséhez folyamatos tanítást igényel. Hátránya, hogy lokális adatbázisban tárolja a betanított mintákat, amelyben egy idő után nehézkesen és lassan tud keresni.

Anti-Spam SMTP Proxy Server: egy 2003 óta fejlesztett nyílt forráskódú, GPLv2 licencű Perl SMTP proxy, amely a következő tulajdonságokkal bír: Bayes-szűrés[42], feketelista (DNSBL[43]), SPF[44], SRS[45], szürkelista[46]. Felróható hátrány: relatív kisebb felhasználói bázis és ebből eredően a támogatói fórum is szűkebb körű a többihez viszonyítva.

SpamBayes: Python programozási nyelven írta Paul Graham, a Bayes-szűrés kidolgozója. Működési módszere, hogy 3 halmazba szervezi a beérkezett leveleket, az első halmaz a spam, a második a normál levél (sűrűn használt elnevezéssel: HAM), a harmadik amelyről nem tudja, hogy az első kettő közül melyikbe tartozzon. A program előnye, hogy relatív kevés hamis pozitív és hamis negatív eredményt ad, viszont a 3. kategóriába eső leveleket a felhasználónak kell minősítenie. Hátránya, hogy így a felhasználóra relatív több manuális munka jut.

Bogofilter: egy szintén Bayes-szűrést alkalmazó nyílt forráskódú, C programozási nyelven írt levélszemétszűrő, melyet Eric S. Raymond fejlesztett ki. Könnyen használható akár asztali környezetben is, azonban szerver környezetben probléma léphet fel a párhuzamos vizsgálatok során, ugyanis egyedileg a felhasználó szkripttel (procmail segítségével) indítja. Létezik hozzá milter-interfész is, amelynek segítségével eredendően Sendmail, újabban Postfix alá is beépíthető.

39. http://bogofilter.sourceforge.net/
40. http://clapf.org/wiki/doku.php/start
41. http://spamassassin.apache.org/
42. http://en.wikipedia.org/wiki/Bayesian_spam_filtering
43. http://en.wikipedia.org/wiki/DNSBL
44. http://en.wikipedia.org/wiki/Sender_Policy_Framework
45. http://en.wikipedia.org/wiki/Sender_Rewriting_Scheme
46. http://en.wikipedia.org/wiki/Greylisting

Clapf: magyar fejlesztésű (Sütő János), viszonylag újnak számító statisztikai szűrő. PHP-ben készült, Postfixhez és Eximhez illeszthető, illetve a lokális levélkézbesítési fázisban pl. a procmail (később még lesz róla szó) segítségével tulajdonképp bármilyen levelezőrendszerhez.

SpamAssasin: egy ún. „pontozásos" rendszerben működő levélszemétszűrő, amely az Amavisd segítségével kapcsolható az MTA-hoz. Többszörösen díjnyertes, Perl nyelven íródott, az e-mailek elemzésével, kulcsszavak, írásstílusok, és sok egyéb jel figyelésével, feketelistákkal, valamint a Bayes-szűrés módszerével operálva próbálja megállapítani egy levélről, hogy átengedhető-e vagy sem. A program nagyjából ezer különböző tesztnek vet alá minden e-mailt. Ezek a tesztek az e-mailek tartalmát, felépítését, szabványosságát, méretét, képek, mellékletek elhelyezkedését, a levél korábbi állomásait vizsgálják át, meghamisításra utaló jeleket keresnek, és még sok mást is vizsgálnak. A tesztek pontozása alapján az előre beállított érték szerint hozza meg a döntést. A pontozás alap értékei 4-5 között szoktak lenni, amely például a következő tényezőkből állhat össze:

– MISSING_SUBJECT: 2,5 pont: nincs az üzenetnek tárgya
– NUMERIC_HTTP_ADDR: 0,9 pont: numerikus IP-cím egy URL-ben
– BAYES_00: −2,6 pont: A Bayes-szűrés szerint egyértelműen ham – azaz jó – levél
– BAYES_99: 3,5 pont: A Bayes-szűrés szerint egyértelműen spam – azaz rossz – a levél
– URIBL_BLACK: 2,0 pont: Egy a levélben szereplő URL szerepel az URIBL feketelistán
– RCVD_IN_BL_SPAMCOP_NET: 2,2 pont: a levél olyan állomáson haladt keresztül, amely megtalálható a SpamCop feketelistán.

Ugyanilyen pontozás alapján dönt a hamisított fejlécű e-mailekről (leggyakoribb előfordulása, amikor a címzett látszólag saját magától kap 3. féltől levelet). A Spamassassin-szűrő SPF (Sender Policy Framework) ellenőrzést is végez. Rengeteg kiegészítő kapcsolható hozzá, amelyek használata ajánlott is, ilyenek pl. a dcc-client,[47] pyzor[48], razor[49] vagy PFassassin, SpamAssassin Rules Emporium[50], vagy a FuzzyOCR[51]. Mivel Perl nyelven íródott – amely nem kifejezetten a teljesítményre való kihegyezés ismérve – a program memóriaigényes, de a memóriaigény jól számolható. Létezik hozzá több adminisztráció frontend, webes felület, ahol mindenki felhasználóbarát módon saját maga tudja az amúgy az átlagosnál bonyolultabb szabályrendszerét állítani. Ilyen pl. a SquirrelMail-be integrálható bővítménygyűjtemény, illetve könnyű hozzá egyedi PHP-alapú végfelhasználói felületet létrehozni (bár ez felvethet némi kockázatot).

A levélszemétszűrők között egészen más szempont szerint kell választanunk, mint az MTA esetében, ahol sok, nagyjából egyenlő tudású MTA között kell döntést hozni. A választás a SpamAssasin-ra esett, tekintve, hogy az összes közül a legjobban és legegyszerűbben integrálható, tanítható és kezelhető, továbbá a legtöbb módszert alkalmazza a levélszemét felderítési mechanizmusában. A támogatottsága a felhasználói fórumok számát tekintve a legnagyobb és a legtöbb extra szolgáltatás ehhez kapcsolható.

47. http://spamassassin.apache.org/full/3.2.x/doc/Mail_SpamAssassin_Plugin_DCC.html
48. http://sourceforge.net/apps/trac/pyzor/
49. http://en.wikipedia.org/wiki/Vipul%27s_Razor
50. http://sourceforge.net/projects/sare/
51. http://wiki.apache.org/spamassassin/FuzzyOcrPlugin

Víruskereső programok

Önmagában a levélszemétszűrés nem elegendő, bár rendszeresen tapasztalható, hogy a levélszemét kiszűrésével tetemes mennyiségű vírusos levéltől lehet megszabadulni. Mára a Linux és BSD-rendszerű operációs rendszerek is vonzó célpontot jelentenek a víruskereső szoftvereket fejlesztő cégek számára. Számos zárt termék integrálható a levélszűrés folyamatába, azonban jellemzően ezek kereskedelmi felhasználásra jogdíjkötelesek.

ClamAV[52]: parancssori alapú több szálon működő víruskereső, amely képes a legtöbb tömörített fájlban ellenőrizni, illetve ELF (futtatható) binárisok vizsgálatára is. Vizsgálni tudja továbbá az MS termékpaletta legtöbb dokumentumformátumát. Naponta többszöri frissítés érkezik a központi adatbázisból (alapbeállítás szerint 15 percenként); az egyik leggyorsabban reagáló csapat, amely gyorsaság eredménye, hogy néha hozhat hamis pozitív eredményt is. A ClamAV vírus-adatbázisa publikusan megnézhető.[53] Számos grafikus frontend létezik hozzá, amely segítségével asztali gépen, illetve szervereken, Samba megosztásokon is használhatjuk, pl: KlamAV, ClamTk, AVScan GTK, ClamWin. Része a legtöbb Linux terjesztésnek, és jól kapcsolható az előbbiekben felsorolt szoftverekhez. Könnyen beállítható és logikus felépítésű konfigurációs állomány jellemzi. Fontos tudni azonban, hogy a ClamAV-ot 2007-ben felvásárolta a Sourcefire, amely megszerezte a ClamAV csapat öt vezető tagjának szerzői jogait, magát a projektet és a hozzá kapcsolódó márkaneveket, védjegyeket is. Ezen felül hozzá kerülnek a ClamAV.org domain feletti jogok, a weboldal tartalma és a SourceForge-os ClamAV projektoldal is. Akkor a ClamAV core fejlesztői folytatták a munkát, mint a Sourcefire alkalmazottai. Ugyanakkor a ClamAV licencelése GNU GPLv2.

A legnépszerűbb zárt termékek Linuxra

AVG Free for Linux[54]**:** parancssori felülettel rendelkezik, otthoni felhasználásra 1 gépig ingyenes.

BitDefender[55]**:** parancssori felülettel rendelkezik, otthoni felhasználásra ingyenes.

Avast for Linux[56]**:** parancssori felülettel is rendelkezik, otthoni felhasználásra ingyenes.

Kaspersky[57]: a linuxos verzióiból csak próba (trial) verzió érhető el ingyenesen, a vállalati felhasználása fizetős.

Összességében elmondható, hogy otthoni védekezésnek, pl. másoktól kapott dokumentumok, USB-kulcs stb. ellenőrzésére kiválóak, azonban ingyenes szerver felhasználásuk jogi okokból nem lehetséges.

Gyakorlatilag alternatíva nélkül a ClamAV víruskereső ajánlható jelenleg. Összességében azonban bátran használható, hiszen cégek kereskedelmi forgalomban lévő csomagolt termékeinek a sikeres alapja. A kezdetektől biztosított a program életút (folyamatos distupgrade lehetőség a kezdetektől), a frissítés, személyre szabhatóság és a könnyű illesztés az egyedi szűrőkhöz, illetve a ClamAV vírus-adatbázis központ országonként választható.

52. http://www.clamav.net/lang/en/
53. http://lurker.clamav.net/list/clamav-virusdb.html
54. http://free.avg.com/us-en/download.prd-alf
55. http://www.bitdefender.com/business/antivirus-for-unices.html
56. http://linux.softpedia.com/get/Security/avast-Linux-Home-Edition-43586.shtml
57. http://www.kaspersky.com/products/business/applications/endpoint-security-linux

Amavis+ClamAV+SpamAssasin konfigurálás

Konfiguráljuk tehát az Amavis+ClamAV+SpamAssasin hármast:

```
apt-get install amavisd-new spamassassin clamav-daemon arj unarj bzip2
  cabextract cpio file gzip lha nomarch pax rar unrar unzip zip zoo
```

A parancs kiadása után települni fog a rendszerünkre az Amavisd, a ClamAV és a SpamAssassin, magukkal hozva a szükséges függőségeket is (valamint nagyon sokféle elterjedt és nem annyira elterjedt tömörítőprogramot), így ne ijedjünk meg, ha azt látjuk a telepítő nagyon sok csomagot kezd el tölteni és beállítani. Az ubuntus telepítő parancsfájl létre fogja hozni a szükséges felhasználókat, így nekünk már csak sorban be kell állítanunk a fenti szoftvereket.

Kezdjük a SpamAssasin alap bekapcsolásával, amelyet a /etc/default/spamassassin állományban tudunk megtenni a következőképpen:

```
ENABLED=1
CRON=1
```

azaz bekapcsoltuk és engedélyeztük neki a szabályok automatikus frissítését cronból.

```
service spamassassin start
```

és már fut is.

A SpamAssasin finomhangolása felhasználónként lehetséges, általában véve a webes levelezők bővítményei segítségével, illetve kapcsolható hozzájuk Pyzor és Razor is, bővebben a finomhangolásról az alábbi[58] linkeken[59] lehet olvasni.

A soron következő lépés, hogy az Amavisd-new-t kapcsoljuk be, és tudassuk vele, hogy vírus- és spamszűrést fogunk a segítségével végezni, azaz a /etc/amavis/conf.d/15-content_filter_mode állományban kommentezzük ki a szükséges sorokat:

```
use strict;
# You can modify this file to re-enable SPAM checking through spamassassin
# and to re-enable antivirus checking.
# Default antivirus checking mode
# Uncomment the two lines below to enable it
@bypass_virus_checks_maps = (
   \%bypass_virus_checks, \@bypass_virus_checks_acl,
   \$bypass_virus_checks_re);
# Default SPAM checking mode
# Uncomment the two lines below to enable it
@bypass_spam_checks_maps = (
   \%bypass_spam_checks, \@bypass_spam_checks_acl, \$bypass_spam_checks_re);
1;  # insure a defined return
```

majd:

```
service amavis restart
```

Most már az Amavisd-new kezeli a SpamAssassint és a ClamAV-ot is, viszont a levelek még nem jutnak el hozzá, mivel a Postfix nem tudja, hogy azokat át kellene adnia, ezért a következő lépésben elmagyarázzuk a Postfix számára, hogy a leveleket minden esetben amikor beérkeznek, a

58. http://linuxgazette.net/105/youngman.html
59. http://wiki.apache.org/spamassassin/UsingPyzor

megfelelő sorrendben át kell adnia az Amavisd számára. Ehhez a /etc/postfix/main.cf végéhez szúrjuk be a következő sort:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Akár kézzel, akár a már korábban használt módon:

```
postconf -e "content_filter = smtp-amavis:[127.0.0.1]:10024"
```

Most pedig hozzá kell nyúlnunk a /etc/postfix/master.cf fájlhoz is (a fájl legvégéhez fűzzük hozzá a következőket):

```
smtp-amavis     unix    -       -       -       -       2       smtp
          -o smtp_data_done_timeout=1200
          -o smtp_send_xforward_command=yes
          -o disable_dns_lookups=yes
          -o max_use=20

  127.0.0.1:10025 inet    n       -       -       -       -       smtpd
          -o content_filter=
          -o local_recipient_maps=
          -o relay_recipient_maps=
          -o smtpd_restriction_classes=
          -o smtpd_delay_reject=no
          -o smtpd_client_restrictions=permit_mynetworks,reject
          -o smtpd_helo_restrictions=
          -o smtpd_sender_restrictions=
          -o smtpd_recipient_restrictions=permit_mynetworks,reject
          -o smtpd_data_restrictions=reject_unauth_pipelining
          -o smtpd_end_of_data_restrictions=
          -o mynetworks=127.0.0.0/8
          -o smtpd_error_sleep_time=0
          -o smtpd_soft_error_limit=1001
          -o smtpd_hard_error_limit=1000
          -o smtpd_client_connection_count_limit=0
          -o smtpd_client_connection_rate_limit=0
          -o
  receive_override_options=no_header_body_checks,no_unknown_recipient_checks
```

Majd pedig keressük meg a “pickup” kezdetű sort, ez az elején található és közvetlenül a pickup sor alá szúrjuk be a következőket:

```
       -o content_filter=
          -o receive_override_options=no_header_body_checks
```

Ezzel kiegészítve a pickup kezdetű sort, amely ezek után így fogk kinézni:

```
pickup    fifo  n       -       -       60      1       pickup
   -o content_filter=
          -o receive_override_options=no_header_body_checks
```

A fenti módosítások eredménye, hogy a Postfix újraindítása után minden levél átadásra kerül elemzésre a 10 024 -es lokális porton figyelő Amavisd-new-nak, amely a továbbiakban a Clamavval és a Spamassasinnal végezteti el a tényleges munkát.

Végezetül pedig indítsuk újra a Postfixet:

```
service postfix restart
```

Ha jól végeztük dolgunkat, akkor le is tesztelhetjük:

```
telnet localhost 10024
```

és valami hasonlót kell visszakapnunk:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
```

A levelek elérése

A fenti megoldások segítségével levélszemét- és vírusmentesen tudjuk tárolni adatbázisban vagy lokális postafiókokban (mailbox/maildir) a felhasználók leveleit. Ezen a ponton merül fel az igény arra, hogy a felhasználó egy azonosítás után kézhez is vehesse leveleit. Ezt a szolgáltatást jobbára ma már csak webes levelezők, illetve titkosított IMAP4 segítségével nyújtjuk, de a régi, bevált POP3-ra is számtalan megoldás kínálkozik. A legtöbb Linux terjesztés alatt, akárcsak a Postfix és a SpamAssasin, ugyanúgy előre csomagolva elérhetőek a minden igényt kielégítő IMAP4- és POP3-szerver implementációk:

Dovecot imap4/pop3[60]**:** az egyik leggyorsabb IMAP4 implementáció, amely nagy terhelés alatt is kiváló teljesítményt nyújt. Az egyik legtöbb azonosítási módot felvonultató kiszolgáló. Az IMAP4 és a POP3 egy felületről vezérelhető. Natív SSL-támogatása van. Fontos további szempont, hogy a Dovecot fejlesztői mindenekelőtt a biztonságos üzemeltetést tűzték ki a projekt alapcéljai közé. Érdemes tudni, hogy igen nagy terhelésű rendszerek esetében proxyként is alkalmazható a megfelelő elosztott rendszer részeként. Telepítéséhez az

```
apt-get install dovecot-common dovecot-imapd dovecot-pop3d
```

parancsot használjuk. Konfigurációs állománya a /etc/dovecot/dovecot.conf fájl, amely igen terjedelmes. Szerencsére az előzőekben felépített, az azonosítást SQL-ből az SASLAuthd segítségével megvalósító Postfix rendszerhez nekünk most nem lesz szükségünk túl sok plusz opcióra. A legegyszerűbb, ha a Postfix-szel tudatjuk, hogy a Dovecot lesz az együttműködő partnere. Az /etc/postfix/master.cf végére írjuk be a következőket:

```
dovecot   unix  -       n       n       -       -       pipe
      flags=DRhu user=vmail:vmail argv=/usr/lib/dovecot/deliver -d ${recipient}
```

Majd a /etc/dovecot/dovecot.conf fájlt töltsük fel a saját paramétereinkkel:

```
protocols = imap imaps pop3 pop3s
log_timestamp = "%Y-%m-%d %H:%M:%S "
mail_location = maildir:/home/vmail/%d/%n/Maildir
ssl_cert_file = /etc/postfix/smtpd.cert
ssl_key_file = /etc/postfix/smtpd.key
namespace private {
    separator = .
    prefix = INBOX.
    inbox = yes
```

60. http://www.dovecot.org/

```
}
protocol lda {
    log_path = /home/vmail/dovecot-deliver.log
    auth_socket_path = /var/run/dovecot/auth-master
    postmaster_address = postmaster@sajatceg.hu
    mail_plugins = sieve
    global_script_path = /home/vmail/globalsieverc
}

  protocol pop3 {
      pop3_uidl_format = %08Xu%08Xv
  }

  auth default {
    user = root
    passdb sql {
        args = /etc/dovecot/dovecot-sql.conf
    }
    userdb static {
        args = uid=5000 gid=5000 home=/home/vmail/%d/%n allow_all_users=yes
    }
    socket listen {
        master {
            path = /var/run/dovecot/auth-master
            mode = 0600
            user = vmail
        }

        client {
            path = /var/spool/postfix/private/auth
            mode = 0660
            user = postfix
            group = postfix
        }
    }
}
```

Mint ahogy a fenti konfigurációból látszik, a korábban létrehozott SSL-tanúsítványt használtuk, illetve a már bekonfigurált Postfix- és SQL-kapcsolat adatait adtuk meg a Dovecot számára. Ezt csak akkor tehetjük meg, ha ugyanazon a néven érhető el a Dovecot-szerver, mint az SMTP-szerver (azaz mind a kettő pl. mail.sajatceg.hu). Hátra van még az SQL-kapcsolat konfigurációjának beállítása, hasonlóan mint a Postfix, illetve az SASLAuthd esetében. Hozzuk létre, vagy ha már van akkor módosítsuk a /etc/dovecot/dovecot-sql.conf fájlt a következőképpen:

```
driver = mysql
connect = host=127.0.0.1 dbname=mail user=mail_admin
password=mail_admin_password
default_pass_scheme = CRYPT
password_query = SELECT email as user, password FROM users WHERE email='%u';
```

Mielőtt újraindítanánk a Dovecotot, állítsuk be a konfigurációs állományokat, hogy a Dovecot is tudja olvasni:

```
chgrp vmail /etc/dovecot/dovecot.conf
```

```
chmod g+r /etc/dovecot/dovecot.conf
```

Ezek után pedig már újraindítható a Dovecot:

```
service dovecot restart
```

Egy tesztet futtassunk azért le kézzel, hogy minden rendben van-e, azaz nézzük meg a mail.log-ot, hogy nincs-e benne valamilyen hibaüzenet (bármi, amiben „error” áll, gyanús), valamint a telnet localhost 110 és telnet localhost 143 parancs segítségével megnézhetjük a Dovecot hallgat-e a megfelelő portokon. Az SSL-kapcsolaton keresztüli működést pedig a következőképpen tudjuk tesztelni:

```
openssl s_client -connect imap.sajatceg.hu 993
```

Fontos tudni, hogy a Dovecot elképesztően érzékeny az időcsúszásokra. Azaz sem előre sem hátra nem jól tolerálja, ha elcsúszik alóla az időszinkront. Ha nem szinkronizálunk ntpd-vel folyamatosan, hanem csak naponta (pl. hajnalban) egyszer, akkor feltétlenül használjuk az időszinkronizációs programok csöpögtető üzemmódját, illetve érdemes az Időszinkronizálás[61] fejezetben leírtak szerint eljárni és egy szerveren folyamatos időszinkront alkalmazni. Ha mégis gondunk támadna a Dovecot és az idő kérdésével akkor vagy a monit[62] használatával figyeltessük, vagy alkalmazzunk egyéb megoldásokat, pl. a dokumentációban felsoroltak közül.[63]

Courier[64]**:** jól optimalizált memóriafoglalású program, amelynél külön kiemelendő a DDOS támadásokra felkészülés gyanánt könnyen állítható maximális kapcsolódások száma felhasználónként és IP-címenként. Működik benne a felhasználók közötti megosztott könyvtár kezelése, viszont csak Maildir támogatása van. Ugyan a Maildir az egyik legjobb választási lehetőség, különösen a nagy terheltségű rendszerek esetében, azonban rengeteg helyen a mailbox is követelmény lehet. POP3/IMAP4 proxy lehetőséget is nyújt.

Cyrus[65]**:** A Cyrus egy BSD licencű projekt, amit 1994-ben indított a Carnegie-Mellon University. A Cyrus a Cyrus SASL Library-t használja, amelyben több különböző azonosítási folyamat választható. Hozzáférési modellje és felhasználókezelése összetett és a felsorolt MDA-khoz képest nehézkesebb. Ugyanezen okokból kifolyólag széleskörűen skálázható, és ötvözi a legtöbb hasonló program jó tulajdonságait és talán az összes közül a leginkább ACL-ezhető, azaz a hozzáférési beállítások itt a legszofisztikáltabbak.

A választás sokkal nehezebb, mint a többi megoldás esetében, ugyanis a Dovecot és a Cyrus megoldásai hasonlítanak. Amíg az egyik komplexebb, addig a másik egyszerűbben kezelhető és karbantartható. Itt is hasonló szempontokat kell figyelembe vennünk, mint az MTA kiválasztásánál, illetve érdemes kipróbálni mindegyik olyan megoldást, amely a tudását tekintve megfelel a feladatra.

Titkosítási réteg: a mai korban felér az adatok önkéntes átadásával, ha a levelezésünket titkosításmentesen, a hagyományos nem titkosított TCP-csatornákon folytatjuk. Az SSL kulcsok mindenki számára kényelmesen elérhetőek, akár ingyenes önaláírt (Self Signed) akár egy harmadik fél által aláírt megbízható (Trusted) módban is, illetve a kettő közötti változatban a startSSL[66] segítségével is, ahol a cég hitelesíti az SSL kulcsunkat, de csak 1 évre teszi ezt díjmentesen, megkötésekkel. A legjobb megoldás természetesen valamelyik ismert és támogatott CA által kiadott magas

61. http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/
62. http://mmonit.com/monit/
63. http://wiki.dovecot.org/TimeMovedBackwards
64. http://www.courier-mta.org/
65. http://cyrusimap.web.cmu.edu/
66. http://www.startssl.com/

bitrátájú kulcs megvásárlása, de ez jelenleg anyagi források bevonását igényli, általában évente, amíg a saját magunk által készített kulcsot jellemzően a kliensek el tudják már menteni, így felhasználó oldali oktatás segítségével ez is megoldható, akár saját készítésű root CA készítésével, amelyet elhelyezünk a kliens oldali programokban.

Tehát a küldés és fogadás folyamatának minden részét érdemes titkosított közegben végezni, amelyre remek beépített megoldásokat kínál a Postfix, illetve az Stunnel4[67] program segítségével IMAP4/POP3 esetén is beállíthatjuk. Érdemes webmail elérés esetén ugyancsak HTTPS protokoll alatt tartani a teljes webmail forgalmat, hiszen a felhasználók a jelszavaikat és az érzékeny információkat küldik folyamatosan a szerver felé. Erre az OpenSSL+Apache tökéletes megoldást jelent.

Levélszemét és hamisítás elleni technológiák

DKIM[68]**:** a metódus megalkotói abból a tényből indultak ki, hogy minden bejegyzett domain mögött biztosan van DNS-szerver, hiszen máskülönben maga a levelezésre használt e-mail cím domain és host neve sem létezne, azaz az e-mail cím sem lenne valós. Ezen címek kiszűrésére pedig más technológiák hivatottak. Éppen ezért a DKIM megoldásában egy, a domain névhez legyártott kulcsot kell elhelyeznünk a DNS adatbázisban TXT rekordként. Ezen TXT rekordot mindenki le tudja kérdezni, hasonlóan ahogy egy sima névfeloldás esetében a host nevet. A TXT rekord által visszaadott kulcsnak egyeznie kell a küldő szervere által generált aláírással. Ez tehát nem kifejezetten a levélszemét elleni eszköz, hanem inkább egy kriptográfiai megoldás arra nézve, hogy egy idegen fél számára biztosítani tudjuk az automatikus, emberi beavatkozás mentes ellenőrzést arra nézve, hogy az adott küldő ténylegesen abból a domainből küldött-e levelet, amely a TXT alapján azonosította. Ez egy remek kiegészítés, amely mégis segít a levélszemét elleni küzdelemben, mivel nagy általánosságban a spammerek nem írják alá a leveleiket a fenti módszer segítségével. Valamint a nagy ingyenes szolgáltatók szinte minden esetben támogatják a fenti módszert, így enyhítve a saját levélszemétszűrőjük terheltségét is. Nagyon fontos ugyanakkor leszögezni, hogy minimum 1024 bit hosszúságú kulcsot érdemes gyártani és elhelyezni a TXT rekordban, mivel a kisebb kulcsot a mai rendelkezésre álló számítási kapacitások mellett könnyen lehet törni és így hamisítani is. A rendszer beüzemeléséhez a *dkim-filter* nevű csomagot kell telepítenünk, amely hasonlóan az Amavishoz, a Postfixszel vagy az egyéb MTA-val fog kommunikálni a levelek kiküldése esetén. A *dkim-genkey* parancs segítségével tudunk kulcsot generálni, majd azt a megfelelő helyen elhelyezni. A Postfix beállítása után már automatikusan aláírt levek fognak kimenni az SMTP szerverünkről. Beállításáról Postfix+Ubuntu szerver esetében érdemes az alábbi[69] leírást elolvasni.

SPF[70]**:** kis túlzással mondhatjuk, hogy a DKIM elődje. A tény az, hogy a technológiai alapjai hasonlóak a DKIM-hez, azonban a megoldása elég korlátozott és számos hibalehetőséget tartogat, használata nem is tanácsos. A lényege, hogy a már ismertetett TXT rekordban nem egy kulcsot helyezünk el, hanem azt mondjuk meg, hogy melyik szerver küldhet, milyen domain név alá tartozó leveleket. Ez a rendszer alapvetően nem veszi figyelembe, hogy a leveleinket egy másik SPF-et támogató szerverre is átirányíthattuk, így ott esetlegesen elutasításra kerülhet a nem megfelelő TXT rekord miatt. Előnye, hogy nem igényel plusz aláírást minden levél esetében, ezáltal az

67. https://www.stunnel.org/index.html
68. http://en.wikipedia.org/wiki/Dkim
69. https://help.ubuntu.com/community/Postfix/DKIM
70. http://en.wikipedia.org/wiki/Sender_Policy_Framework

SMTP-szerver erőforrásait kíméljük. A DKIM viszont sokkal összetettebb és kifinomultabb technológia, amely sokkal inkább alkalmas egyedi azonosításra, ezért az SPF használatát kifejezetten nem ajánljuk, csak a történeti áttekintés miatt szerepel itt.

Webmail

A webmail a felhasználói oldalról ha lehet az egyik legfontosabb tényező. A Google Mail, mint a legtöbb ember által használt webmail-felület magasra tette a mércét az összes webmail felületnek. Gyakorlatilag a piacon standard megoldásként ismerik el és mindenki igyekszik a sajátját igazítani hozzá. Természetesen a versenyt nem igazán lehet felvenni a világ egyik legnagyobb fejlesztőcégével, azonban nyílt forrású alapokon rengeteg jól használható és széles körű támogatást élvező megoldás született. A legtöbb megoldás a LAMP[71] környezethez készült, nem is véletlen, hiszen ez a leginkább elterjedt. Egy webmail kiválasztásánál a legfontosabb szempont, hogy olyan eszközt válasszunk, amely a felhasználók szokásait és igényeit veszi leginkább figyelembe, miközben az is fontos szempont, hogy a legtöbb PHP nyelven íródott webmail program sem biztonsági sem pedig terhelési kockázatot ne jelentsen szerverünk számára. Általánosan elmondható, hogy a webmailek postafiókonként 2-3000 levélig kezelik normálisan, azaz a felhasználók számára akadásmentesen a leveleket. Egyesek már jóval a 2000 levél/postafiók szám alatt elveszítik gyorsaságuk és nehézkessé válnak, esetleg nagy terhelést rónak az IMAP- és a webszerverekre is. Érdemes tehát már az elején a kiválasztás után tesztelni ezen programok együttműködését nagy terhelés sok levél mellett. Valamint kvóta rendszert bevezetni, vagy oktatni a felhasználókat az archív tárolók használatára.

A webmail rendszer szinte sosem jár csak magával a levelező felület kiválasztásával, hiszen a felhasználó itt akarja kezelni a címtárát és egyéb közös szolgáltatásait, a naptárfunkciókat, valamint a közös mappákat is. Jelen fejezet kifejezetten az egyszerű és gyors, de azért funkcionalitásban gazdag webmaileket tárgyalja, amelyeknek sok esetben a kibővített funkciók nem részei. Sok esetben a felhasználó a közös címtárat egy egész más felületen éri el, mint a leveleit, illetve semmi akadálya nincs annak, hogy egy szerveren akár több webmailt kínáljunk a felhasználónak. Egyet használhat, ha „csak levelezni” szeretne, egyet pedig, ha komplexebb feladatokat akar végrehajtani.

SquirrelMail:[72] egy igen régóta fejlesztett, nagy múltra visszatekintő webmail felület, amelyet teljesen PHP-ban írtak. HTML4 kompatibilis, használata nem igényel JavaScriptet. Teljes körűen támogatja az IMAP és SMTP protokollokat, rendkívül egyszerűen konfigurálható és kezelhető. Része a legtöbb Linux terjesztésnek, így rendelkezik biztonsági követéssel is. Standard bővítményillesztő felülettel rendelkezik, így ha a sok száz gyárilag hozzá adott, vagy a weboldaláról letölthető számtalan funkciót kínáló külső program számunkra nem elegendő, akkor könnyedén illeszthető hozzá szinte bármi egy PHP-programozó segítségével. Nagy felhasználói bázissal rendelkezik, és nagyjából lefedi egy kis és közepes vállalkozás igényeit is. Jól testre szabható a kinézete is, azaz akár céges logók és egyéb felületek is könnyedén kialakíthatóak. Paraméterei nagyban javíthatóak, ha a PHP futási értékeit igazítjuk a felhasználók igényeihez és a gép fizikai keresztmetszetéhez. A memóriafoglaltsági határ értékét és a maximális futási idejét kell emelnünk, de vigyázzunk ezekkel az emelésekkel, mivel rendszerünk sérülékenységét és támadhatóságát növelik. Fontos szempont, hogy teljes körű nyelvi támogatás létezik hozzá, köztük magyar nyelvű is. A biztonság

71. http://hu.wikipedia.org/wiki/LAMP_(szoftvercsomag)
72. http://www.squirrelmail.org/

iránt elkötelezetteknek jó hír, hogy letöltéskor a forráskód valódiságát (ha a terjesztésben lévőnél frissebb verziót akarunk használni) MD5, SHA1 és GPG aláírás metódusokkal is igazolják, ami biztosíthatja számunkra, hogy valóban azt töltjük le, amit a programozócsapat aláírt (GPG aláírás esetében). Gyakorlati tapasztalat, hogy a legtöbb frissítés után a felhasználó saját adatai (amely egyénenként egy szeparált DATA könyvtárban található) illeszkednek az új verzióhoz, azaz nem szükséges semmilyen beavatkozás a frissítések után.

Roundcube:[73] tudásban hasonló, mint a Squirrelmail, sőt mivel tudása ugyanolyan könnyen bővíthető bővítmények segítségével, ezért igazán egy súlycsoportban indulnak. A kezelőfelület 70 nyelvhez rendelkezik fordítással (köztük a magyarhoz is), gyors és igen könnyen kapcsolható LDAP- illetve SQL-szolgáltatásokhoz. Kezeli a megosztott IMAP-mappákat is. Szintén nagy felhasználói bázissal rendelkezik de a visszajelzések alapján elmondható, hogy inkább a fejlesztői weboldalról érdemes az aktuális friss kiadást használni, mint a terjesztésekbe csomagolt valamivel régebbi verziót. Gyakorlati tapasztalat, hogy a nagy levélszámú postafiókokkal nehezen boldogul, csakúgy mint a SquirrelMail, valamint esetleírások említik, hogy verziófrissítésnél a felhasználók egyéb adatai (naptár, címlista) néha rendszergazdai igazítást igényelnek a migráció után.

Zimbra:[74] egy nyílt forrású naptár, webmail és csoportmunka-támogató szoftvercsomag. Igen nagy tudású, és rendkívül szélesen illeszthető eszköz. Sajnos azonban a szabadon felhasználható része erősen butított, főként a csoportmunka-támogatás és a harmadik fél által írt szoftverek tekintetében. Ugyanakkor az egyik legfejlettebb keresési lehetőségeket nyújtja pl. nagy postafiók esetén, illetve az extrém nagyságú postafiókokat is könnyedén kezeli. Komoly hátránya, hogy csak fizetősen vehető igénybe jó néhány szolgáltatása. Ugyanakkor az ingyenes verzió is jól használható, ha tisztában vagyunk a korlátaival.

OpenExchange:[75] egy igen komplex szolgáltatásokat felvonultató webmail, naptár, közös mappa megosztásra is alkalmas eszköz. Nagy múltra tekint vissza és igen funkciógazdag. Az ingyenes GPL-es változat számos funkciót nem tartalmaz (pl. a mobiltelefonos részletes szinkront a legnépszerűbb telefonokra), de az alap funkcionalitás is kiválóan használható. A webes felület jelenleg nem szabad szoftver[76]. Nagy felhasználói bázissal és magyar lokalizációval is rendelkezik. Fontos szempont, hogy saját csomagkezelő szervere van, így a kiadások követése jól megoldott és gyakorlatilag terjesztésfüggetlen.

Zarafa:[77] GPL licenc alatt elérhető komplex és könnyen kezelhető csoportmunka-támogató rendszer. Rendelkezik minden olyan tulajdonsággal, amely szükséges: webmail, naptár, közös mappák kezelése, teendők stb. ActiveSync kompatibilis, illetve MAPI-konform, amelynek köszönhetően az egyik leginkább rugalmasan használható MS Exchange-kompatibilis eszköz. Hátránya, hogy az ingyenes verzió csak 3 db. Outlook klienst támogat, e szám fölötti Outlook kliens igény esetén licencek vásárlására kell számítani. (Ezen kívül a jelenlegi magyarítása kívánnivalókat hagy maga után.)
SOGo:[78] jól skálázható csoportmunka-támogató szoftver GPL-licenc alatt kiadva. Nagy előnye, hogy 2003, 2007 és 2010-es natív Outlook támogatással rendelkezik. Tudása nagyon hasonló a többi komplex megoldásokat felvonultató eszközhöz. A felhasználói visszajelzések szerint jelenleg még az elején tart a fejlesztési ciklusnak, azaz jól használható, de néha lassan reagál. Rendelkezik magyar nyelvű támogatással is.

73. http://roundcube.net/
74. http://www.zimbra.com/downloads/os-downloads.html
75. http://www.open-xchange.com
76. http://en.wikipedia.org/wiki/Open-Xchange#Licensing
77. http://www.zarafa.com
78. http://www.sogo.nu/

A webmail tipikusan egy olyan eszköz, amely könnyen cserélhető, illetve tapasztalati úton érdemes tesztelni, hogy az adott rendszerben melyik használható jobban. Ennek ellenére, ha már választottunk és az adatok gyűlnek benne, akkor a migrációs probléma miatt (eltérő SQL-táblák, stb.) érdemes egy tesztidőszak után maradni a legjobb választásnál. Abban az esetben, ha csak kifejezetten webmail és naptár funkcionalitásra van szükség, akkor a SquirrelMailt érdemes használni. Ha csoportmunka-támogatás, vagy bővebb funkcionalitás a cél, akkor sorrendben a Zarafa, illetve az OpenExchange használata javasolt.

Általános célú hálózati szerver: az inetd és xinetd

A hálózati szolgáltatásokat nyújtó szoftverek jellemzően olyan háttérben futó programok, amelyek az operációs rendszer és a függvénykönyvtárak átviteli-rétegbeli absztrakcióit használva hallgatnak egy-egy TCP vagy UDP porton, majd ha egy ügyfél kapcsolatot kezdeményez, azt egy másik szálon vagy folyamatban szolgálják ki. Könnyen belátható, hogy minden hasonlóan működő szolgáltatás folyamatosan erőforrásokat – különösen memóriát – használnak, valamint azonos funkciókat kell a fejlesztőknek megvalósítaniuk, felelősségi körükben tartaniuk.

Ezen felismerés nyomán jött létre az inetd, az „internetes szuperszerver”. Az általános célú hálózati kiszolgáló egy olyan szolgáltatás, amely egyszerre hallgat több TCP és UDP porton, több szolgáltatást kiszolgálva, majd ha a portok valamelyikére kérés érkezik, megválaszolására egy új folyamatban elindítja a konfigurációnak megfelelő programot, amelynek egyrészt a hálózati kommunikáció részleteivel már nem kell foglalkoznia, másrészt a programot csak lekérés esetén kell elindítani.

Ezzel a ritkábban használt szolgáltatások erőforrást, folyamatokat takaríthatnak meg, azonban teljesítménye a modern kiszolgálóalkalmazások nagy terhelésre optimalizált párhuzamos futtatási megoldásaival ritkán mérhető össze.

Használata nagy forgalmú, komplex kiszolgálófeladatok futtatására napjainkban nem jellemző, azonban egyes egyszerűbb, ritkán vagy kevés ügyfél által használt szolgáltatások – tftp, ident, nrpe, git – vagy prototípus-rendszerek futtatására, valamint beágyazott rendszereken gyakran alkalmazzák.

> A démonként futni képes hálózati szolgáltatások maguk végzik a beérkező hálózati kapcsolatok fogadását és a kezelésükhöz szükséges folyamatok vagy szálak indítását, életciklusuk felügyeletét. Ennek a helyes megvalósítása nem könnyű feladat – sok esetben erre nincs is szükség, mivel pontosan ezt a felelősséget veszi át az alkalmazástól az inetd: az alkalmazásnak csupán egyetlen kapcsolatot kell feldolgoznia, és különleges hálózati feladatai sincsenek, a kommunikáció egyszerűen a szabványos ki- és bemenetén keresztül zajlik. Nehézkesen működik azonban ez a megoldás nagy számú kapcsolat esetén, mivel nincs lehetőség az adott szolgáltatásra optimalizált erőforrás-takarékos párhuzamosítási technikák használatára.

Az általános célú hálózati kiszolgálónak két ismert megvalósítása a hagyományos inetd és továbbfejlesztett változata, a xinetd.

Inetd

A szolgáltatás eredeti, BSD-ből származó megvalósítása az inetd. Hosszú története miatt számos változatban érhető el, azonban a GNU/Linux rendszereken jelentős részben kiszorította fejlettebb változata, a xinetd.

Debian és Ubuntu rendszereken az openbsd-inetd nevű csomagokban találjuk meg támogatott változatát, míg a CentOS tárolói nem is tartalmazzák.

Az inetd indítása és leállítása a szokásos módon történik, például a fenti csomag esetén a sudo /etc/init.d/openbsd-inetd start, valamint a sudo /etc/init.d/openbsd-inetd stop parancsokkal. Ha nincs szolgáltatás konfigurálva, az inetd nem indul el.

Amennyiben egy szolgáltatást az inetd-vel szeretnénk kiszolgálni, ellenőrizzük, hogy az adott szolgáltatás szerepel-e a TCP- és UDP-portok központi nyilvántartásában, a /etc/services fájlban. Ez a fájl egy szimbolikus elnevezéshez egy portszámot és egy protokollt ad meg, majd szükség esetén álneveket sorol fel.

```
tftp            69/udp
  nrpe          5666/tcp                # Nagios Remote Plugin Executor
```

Ha a listában nem szereplő portot szeretnénk használni, ellenőrizzük, hogy nem szerepel-e más néven az adott port, majd vegyük fel a listába.

Ezután nyissuk meg a /etc/inetd.conf konfigurációs fájlt. Ebben alapesetben egy szolgáltatás sem szerepel, csupán # jellel kezdődő megjegyzések, amelyek röviden bemutatják a fájl formátumát. A fájlban egy szolgáltatást egy sorban, a mezőket szóközökkel elválasztva kell leírnunk, a következő sorrendben:

– Szolgáltatás neve, ahogy a /etc/services-ben szerepel (például nrpe).
– Socket típusa: stream (TCP esetén), dgram (UDP esetén).
– Átviteli-rétegbeli protokoll: tcp vagy udp – ahogy a /etc/services-ben szerepel. Ha a szolgáltatás tcp-vel és udp-vel is szerepel, válasszunk, vagy két külön sorban, mindkét változatban konfiguráljuk.
– Kapcsolók: wait vagy nowait – várja, vagy ne várja meg az előző kiszolgálófolyamat kilépését újabb szolgáltatás kiszolgálásához. Tipikus beállítása UDP esetén wait, TCP esetén nowait.
– Felhasználó: a szerverprogram ezen felhasználó (vagy user:group formában megadva ezen felhasználó és csoport) nevében kerül futtatásra. Lehetőség szerint ne adjuk meg a root felhasználót.
– Szerverprogram útvonala: a szerverprogram futtatható fájljának abszolút elérési útvonala (például /usr/sbin/nrpe).
– Argumentumok: a programnak átadandó parancssor – a program neve és az argumentumok (például nrpe -i -c /etc/nagios/nrpe.cfg).

Amennyiben az inetd használatára felkészített szoftvert használunk, a dokumentációjában meg fogjuk találni a javasolt beállítást, és a csomagkezelők is gyakran elvégzik a szükséges konfigurációt: például Ubuntu alatt a tftpd csomagot telepítve a /etc/inetd.conf fájlban megjelenik a következő sor:

```
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /srv/tftp
```

Ha a kívánt szolgáltatás nem szerepel még a konfigurációs fájlban, vegyük fel a leírását az ismertetett formátumban.

Ezután indítsuk el a kiszolgálót, vagy töltsük újra a beállításait, például a sudo /etc/init.d/openbsd-inetd reload paranccsal.

TCP porton hallgató kiszolgáló is használható az inetd-vel. Például a Nagios állapotmonitorozó rendszerhez állapotinformációt szolgáltató nrpe ágens beállításához helyezzük el a /etc/inetd.conf-ban a következő sort:

```
nrpe stream tcp nowait nagios /usr/sbin/nrpe nrpe -c /etc/nagios/nrpe.conf -i
```

Az így létrehozott szolgáltatás az nrpe (5666/tcp) portra csatlakozó kliensek kéréseit az nrpe nevű programmal szolgálja ki, amely az argumentumoknak megfelelően a megadott konfigurációs fájlt használja, és inetd üzemmódban indul. Ki is próbálhatjuk a szolgáltatást az /usr/lib/nagios/plugins/check_nrpe -H localhost -c check_load paranccsal.

Xinetd

Az inetd egyszerűsége miatt számos, főleg biztonsági elvárásnak nem tudott megfelelni. Emellett konfigurációja nehézkes, és új funkciók bevezetésével tovább bonyolódna. Ezekre a problémákra ad választ a xinetd.

Alapvető funkciói megegyeznek az inetd-vel, azonban könnyebben kezelhető konfigurációs formátumot használ, és olyan beállításokat is meg lehet adni az egyes szolgáltatásokra, mint hogy melyik interfészen hallgassanak, mely lekérésekre válaszoljon, vagy mely eseményeket naplózza.

Konfigurációs fájlja a /etc/xinetd.conf, azonban alapbeállításban a /etc/xinetd.d/ könyvtárban lévő fájlokat tartalmát is hozzáfűzi konfigurációjához.

A fájl szolgáltatásdefiníciókból áll, amelyek a service kulcsszóból, a szolgáltatás nevéből, majd kapcsos zárójelek között kulcs=érték formában megadott paraméterekből állnak. A fentebb ismertetett nrpe-szolgáltatás xinetd formátumú leírása például:

```
service nrpe
  {
   socket_type      = stream
   wait = no
   user = nobody
   server     = /usr/sbin/nrpe
   server_args      = -i -c /etc/nagios/nrpe.cfg
  }
```

A fontosabb beállításokat az 1. táblázat mutatja be.

Paraméter	Leírás	Példák
socket_type	Socket típusa.	stream (TCP esetén) dgram (UDP esetén)
protocol	Átviteli-rétegbeli protokoll – ahogy a /etc/services-ben szerepel. Ha a szolgáltatás tcp-vel és udp-vel is szerepel, válaszszunk, vagy két külön definícióban, mindkét változatban konfiguráljuk. Elhagyható.	tcp udp
wait	Várja (yes), vagy ne várja meg (no) az előző kiszolgálófolyamat kilépését újabb szolgáltatás kiszolgálásához. Tipikus beállítása UDP esetén yes, TCP esetén no.	yes no
user	Felhasználó: a szerverprogram ezen felhasználó nevében kerül futtatásra. Lehetőség szerint ne adjuk meg a root felhasználót.	nobody
server	Szerverprogram útvonala: a szerverprogram futtatható fájljának abszolút elérési útvonala.	/usr/bin/git
server_args	A programnak átadandó további argumentumok.	daemon --inetd –export-all /home/git/repo
cps	Kapcsolatok másodpercenként. Két számot vár, az első megadja, hogy másodpercenként hány kapcsolat után függessze fel a kiszolgálást, a második számmal megadott másodpercre.	50 10
per_source	Forrás IP címenként a megadott számú kapcsolatot kezel egyidejűleg.	10
disable	A szolgáltatás tiltása (yes esetén).	yes no

1. táblázat: A xinetd szolgáltatások paraméterei.

CentOS alatt a tftp-server csomagot telepítve a /etc/xinetd.d/tftp konfigurációs fájl jön létre a következő tartalommal:

```
service tftp
  {
        socket_type       = dgram
        protocol          = udp
        wait              = yes
        user              = root
        server                  = /usr/sbin/in.tftpd
        server_args       = -s /var/lib/tftpboot
        disable                 = yes
        per_source        = 11
        cps               = 100 2
        flags             = IPv4
  }
```

Központi időszinkron szerver: NTP szerver

Történeti áttekintés

A számítógépes hálózatok korai időszakában a gépek órájának szinkronban tartására egyszerű eszközöket használtak. Mivel sok gép nem tartalmazott beépített órát, a gép bekapcsolásakor az automatikusan lefutó alkalmazások egyike a rendszergazdától megkérdezte a helyi időt, és a továbbiakban az adatot begépelő személy karórájától függött a pontosság. Ez meglehetős pontatlanságot eredményezett. (Ráadásul az idő múlását az operációs rendszer tartotta számon, ami szintén okozhatott időcsúszást.)

Viszonylag hamar kitaláltak olyan megoldást (RFC-868[79] [80]), amely TCP/IP protokollt használó gépek számára akár TCP, akár UDP használatával lehetővé tette egy központi géptől a pontos idő lekérdezését. Ez az idő már nem lokális, hanem a Greenwich-i középidő (GMT) szerinti volt, ezt kellett a helyi gépnek az időzóna információ helyes beállításával korrigáltan mutatni a felhasználók számára.[81] Ez másodperc pontosságot biztosított, és a használt adatformátum miatt 2036-ig nyújt lehetőséget a használatra. Ez a megoldás már 1-2-re csökkentette egy hálózat esetén a kézzel beállítandó gépek számát, és növelte a pontosságot. Mivel a TCP/IP hálózat UNIX-rendszerekbe integrálásának kezdete óta a legtöbb rendszeren alapértelmezetten volt és futott ez az időszerver szolgáltatás (tipikusan az inetd nevű eszköz nyújtotta time néven), elterjedt volt a gépek szinkronizálása egy rdate nevű parancs segítségével; használata például `rdate time.kfki.hu` formában történt.

Ez a megoldás több problémát is felvet[82]. Ezért újabb protokoll-leírás jött létre (RFC-958[83], RFC-1059[84] NTP – azaz Network Time Protocol néven), amit hamarosan több másik követett (RFC-1119, RFC-1305, RFC-1361, RFC-1769, RFC-2030, RFC-4330, RFC-5905) – egyrészt a különféle továbbfejlesztett verziók (jelenleg v4-nél tartunk), illetve az NTP részhalmazát leíró SNTP (Simple Network Time Protocol). Az NTP segítségével lokális hálózaton akár 1 milliszekundumos pontosság is elérhető, interneten keresztüli szinkronizáció esetén ez a 10 msec nagyságrendű pontosságra csökkenhet.

Fentebb emlegetett (S)NTP kliens oldali megvalósítására bizonyos rendszereken használható az előzőekben már emlegetett rdate parancs a kifejezetten erre szolgáló „-n” opció segítségével

79. http://tools.ietf.org/html/rfc868
80. Párja volt az RFC-867, (daytime) mely a helyi időt szöveges formában tette távolról lekérdezhetővé. De mivel a formátumra csak ajánlás szerepelt a szövegben, így használata inkább hibakeresésre korlátozódott
81. Ma a Unix és Unix-szerű (például Linux) rendszerek használata esetén kifejezetten javasolt a gép fizikai óráját a régebben GMT, újabban UTC néven emlegetett - nem-lokális - idő szerint beállítani. Ha egy gép órája helyi idő szerint jár, az az időzónák, illetve a téli-nyári óraátállítás miatt problémákat vet fel.
82. A protokoll nem veszi figyelembe az adatcsomag áthaladásának időtartamát - ez főleg távoli szerverek, lassú adatátvitel esetén jelenthet problémát; a 2036. év utáni időre nem nyújt megoldást - ez nyilván ma még nem kritikus, de foglalkozni kellett vele; és végül a másodperces pontosság sok esetben már nem elegendő.
83. http://tools.ietf.org/html/rfc958
84. http://tools.ietf.org/html/rfc1059

(http://www.openbsd.org/cgi-bin/man.cgi?query=rdate&sektion=8), ellenben Linux rendszereken az rdate ezen funkciója hiányzik.

Döntési mechanizmus

Gyakorlatilag nincs választási lehetőség, minden ma elterjedt rendszer (S)NTP-t használ. Két elterjedt megvalósítása a http://www.ntp.org/ referenciaszoftvere, illetve az OpenBSD csapat által fejlesztett OpenNTP (http://www.openntpd.org/). Ez utóbbinak a fejlesztése az utóbbi 2-3 évben meglehetősen lelassult, ráadásul a legtöbb Linux-terjesztés az ntp.org-félét kínálja fel alapértelmezetten, e mellett maradtunk.

Használat

Ma már a hálózatba kötött számítógépek óráját javasolt (bizonyos helyen kötelező) szinkronban tartani. Ugyan ma már a legtöbb számítógép rendelkezik beépített óracsippel, de ezek az órák meglehetősen nagy szórást mutatnak a minőség tekintetében. Így az általában használt megoldás az, hogy kis számú gép esetén 1, nagyobb géppark esetén 2 esetleg 3 központi gép órájához igazítjuk a hálózat többi gépét. A szinkronizálást ma már általában a fent említett NTP segítségével oldják meg.

Kliens

Kliens oldalról két elterjedt módszert szoktak használni:

– a napi rendszerességgel ki- és bekapcsolt gépek esetén, közelítőleg megfelelő eredményt érhetünk el egy a gép bekapcsolásakor automatikusan lefutó, a gép óráját a központi órához igazító parancs segítségével. Ez az ntpdate parancs, használata: `ntpdate -b time.kfki.hu` Míg a „-b” opció[85] használata javasolt, addig a time.kfki.hu helyett inkább válasszunk valami nekünk megfelelőt, ilyen lehet például a hu.pool.ntp.org (Magyarországról - más ország esetén javasolt azon ország országkódját használni a „hu” helyett).
– amennyiben a gép folyamatos üzemben működik (a szerverek tipikusan ilyenek, de kényelmi okokból sokan a munkaállomásokat se kapcsolgatják), a bekapcsoláskor történő órabeállítás kevés lehet. Jellemzően a gépek órája sietni, vagy késni fog egy idő után. Ebben az esetben a javasolt használat az, hogy az órát folyamatosan igazítsuk a központi órához. Ellenben az, hogy meghatározott időnként (például óránként, naponta) **át**állítjuk a gép óráját, felvet egy nagyon súlyos problémát - ilyen esetben elveszítjük az idő egy vagy több fontos jellemzőjét - a folytonosságot, illetve a monotonitást. Azaz vagy „luk” keletkezik - kimaradó időpont; vagy ugyanaz az időpont többször is bekövetkezik. Ezért a bevált módszer a régi vekkerórákban alkalmazott „óralassítás” / „óragyorsítás”. Azaz lekérdezzük a központi gép óráját, és ha a miénk késést mutat, akkor felgyorsítjuk, ha pedig a miénk siet, akkor lelassítjuk azt.

85. jelenleg a „-b” opció által kért funkció az alapértelmezett, így akár el is hagyható, de hosszú távon preferált a használata (láttunk már szoftverben alapértelmezést megváltozni, kicsit sűrűbben, mint opciót)

– Ezt szintén megtehetjük az ntpdate parancs segítségével, ekkor ki kell hagyni a fent említett „-b” opciót, helyette a „-B” használandó: `ntpdate -B time.kfki.hu` és ezt a parancsot kell óránként / napi rendszerességgel lefuttatni (tipikusan **cron**-ból időzítve). Ily módon futtatva az ntpdate-et, a gép szépen lassan „hozzákésik/hozzásiet” a központi időhöz.
– Komplexebb konfigurációt igényel az ntpd (régebben xntpd) nevű szoftver, ami elindítása után folyamatosan fut, és rendszeresen (alapból 64 másodpercenként, ez 16 másodperc és 36,4 óra között változtatható) lekérdezi a pontos időt, majd az előbb tárgyalt módon, a gép belső órájának felgyorsításával/lelassításával hozza összhangba a gép óráját a külvilágéval. Működéséhez viszont szükséges létrehozni egy konfigurációs fájlt (tipikusan /etc/ntp.conf néven létezik), amelyben minimálisan a lekérdezendő szerver(ek) nevét kell megadni, „`server time.kfki.hu`” formában.

Nem árt tudni, hogy jópár szoftver (például a Dovecot IMAP-szerver, vagy általában az adatbázis-kezelők) rendkívül zokon veszi az „időugrást” – ezek miatt szintén nem javasolt az a fajta óraállítás.

Megjegyzendő, hogy az óra folyamatos szinkronban tartásának két módja (ntpdate -B vs ntpd) biztonsági szempontból is különbözik. ntpdate használatakor csak a parancs futtatásakor lesz egy rövid ideig tartó hálózati forgalom, míg ntpd használata esetén az ntpd szerverprogram folyamatosan fut és a 123-as UDP porton keresztül elérhető. (Természetesen ez utóbbi használat esetén van lehetőség a kliensek korlátozására.)

Szerver

Szerver, amelyikhez a többi gépet szinkronizáljuk. Ezen gép órájának pontos beállítására több módszert használhatunk. A leggyakoribb, hogy megkérdezzük valamely másik szervert. Ehhez bevezetünk egy hierarchiát. A hierarchia csúcsán álló gép műholdaktól, célhardver segítségével kapja a pontos időt. Ezeket a gépeket Stratum-0 pontosságú szervernek hívjuk; azok a gépek, amelyek ezektől kapják a pontos időt, azok a Stratum-1-es szerverek. Tőlük kapják az időt a Stratum-2-esek, és í. t. (Noha a szabvány elvileg megkülönböztet 256 szintet, a gyakorlatban már egy Stratum-15-ös gépet is nem-szinkronizált órájúnak tekintünk.) Az időinformációt a szerverek TCP/IP protokollon broadcast vagy multicast címeket használva szórják a hálózatban, illetve lehet kérdezni is a szervereket. Attól függően, hogy mi kérdezzük a szervert, vagy csak hallgatni akarjuk a tőle érkező szórt információt, változik a konfiguráció a /etc/ntp.conf fájlban. (A következő példákban feltételezzük, hogy a 192.168.1.42.0/24-es hálózatban dolgozunk, ahol az NTP-szerver a 192.168.42.1-es címen érhető el.)

– üzenetszórásos szerver, hallgatózó kliensekkel

Itt a szerver konfigja:

```
szerver$ cat /etc/ntp.conf
server hu.pool.ntp.org
broadcast 192.168.42.255
```

Itt a kliens konfigja:

```
kliens$ cat /etc/ntp.conf
broadcastclient
```

– lekérdezős szerver, kérdezősködő klienssel

Itt a szerver konfigja:

```
szerver$ cat /etc/ntp.conf
server hu.pool.ntp.org
```

Itt a kliens konfigja:

```
kliens$ cat /etc/ntp.conf
server 192.168.42.1
```

Nagyon sok helyen megelégednek azzal, hogy valamely gépet kinevezik óraszervernek, ennek a belső hardveróráját kézzel beállítják, és ehhez szinkronizálják az összes többit. Ebben az esetben például ez a megfelelő beállítás:

```
gép.belső.órájáról.szinkronizáló.szerver$ cat /etc/ntp.conf
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Látható, hogy a szervergépen egy elég furcsa, normálisan nem használatos IP-címet adtunk meg az idő forrásaként. A 127.127.T.U típusú címet a szoftver speciálisan kezeli és azt jelenti, hogy valamilyen hardveren keresztül kapja meg a pontos időt (típusát és a konkrét darabot adják az IP-címbeli számok: T – typeID, U – unitnumber). A „fudge” sor pedig azt jelzi, hogy az alapértelmezett 0-s stratum szint helyett ennél jóval alacsonyabb pontosságúnak minősítjük az idő forrását (és ennek következtében alacsony pontosságúként hirdetjük) – ezzel elérve azt, hogy ha egy kliens gép rajtunk kívül valahonnan máshonnan is kapna pontosabb időinformációt, ne a pontatlan időt használja.

Az NTP protokoll nagyon sok lehetőséggel rendelkezik, mint korábban utaltunk rá, például a kliensek és szerverek a kommunikáció során akár azonosíthatják is egymást. Ez biztonsági szempontól nem elhanyagolható lehetőség, hisz e nélkül könnyen lehetne hamis információkkal megzavarni a rendszer működését. A további információk, az egyes beállítások a használt szoftver kézikönyvében megtalálhatók.

Névfeloldás a hálózaton: DNS szerver (dnsmasq és BIND)

A ma elterjedt IP-alapú lokális hálózatok használata esetén az egyes gépek fizikai kommunikációjához szükséges, hogy egyedi IP-címük legyen, a felhasználók életét pedig nagymértékben megkönnyíti, ha nem a számukra megjegyezhetetlen IP-címekkel, hanem emberek által értelmezhető formában (például jelentést hordozó tartalommal bíró néven) hivatkozhatnak a gépekre. Az első funkciót általában a DHCP-szerver, a másodikat pedig egy DNS-szerver beállításával lehet elérni.

Minden egybe

Dnsmasq

Ahogy a mondás tartja: a BIND azoknak való, akiknek saját ISP-jük van. Noha ez így erős túlzás, átlag földi halandó számára a dnsmasq is tökéletes választás a pár (tíz) gépes hálózathoz. A beépített DHCP, DNS és TFTP funkciókon kívül a dnsmasq használatának nagy előnye, hogy ezen alapfunkciók konfigurálásához használhatók a rendszer eleve meglevő, egyszerű szintaxisú (és általában már amúgy is jól ismert formátumot használó) rendszerfájljai.

Használata egyszerű, kezdeti teszteléshez elegendő a szükséges paramétereket a parancssorban megadni. Természetesen a mindennapi használathoz ez nem túl szerencsés megoldás, ritkaság, hogy az ember csak annyi paramétert szeretne beállítani, amennyit fejben tud tartani. Általában éles üzemben ezeket a paramétereket nem a parancssorban, hanem egy központi konfigurációs fájlban szokták megadni. Ezt is igen egyszerűen oldotta meg a fejlesztő: a parancssorban egy paraméter megadásakor vagy a UNIX-világból örökölt rövid, egy karakteres opciókat használja az ember (ezekkel sajnos a fő gond, hogy nem túl sokan tudnak értelmetlennek, logikátlannak tűnő „rövidítéseket” megtanulni, ráadásul nem is mindennek van ilyen rövid formája); vagy használhatjuk a GNU/Linux világában elterjedt ún. hosszú opciókat – a hosszú opciók használatának nagy előnye, hogy ugyanazt kell a konfigurációs fájlba is írni, az apró eltérés, hogy a bevezető -- nélkül. (Ezek a hosszú opciók néha nem annyira logikusak: például -d a debug – hibakereső – mód rövid formája, de a hosszú formája már --no-daemon a parancsssorban, míg a konfigurációs fájlba no-daemon formában írandó).

Az előzőek alapján egy kisebb hálózat DHCP és DNS-igényének kiszolgálására legegyszerűbb esetben elegendő a következő beállítás:

A /etc/hosts fájlba felvesszük a hálózat összes gépét, a /etc/resolv.conf -ba a 127.0.0.1 címet – ezen a címen a dnsmasq figyeli a kéréseket, a dnsmasq-nak pedig parancssorból megadjuk, hogy milyen címtartományt kell kiosztani a DHCP-klienseknek, valamint megadjuk a külső névszerverek címeit, akiket a nem helyi adatok lekérdezéséhez meg kell kérdezni, és készen is vagyunk.

```
dnsmasq --dhcp-range=192.168.1.100-192.168.1.200,10h --local=/pelda.hu/ --no-
  resolv --server=192.168.42.42
```

Fenti példában a --dhcp-range opció a paraméterével a DHCP-klienseknek szolgáltatott címtartomány megadását teszi lehetővé tíz órás elévülési idővel; a --local jelzi, hogy az adott tartomány-

névért a dnsmasq a felelős, és nem kell a külső szervernek elküldenie az ilyen irányú kéréseket (csak a /etc/hosts fájl és a DHCP adatbázisa alapján válaszoljon); a --no-resolv azt jelenti, hogy a /etc/resolv.conf fájlt a dnsmasq-nak már nem kell feldolgoznia; végül a parancssori --server adja meg a külső névszerver címét. (A --local és a --server opciók amúgy szinonimák, és a fenti két példánál bonyolultabb beállításokat is meg lehet adni, pl. különböző tartományokhoz különböző szervereket kérdezzen.) Fenti paramétereket persze betehetjük a /etc/dnsmasq.conf fájlba is (oda ugye -- nélküli formában), és akkor a későbbiekben nem kell kitalálni és kézzel megadni őket.

Külön-külön

Névfeloldás

A TCP/IP hálózatok elterjedésének kezdetén sem a gépek nevének, sem a használatos IP-címeknek a kiosztása semmilyen módon nem volt központosítva. A fejlődés első lépcsőjében előbb központilag menedzselt HOSTS.TXT fájlban tárolták az adatokat, majd amikor ennek karbantartása (és főleg a naprakészen tartása) közelítette a kezelhetetlent, a hálózatos világ áttért egy elosztott adatbázis, az ún. DNS (Domain Name System) használatára.

Ennek a rendszernek fontos részét képezi a gépek nevének fa-struktúrába szervezett koncepciója, az adatok tárolásáért (és az adatbázisban történő keresések kiszolgálásáért) felelős névszerverek (nameserver), illetve a névfeloldást a felhasználói oldalon megvalósító függvénykönyvtár (resolver).

A névszerverek egy része a gépnév-IP-cím információk elosztott tárolásáért felelős, ezek a náluk tárolt adatok jogosult szolgáltatói (ún. autoratív névszerverek); másik részük nem tárol adatot (vagy csak egy bizonyos, lejárati időnek nevezett időtartamig), csak más (adatot tároló) szervertől elkéri azokat, és a kliensek számára átadja (ill. átmeneti tárban tárolja) – ezek a caching-only szerverek. A konfigurációtól (és az igényektől) függő módon egy névszerver nyújthatja ezek közül vagy csak az egyik, vagy akár mindkét szolgáltatást; a funkciókat bizonyos szoftverek akár egyedül is el tudják látni, mások csak egyik vagy másik felét végzik el.

Ma a feloldást végző függvénykönyvtár általában az operációs rendszer szerves részét képezi, így legfeljebb annak konfigurálása lehet kérdéses, ezzel szemben névszerverből nagyon sok megvalósítás létezik. (BIND, PowerDNS, MaraDNS, DJBDNS, NSD/Unbound, Dnsmasq, stb.)

Névtér

A névtér tartományokra (domain), résztartományokra (subdomain) illetve gépekre (host) osztható. Bizonyos határokon belül a tartomány résztartományának résztartományának résztartományának a … is értelmezett. Kis és nagybetűket nem különböztetünk meg, általában csak figyelemfelhívó céllal, vagy az olvashatóság javítása érdekében szokták a nagybetűs írásmódot használni.

A névtér csúcsán az ún. „root domain” található. Neve „.”. Az elnevezés és a vizuális ábrázolás általában ellentmondanak egymásnak, ugyanis tipikusan egy fölülről lefelé haladva egyre jobban elágazó – egy fa gyökérzetére hasonlító – ábrával szokták bemutatni a névteret, ennek ellenére csak a legtetején levő objektumot (és nem az egészet) nevezik root-domain-nek – azaz „gyökér-tartománynak”.

Ezen „root domain” alatt találhatók az ún. legfelső szintű tartományok (TLD, Top Level Domain). Ezek neve kötött. Egy része szabványos, kétbetűs országnév rövidítés (Magyarország: HU; Ausztria: AT; Amerikai Egyesült Államok: US – bár eléggé ritkán használják; és í. t.). Van néhány 3-betűs. Ezek nagy része valamilyen szervezet jellegét jelentő rövidítés (EDU: oktatási intézmény; GOV: kormányhivatal; MIL: valamilyen honvédelemmel kapcsolatos szervezet – ezek mind USA-beliek), akad néhány az USA-tól független elnevezés, mint a korábban amerikai, profit-orientált cégeket jelentő, ma már egyértelműen nemzetközi COM (commercial), vagy a non-profit szervezetek számára fenntartott ORG (organization), illetve az elsősorban (de nem kizárólagosan) internet-hozzáférést nyújtó szervezeteket azonosító NET. Időről időre jelennek meg újabb TLD-k, ez általában meglehetősen hosszan tartó procedúra eredménye. A teljesség igénye nélkül: BIZ, INFO, NAME, vagy a kevésbé ismer MUSEUM, COOP, PRO – ezek elfogadása 2000 november 16-án történt meg, de míg párat már 2001 júniusában aktiváltak, például a PRO csak 2004 júniusában lett használható. A lassú ügymenetre tökéletes ellenpélda az XXX top-level-domain, aminek elfogadása 2011 márciusában, használatba vétele ezzel szemben már 2011. április 15-én megtörtént. Az XXX névtartomány pozitív hatása[86] feltehetően akkor lesz érzékelhető, ha az ehhez a témakörhöz tartozó oldalak kizárólag csak ilyen neveken lesznek elérhetőek – erre egyébként nem sok esély mutatkozik.

Az egyes TLD-k alá már lehet regisztráltatni saját tartományneveket. Ezeket hívják első szintű tartománynak (first level domain). A nevekre nézve szoktak lenni korlátozások – mint például bejegyzett márkanevet tartománynévként csak a licenctulajdonos, vagy a tulajdonos írásbeli engedélyével rendelkező személy, szervezet regisztrálhat be; netán sértő, másokat megbotránkoztató kifejezések nem engedélyezettek – ezeket a korlátozásokat általában hosszabb-rövidebb idő után átlépik. (Vagy kikerülik, mert ha egy kifejezés magyarul csúnya lenne, és ezért esetleg a HU tartományba nem lehetne felvetetni, akkor még mindig rendelkezésre áll a COM, a NET, és í. t.) Nem minden TLD nyílt, például a fent említett MUSEUM alá – nem meglepő módon – csak múzeumok regisztrálhatnak.

Noha korábban elterjedt ajánlás volt, hogy TLD-ket nem használunk első szintű tartománynévként, ez egyre jobban feledésbe merül – és az egyre konkrétabb TLD-k, mint például az INFO, egyre kényelmetlenebbé is teszik ezt a korlátozást. Már korábban is sok ország próbálkozott azzal, hogy az „amerikai” mintának megfelelő, szervezet jellege alapú elnevezési konvenciót használjon (itthoni példa: gov.hu). Noha az egyes országokhoz tartozó, szabványos nevek kiválasztása független szervezet feladata, szerencsésebb országok – kis túlzással – GDP-jük meghatározó részét köszönhetik jól hangzó TLD-jüknek (Tonga-szigetek: TO, vagy Tuvalu: TV). Kisebb-nagyobb – elsősorban profitorientált – cégek vásárolnak és vásároltak náluk könnyen megjegyezhető nevet (például digi.tv, go.to).[87]

Az első szintű tartományok alatti elnevezési konvenció már a tartomány tulajdonosának belátásán múlik. Jellemzően kisebb szervezet esetén értelmes, megjegyezhető, egyszerű logikájú névkonvenció a szokásos, míg komolyabb, sok száz, ezer gépes környezetben már az automatizmus látszik a neveken. (Egyre elterjedtebb az ún. címrövidítők használata, ami a nagy cégek automatikusan generált, általában megjegyezhetetlen hosszúságú webcímeit legfeljebb 10-15 karakteres – amúgy szintén megjegyezhetetlen, és szintén automatikusan generált – rövidebb nevekre cseréli.)

A gépek hálózatban használt neve egy pontokkal elválasztott, több tagból álló azonosító, mint például prep.ai.mit.edu. Az adott példában az „edu” a TLD, a „mit” az első szintű tartománynév, az

86. nyilván könnyebb a tartalomszűrés, ha elegendő már az oldal nevének ilyen csekély részét ellenőrizni
87. De megemlíthető a rendkívül frappáns nevű yes.no, vagy a számítógépes biztonságra rendkívül sokat adó Daniel J. Bernstein (ő írta például a fentebb emlegetett djbdns-t) tulajdonában levő yp.to tartomány, aminek egyetlen résztartománynevét használja: cr.yp.to.

„ai” a második szintű, és a „prep” azonosítja magát a konkrét gépet. Tulajdonképpen a név hátulról előre haladva egyre jobban szűkíti a kört, hogy az adott gép szervezeti felépítés szerint (vagy ritkábban fizikailag) hol található. Az előző példa:

– az „edu” tag azt jelenti, hogy valamilyen oktatási intézmény (az Egyesült Államokban)
– az „mit” az amerikai Massachusetts Institute of Technology nevű intézményt konkretizálja
– az „ai” az egyetemen belül a mesterséges intelligenciával (artifical intelligence) foglalkozó részleg
– „prep” - ez pedig magának a konkrét gépnek neve

A későbbiekben még fontos lesz egy eddig nem említett TLD, az ARPA. Ide a DNS-infrastruktúra kezeléséhez szükséges speciális, ha úgy tetszik virtuális tartományok tartoznak. Régebbi az IN-ADDR.ARPA és újabb az IP6.ARPA. (Az első az IPv4-es, a második pedig az IPv6-os címek feloldásához szükséges adatokat tartalmazza – ez utóbbi tartományt korábban amúgy IP6.INT-nek hívták, régebbi dokumentumokban időnként lehet még ezzel a névvel találkozni.)

Resolver

A tipikus napi használat (böngészés, levelezés, esetlegesen a hálózat valamely erőforrásának elérése) során hálózati nevek IP-címekre fordítása a háttérben sűrűn előforduló feladat, ami a resolver-könyvtáron keresztül, a felhasználó számára észrevétlenül történik. (Ritkább, de szintén szükséges a fordítottja: az IP-hez tartozó névkeresés, ezt reverse-, vagy inverz névfeloldásnak nevezi a szakirodalom.) A névfeloldásért felelős szoftverkomponens bekonfigurálása elég egyszerű feladat, a /etc/resolv.conf nevű fájlban csak meg kell adni a használni kívánt névszerver IP-címét[88], és a tartománynév nélküli névkeresések esetén a keresendő gép nevéhez hozzáfűzendő tartományneveket.

```
search example.com hu.example.com example.hu
nameserver 192.168.42.44
nameserver 192.168.42.42
```

formában.

Nem árt tudni, hogy ma már az esetek jelentős részében a hálózaton DHCP-t használnak, amin keresztül a névfeloldáshoz szükséges paraméterek is átküldhetők, amely esetben a resolv.conf fájl kézzel való kitöltése kihagyható, sőt általában kifejezetten ellenjavallt, hiszen a DHCP-szervertől kapott információval felülíródik, így az általunk gondosan beállított paraméterek elvesznek. Szintén meg kell említeni, hogy névfeloldásra nem a DNS szerver az egyetlen lehetséges adatforrás. Kis gépszámú, statikus hálózat esetén sokáig alkalmazott módszer volt, hogy az adatokat egy egyszerű szöveges fájlban (hagyományos neve: /etc/hosts) tárolták, és minden gépnek saját példánya volt. (Hátulütője, hogy minden gépen kellett belőle egy példány, amelyeket minden módosításnál szinkronizálni kellett.) De nagyobb gépszámú hálózatnál, ahol a központosított adattárolás előnyei kiütköznek, ott sem feltétlenül a DNS az egyetlen lehetséges megoldás. Ezért szükséges lehet az NSS-keretrendszer konfigurálása. Az NSS (Name Search Switch vagy Name Service Search) adatkeresés rugalmas konfigurálására alkalmas szoftveregyüttes, amelynek a gépnév-IP-cím keresés csak az egyik feladata (mellette felhasználói név – UID keresés, csoportnév – GID és még sok más is a hatókörébe tartozik).

Kitérő az NSS-hez:

88. Csak IP-cím adható meg név nem, hiszen a név használata megint csak névfeloldást, így a névszerver IP-címének ismeretét követelné meg.

Az NSS-rendszert egy darab, /etc/nsswitch.conf nevű, egyszerű felépítésű konfigurációs fájl, és a keresést különböző módon megvalósító ún. NSS-modulok alkotják. A konfigurációs fájlban adható meg, hogy mely komponensek, milyen sorrendben próbálják meg a keresést elvégezni, míg az NSS-modulok végzik magát a keresést (az adatforrástól függő módon). A konfigurációs fájl felépítése egyszerű: a sor elején áll a keresendő adatbázis neve (ez sokszor megegyezik a hagyományos UNIX adatbázis /etc-könyvtárbeli fájlnevével, azaz felhasználói nevek esetén passwd, felhasználói csoportok esetén group, hálózati nevek/címek esetén hosts, és í. t.), majd kettőspont után az adatforrás azonosítására szolgáló kulcsszó, vagy ha több keresendő adatforrás van, akkor a megfelelő keresési sorrendben a kulcsszavak állnak. A kulcsszó valamely NSS modulra hivatkozik; a kulcsszó az adott keresést megvalósító osztott függvénykönyvtár nevére utal, ahol a függvénykönyvtár 32-bites Ubuntu 12.04.1 kiadás esetén a /lib/i386-gnu-linux/libMODULNEVE.so.x nevű fájlban található. A hagyományos, /etc-beli adatfájlban való keresést a „files” kulcsszó segítségével megadott modul intézi, azaz a /lib/i386-gnu-linux/libnss_files.so.2; az LDAP-szerveren való keresést az nss-ldap modul teszi lehetővé, ekkor a modul fájlneve /lib/i386-gnu-linux/libnss-ldap.so.2; az nss-dns modul a DNS rendszerbeli keresésre szolgál, a hozzá tartozó fájl pedig: /lib/i386-gnu-linux/libnss-dns.so.2; és í. t. Megfelelő modulokkal tetszőleges helyen és formában tárolt adatbázis is lehet a keresendő adatforrás. Azaz ha a név-IP-cím feloldást elsőként a hagyományos hosts-fájlban szeretnénk kezdeni, a bekonfigurált DNS-szerveren folytatni, végül pedig az LDAP-szerver adatbázisában keresnénk ha a korábbi adatbázisokban nincs találat, akkor

```
hosts: files dns ldap
```

a megfelelő beállítás.

Névszerverek

BIND (Berkeley Internet Name Daemon)

A BIND volt az egyik legelső UNIX-rendszerre írt DNS-szerver megvalósítás. Korábbi (az elterjedten használt 4-es és 8-as) verziói ma már elavultak, frissítések nincsenek hozzájuk, ezért ezek használata erősen nem javasolt. (Bizonyos – a 4-es verzióban használatos – elnevezési konvenciók viszont a mai napig megmaradtak.)

A BIND9 működéséhez alapvetően egy darab konfigurációs fájl és néhány „adatbázis” szükséges. A konfigurációs fájl írja le a névszerver funkcióját (elsődleges vagy másodlagos adatforrásként használható egy tartomány adatainak lekérésekor; esetleg csak gyorsítótár szerepe van); itt találhatók a működését befolyásoló paraméterek, a hozzáférés-szabályozás beállításai; valamint itt adjuk meg, hogy egy tartomány adatai hol találhatóak – amennyiben az adatok ennél a szervernél találhatók. A konfigurációs fájl köznapi neve „boot-fájl”[89] – noha named.conf néven szokott létezni. Eredeti helye a /etc könyvtár, de ezt az egyes disztribúciók előszeretettel változtatják meg.[90]

A tartományra vonatkozó információkat tároló „adatbázisokat” zónafájlnak hívjuk. A zónafájlokban ún. RR-ek (resource record, azaz erőforrás-rekord) találhatóak. Noha alapvetően név-IP-cím adatok tárolására (és lekérdezésére) készült a rendszer (az ide tartozó RR-ek: A, AAAA, PTR és esetleg a CNAME), ezeken az adatokon kívül a zónafájlokban találhatók egyéb információk is. Vannak például a rendszer helyes működését biztosító RR-ek (mint a SOA, vagy NS), vagy akár teljesen más területhez tartozó információk (például az MX, vagy az SRV).

89. a BIND4-ben a konfigurációs fájl neve named.boot volt, innen maradt a köznapi elnevezés
90. hogy bonyolultabb legyen az élet, a program indításakor megadható, hogy az eredeti könyvtár és név (/etc/named.conf) helyett hol és milyen néven keresse

A fő konfigurációs fájl és a zónafájlok is szövegesek, de jelentősen eltérő a szintaxisuk. A named.conf fájlban megadhatóak megjegyzések, akár a hagyományos, Unix-világban elterjedt:

```
# megjegyzés a sor végéig
```

akár a C++ vagy Java programozási nyelvből ismerős

```
// egysoros
```

vagy pedig

```
/*
ez itt
több soros
is lehet
*/
```

formában.

A konfigurációs fájlban minden egyes bejegyzést egy kulcsszó vezet be (mint például options, zone, key), ezt a kulcsszótól függően 0 vagy több kötelezően (és fix sorrendben) megadandó paraméter követi, majd pedig kapcsos zárójelek – „{“ és „}” – között az adott kulcsszóhoz tartozó paraméterek állhatnak. (Ezeket egymástól „;” választja el.) A teljes bejegyzést szintén a „;” karakter zárja le.

```
options { directory „/var/db/namedb”; listen-on { 127.0.0.1 ; }; };
zone „example.hu” { type master ; file „examplehu.db” ; };
```

Ezzel szemben a zónafájlok szerkezete sokkal kötöttebb. Csak egysoros megjegyzések lehetnek benne, azok is csak

```
; ez a megjegyzés a sor végéig tart
```

formában. Minden erőforrás rekord egy sorba írandó. Amennyiben valamely RR túl hosszú lenne, akkor az eredeti sorba egy „{“ karaktert írva (bárhova, ahol szóköz karakter állhat – például a sor végére) lehet több sorba tördelni, mely esetben a bejegyzést természetesen le is kell zárni a „}” karakterrel.

A sor elején[91] kötelezően áll egy név mező[92], melyet kötött sorrendben követ pár (opcionális) paraméter, az adott RR típusa, és típustól függően pár paraméter.

A fontosabb RR típusok:

SOA – a név mező által azonosított tartomány hitelesített adatainak kezdete (Start of Authority)

NS – az adott tartomány névszerverének meghatározása (NameServer)

A – az adott nevű gép IPv4-es címe (Address, IPv4)

AAAA – IPv6-os cím (Address, az IPv6-os címek 4-szer olyan hosszúak, valószínűleg innen az elnevezés)

PTR – a név mező által meghatározott IP-címhez milyen gépnév tartozik (Pointer)

CNAME – a név mező a gép „beceneve”, mi a hozzá tartozó hivatalos név (Canonical Name)

Egy BIND-szerver beállításának megértéséhez már csak pár információ szükséges. Hogyan történik egy gép nevéhez tartozó IP-cím megkeresése? Minden tartományhoz tartozik (egy vagy

91. a név a sor legelső karakterén kell kezdődjön. Ha a sor első karaktere szóköz vagy tabulátor, akkor a név mező hiányzik, mely esetben az adott bejegyzés örökli az előző bejegyzés név mezőjét
92. ez a név az RR-től függően vagy egy gép, vagy egy tartomány nevét jelenti

több) névszerver, amely az adott tartományon belüli kérések megválaszolásáért felelős. Ha ismert egy tartomány névszerverének neve és címe, akkor tőle a szükséges információ begyűjthető. Már csak ezeket az információkat kell valahogy megszerezni. A módszer nagyon egyszerű. A névhierarchiában egy szerver vagy ismeri az abba névtérbe tartozó összes információt, vagy valamely részét a névtérnek átadja másvalaki kezelésébe – ahogy mondani szokták, delegálja. De ehhez tudni kell az adott résztartományért felelős gép nevét és a hozzá tartozó IP-címet. Azaz a hierarchiában feljebb álló névszerver a közvetlenül alatta elhelyezkedő névszerverek adatait ismeri. Ezek a névszerverek szintén ismerik az alattuk levőket – ilyen módon felülről lefelé haladva el lehet jutni a minket érintő névszerverig.

A korábban említett „root domain" meglehetősen statikusnak tekinthető. (Nem túl sűrűn jelennek meg vagy tűnnek el legfelső szintű tartományok.) A névszerverei szintén nem túl sűrűn változnak. Ezen névszerverek – más néven a „root-szerverek" – ma már egy erre a célra fenntartott névtartományban vannak, ez a root-servers.net. Nevük is elég egyszerű: a.root-servers.net, b.root-servers.net, stb. m.root.servers.net-ig. Az ezekhez a nevekhez tartozó címek sem túl gyakran változnak, akár statikus információként bele is építhetők a névszerverünk adatbázisába.[93]

Ily módon a keresés végrehajtása egyszerű. Ha a korábbi példában szereplő prep.ai.mit.edu névhez tartozó címet keressük, akkor először valamelyik root-server géphez kell fordulni – hisz ők azok, akik a legfelső szintű edu névtartomány névszervereiről adatot tudnak szolgáltatni. Az edu-névszervertől megkapható az mit.edu névszerverének adata, tőle az ai.mit.edu adata. A legvégén a kezünkben van az adott tartományért felelős névszervernek a válasza, ami vagy a keresett névhez tartozó cím, vagy pedig az „ilyen adat nem létezik" hiba.

Látszólag sokkal ritkábban fordul elő fordított keresés, azaz egy IP-címhez tartozó név lekérdezése (a valóságban ez is meglehetősen gyakori). Ahhoz, hogy ez is könnyedén megoldható legyen, egy ügyes trükköt használunk. A gépek címe elölről hátrafelé haladva egyre specifikusabb. Elöl van a hálózatot azonosító pár bit, utána az alhálózatot azonosító néhány bit, és a cím végén állnak magát a konkrét gépet azonosító bitek. Ellenben a nevek pont fordítva néznek ki: elölről hátrafelé haladva egyre általánosabb információt tartalmaznak: nameserver.budapesti-hivatal.example.hu. A nevek keresésekor hátulról (az általánostól) haladunk előre, az egyre specifikusabb irányba. A trükk a következő: minden IPv4-címhez rendeljünk hozzá egy nevet, amit úgy kapunk, hogy megfordítjuk az IP-cím byte-jainak sorrendjét, majd ehhez ragasszunk hozzá egy speciális tartománynevet: IN-ADDR.ARPA (inverz address, fordított cím). Azaz például a 192.168.42.1 címhez rendeljük azt a nevet, hogy 1.42.168.192.in-addr.arpa. Ez a hozzárendelés kölcsönösen egyértelmű, tehát az IP keresése helyett kereshetjük ezt a nemlétező nevű gépet. Ha pedig a DNS adatbázisban az IN-ADDR.ARPA névtérbe tartozó adatoknál gépek IP-címe helyett a gépek valódi nevét tároljuk, máris eljutottunk a kívánt eredményhez – az eddigi IP-címkereséshez nagyon hasonló módon tudunk nevet keresni. Egy IP-címhez tartozó gépnév megkeresése tehát 2 lépésből áll: előállítjuk a fordított, IN-ADDR.ARPA névtérbe tartozó nevet, és megkeressük a fentebb már ismertetett módon a hozzá tartozó valódi nevet.

Ugyanezt a trükköt, egy kicsit módosítva használhatjuk a sokkal hosszabb IPv6-os címek keresésénél is. A módosítás lényege, hogy az IPv6-címet nem decimális, hanem hexadecimális[94] formában ábrázoljuk, és nem az egyes byte-okat, hanem az egyes félbyte-okat (azaz magukat az egyes hexadecimális számokat) használjuk az aldomainek megadására. Például ha egy gép IPv6-os címe 2001:db8::567:89ab , akkor az ehhez a címhez tartozó reverz bejegyzés (amely a korábbiak alapján az IP6.ARPA tartomány része, és nem az IN-ADDR.ARPA-é) a következő:

93. Természetesen ezek az adatok **nem** statikusak, időről-időre bizonyos részeik megváltoznak.
94. Az IPv6-os címek 16-os számrendszerben való ábrázolása nem a DNS-rendszerbeli trükk, hanem a szabványos írásmód (és így a napi gyakorlat része)

b.a.9.8.7.6.5.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.[95]

Példakonfiguráció

Caching DNS:

```
/etc/bind/named.conf:
    [...]

   forwarders {
   8.8.8.8;
   8.8.4.4 ;
   };

   [...]
```

A fenti konfigurációval a Bind-ot egy egyszerű továbbító funkcionalitással állítottuk be, azaz a kéréseket a Google névszerverei felé fogja Cache szerver üzemmódban továbbítani, ő maga saját zónákat nem fog tárolni és kiszolgálni. A fenti konfiguráció elmentése után a *service bind9 restart* után már tesztelhetünk is a *dig -x 127.0.0.1* vagy a *dig sajatdomain.hu* parancs kiadásával. (a Google ingyenes és gyors DNS szervereket üzemeltet, de nem szeretetből vagy kedvességből teszi ezt, éppen ezért csak akkor használjuk, ha a helyi szolgáltató szerverei nem megfelelőek, és nem is tudunk mást szervert elérni, azaz szükségmegoldásként időlegesen elfogadhatóak).

Elsődleges (mester) kiszolgáló

A /etc/bind/named.conf állományt szerkesszük, hogy hasonlóképpen nézzen ki:

```
  [...]

   zone "sajatdomain.hu" {
   type master;
   file "/etc/bind/db.sajatdomain.hu";
   };

  [...]
```

Majd az egyszerűség kedvéért a meglévő helyi adatbázist mintának használva készítsük el a saját domainünk adatbázis-állományát:

```
cp /etc/bind/db.local /etc/bind/db.sajatdomain.hu
```

Majd szerkesszük az elkészített másolatot, és a localhost értéket cseréljük ki az FQDN[96] értékünkre:

95. A példa a Wikipediának a fordított DNS-lekérdezésről szóló oldaláról van: http://en.wikipedia.org/wiki/Reverse_DNS_lookup#IPv6_reverse_resolution
96. http://hu.wikipedia.org/wiki/Fully_qualified_domain_name

```
; BIND data file for local loopback interface
  ;
  $TTL    604800
  @       IN      SOA     ns.sajatdomain.hu. root.sajatdomain.hu. (
   1         ; Serial
   604800         ; Refresh
   86400         ; Retry
   2419200         ; Expire
   604800 )       ; Negative Cache TTL
  ;
  @       IN      NS      ns.sajatdomain.hu.
  ns      IN      A       172.20.1.10

  ;also list other computers
  www     IN      A       172.20.1.50
```

Amint az látszik, elhelyeztük az első A rekordot a www.sajatdomain.hu részére, amely a 172.20.1.50 IP-című gépre fog mutatni. A *service bind9 restart* kiadása után pedig máris ki fogjuk szolgálni a www.sajatdomain.hu A rekordot. Ha már létrehoztuk az első A rekordot, akkor ne feledkezzünk meg a Reverse Zone létrehozásáról is, szerkesszük a /etc/bind/named.conf.local fájlt:

```
zone "1.20.172.in-addr.arpa" {
    type master;
    notify no;
    file "/etc/bind/db.172";
  };
```

Majd másoljuk át ugyanúgy a helyi adatbázis mintát:

```
cp /etc/bind/db.127 /etc/bind/db.172
```

Ezek után már csak el kell helyeznünk a web szerver reverse IP-jét a konfigurációban, tehát szerkesszük a /etc/bind/db.172 állományt:

```
; BIND reverse data file for local loopback interface
  ;
  $TTL    604800
  @       IN      SOA     ns.sajatdomain.hu. root.sajatdomain.hu. (
                                2         ; Serial
                           604800         ; Refresh
                            86400         ; Retry
                          2419200         ; Expire
                           604800 )       ; Negative Cache TTL
  ;
  @       IN      NS      ns.
  10      IN      PTR     ns.sajatdomain.hu.

  ; also list other computers
  50      IN      PTR    www.sajatdomain.hu.
```

Ezek után is szükséges a *service bind9 restart* kiadása.

Ha készen vagyunk már csak tesztelni kell, jól dolgoztunk-e:

nslookup www.sajatdomain.hu és *nslookup 172.20.1.50* illetve a DNS frissülése és elterjedése után már lekérdezhetjük a Google DNS-t is arról, tud-e rólunk:

```
dig www.sajatdomain.hu @8.8.8.8.8
```

Fontos, hogy a TTL értéket minden átállás előtt, azaz ha webszerverünk vagy az e-mail szerverünk új IP-re költözik (másik szerverterembe vagy gépre), akkor csökkentsük le akár már egy héttel a költözés előtt egy elfogadható kicsi értékre, hogy a root DNS szerverek is értesüljenek arról, hogy frissen kell tartaniuk a mi rekordunkat. Ez nagy segítség lesz majd a költözés után, amikor a DNS szerveren átállítjuk az új IP-re, akkor az elterjedés a lehető legrövidebb időn belül végbemenjen, legalább az érintett területen (pl. Magyarországon). A TTL aktuális értékét a konfigurációban tudjuk megnézni, vagy a dig parancs segítségével is ellenőrizhetjük:

```
dig +nocmd sajatdomain.hu +noall +answer
```

Bind Chroot-ban[97]

A Bind fejlesztését több évtizedes tapasztalattal bíró fejlesztők végzik, azonban mégis felépítéséből adódóan is folyamatosan a támadások célpontja. A leginkább akkor járunk jól, ha a Bind-ot lehetőségünk van csak lokális forwarder funkciójában használni, és kívülről az internet irányából a bind portunk nem látszik ki. Vagy ha csak 1-2 domain kiszolgálására van szükség, akkor pedig érdemes szétnézni a piacon, mivel ma már 1-2 ezer Ft / éves árszinten elérhető DNS kiszolgáló (egyben elsődleges és másodlagos is), amely a legtöbb esetben webes felületen kezelhető. Ha csak lehetőség nyílik erre, és tudunk megbízható DNS üzemeltetőt választani, akkor a külső esetben mindenképpen érdemes ezt választani. Ha viszont feltétlen nekünk kell üzemeltetnünk a DNS szervert, akkor egyrészt gondolkodjunk el a BIND chroot-olt környezetben futtatásán, illetőleg ha tudunk, akkor erre a feladatra dedikáljunk egy külön virtualizált gépet, és még azon belül is helyezzük Jail-be a Bind-ot. A tapasztalat azt mutatja, hogy érdemes óvatosan kezeli a Bind üzemeltetését.

ISC-DHCPD

Mivel a két alapfunkcióból a BIND csak az egyiket látja el, mellé szükséges egy önálló DHCP-szerver is. Javasolt erre a funkcióra a szintén az ISC által fejlesztett DHCP-szervert használni. A telepítés után (Ubuntu 12.04 LTS esetén: apt-get install isc-dhcp-server) az alap konfigurációs fájlban (/etc/dhcp/dhcpd.conf) kell beállítani a paramétereket. A következő paraméterek beállítása elindulásnak megfelelő:

```
ddns-update-style none;
option domain-name ”example.hu”;
option domain-name-servers 192.168.42.42, ns2.example.hu;
default-lease-time 36000;
max-lease-time 604800;
log-facility local7;
option subnet-mask 255.255.255.0;
subnet 192.168.1.0 netmask 255.255.255.0 {
```

97. https://help.ubuntu.com/community/BIND9ServerHowto#Chrooting_BIND9

```
  range 192.168.1.100 192.168.1.200;
  option routers 192.168.1.1;
}
host fixedclient {
  hardware ethernet aa:bb:cc:dd:ee:ff ;
  fixed-address 192.168.5.5;
}
```

A megszerkesztett konfiguráció tesztelésére a dhcpd -t parancs alkalmas, a szerver elindítása pedig a jól ismert service dhcpd start paranccsal történik. (És természetesen használhatóak a restart/stop/status paraméterek is.) A szerver futását ellenőrizhetjük a processzlistában (ps / top parancsok), illetve vizsgálhatjuk, hogy figyel-e a számára fenntartott hálózati porton (netstat -naput / lsof -i udp:bootps / ss -naput parancsok). A fenti konfigurációt használva a dhcpd a syslogon keresztül naplóz (az alapértelmezett beállítás szerint LOG_DAEMON, a mienk szerint LOG_LOCAL7 szolgáltatásként), így probléma esetén a syslog megfelelő naplóját kell vizsgálni.

Kliensek hálózati paramétereinek szolgáltatása: DHCP-szerver

Kliensek hálózati paramétereinek központi kiosztása, DNS lekérdezések gyorsítása

Már pár gépes környezetben is érdemes elgondolkodni, nagyobb kliensszám mellett pedig szinte kötelező, hogy a gépeink egy központi helyről kapjanak IP-címet, illetve egyéb hálózati adatokat, mint például az alapértelmezett átjáró címe és a DNS-szerverek címei. Ráadásul, ha bármi változás van a fenti adatokban, nem szükséges végigzarándokolnunk a gépeket, és egyesével elvégezni a módosítást, hanem csak egy helyen kell megtegyük azt: a DHCP-szerveren.

Attól, hogy a címek kiosztása dinamikusan történik, még nem jelenti azt, hogy nem határozhatjuk meg mely gépek milyen címet kapjanak, és akár gépenként eltérő hálózati paramétereket is megadhatunk (pl. más alapértelmezett átjáró, DNS), illetve megtehetjük azt, hogy amely eszközöket nem soroltunk fel, azok semmilyen hálózati adatot és IP-címet nem kaphatnak.

A fentieket a régóta ismert és használt ISC DHCP-szerver[98] segítségével fogjuk megvalósítani az alábbi példákon keresztül.

Gyakran egy hálózat gépein látszólag azért „lassú az internet”, mert a névfeloldás lassan történik a szolgáltató DNS-ein keresztül. Ezen csak egy jó DNS cache tud segíteni, olyan mint az alább bemutatásra kerülő DNSMasq[99]. Ennek segítségével a lekérdezéseink „közelebb kerülnek a hálózat gépeihez”, mivel amit egyszer már egy kliens kért, azt eltárolja, és legközelebb már rögtön a DNSMasq adja vissza az ügyfeleknek a kért címet, nem továbbítódik a kérés külső DNS-szerverek felé.

A DNSMasq használatával kiválthatjuk, hogy egy kisebb hálózatban minden gépen hosts fájl szerkesztésével vagy egy „komoly” DNS kiszolgáló beüzemelésével (pl. Bind) kelljen az esetleg szükséges pár host nevet a hálózat gépei számára feloldhatóvá tenni. A DNSMasq képes rá, hogy a kiszolgálón lévő /etc/hosts fájlban lévő név-ip párosok alapján a hozzá forduló kliensek számára ezeket az adatokat kiszolgálja, így csak egy helyen, a kiszolgálón kell egy egyszerű szöveges állományban felsorolnunk a hálózatunk azon eszközeit, amelyeket a kliensek számára elérhetővé szeretnénk tenni.

Az alábbi példák az aktuális Debian stabil rendszeren, Debian 7.2 Wheezy[100] -n készültek.

98. https://www.isc.org/downloads/dhcp/
99. http://en.wikipedia.org/wiki/Dnsmasq
100. https://wiki.debian.org/DebianWheezy

ISC DHCP-szerver telepítése, konfigurálása

Telepítés, alap beállítások:

Telepítsük a DHCP-szervert a következő paranccsal:

```
apt-get install isc-dhcp-server
```

A szerver konfigurációs állományát az /etc/dhcp/dhcpd.conf néven találjuk meg.

Nézzük az alapvető beállításokat:

```
ddns-update-style none;
```

Ez mondja meg, hogy kommunikálhat-e dinamikus frissítésre képes DNS-szerverrel. Az alapérték a none, hagyjuk mi is így.

```
option domain-name "lok";
```

A tartomány nevét adhatjuk meg, belső hálózat esetén alapvetően csak rajtunk múlik, mi ez.

```
Default-lease-time 86400;
max-lease-time 604800;
```

A kliensek számára mennyi az alapértelmezett címbérleti idő, illetve meddig használhatják azt megújítás nélkül. Számunkra nincs különösebb jelentősége, hagyhatjuk az alapértelmezetten.

Szükséges deklarálnunk még a DHCP-szerver számára, hogy melyek azok a hálózatok amelyeket kezel. Ha több interfész is van a gépben és nem szeretnénk, hogy valamelyiket kezelje, akkor is fel kell sorolni mindet, különben a syslogban panaszkodni fog deklarálatlan hálózatra. Azt, hogy valamely interfészen ne működjön, majd külön kell beállítanunk.

```
subnet 10.0.0.0 netmask 255.0.0.0 {
            range 10.0.2.100 10.0.2.120;
  }
  subnet 195.199.157.270 netmask 255.255.255.248 {
  }
```

Íme a hálózatok megadása. Mivel a példagépünkben két hálózati kártya van, így most itt mindkét kártyát deklaráltuk. *Amennyiben azt szeretnénk, hogy pl. az eth0 eszközön ne fogadja a kéréseket, csak az eth1-en, akkor az /etc/default/isc-dhcp-server fájlban kell az* ***INTERFACES="eth1"*** *sort beszúrni.*

Az első subnetben a két zárójel között szerepel egy range nevezetű IP-tartomány, ebből a tartományból fog IP-címeket osztani a hozzá forduló klienseknek (jelen esetben a 10.0.2.100-tól a 10.0.2.120-ig terjedő tartományból oszt címeket, mivel ez a belső hálózatunk, amely az eth1 eszközön érhető el).

```
option domain-name-servers 10.0.0.1,8.8.8.8;
  option subnet-mask 255.0.0.0;
  option broadcast-address 10.255.255.255;
  option routers 10.0.0.1;
```

A fenti sorokkal értjük el, hogy a kliensek megkapják a **DNS-szerverek** neveit, címeit (vesszővel elválasztva lehet felsorolást készíteni), az **alhálózati maszkot**, az **üzenetszórás** címét és az **alapértelmezett átjáró** címét.

Ezzel már van is egy működő DHCP-szerverünk, amely osztja a címeket a hozzá forduló klienseknek, és megadja számukra az alapvető hálózati adatokat. Vannak azonban még további beállítások.

Egyéb hasznos option definíciók

```
option netbios-name-servers 10.0.0.1;
  option smtp-server 10.0.0.1;
  option pop-server 10.0.0.1;
```

A gépek számára megadja a netbios névkiszolgáló (WINS szerver) címét – értelemszerűen Windowsos kliens környezetben van létjogosultsága, – és az SMTP és POP3 szerver címét is közzétehetjük (már amennyiben kliens oldalon tudják ezt értelmezni).

IP-címek kliensekhez rendelése

Gyakran előfordul, hogy azt szeretnénk, hogy a hálózatunk bizonyos gépei mindig pontosan ugyanazt az IP-címet kapják. Ezt a DHCP-szerveren tökéletesen meg lehet oldani, ahol összeköthetjük a host nevet, az eszköz MAC-címét és az IP-címet.

Ehhez tanácsos (lehet csoport nélkül is, de átláthatóbb) egy csoportot létrehoznunk, majd a csoport nyitó és záró zárójele között felsorolni a klienseket így: **group {}**. Nézzünk erre egy konkrét példát:

```
group {
  host muszashi-netbook {
      hardware ethernet bc:ae:c5:a8:d9:c1;
      fixed-address 10.0.0.11;
      }
  }
```

Először definiáltuk a csoportot, majd a nyitó zárójel után jött a host azaz a gép neve, majd a gép deklarálása is zárójellel kezdődött. Először megadtuk a *hardware ethernet* sorban a kliens MAC-címét, majd a *fixed-address* sorban a gépnek adott IP-címet. Ezután jött a gépdeklarációt bezáró zárójel, és végül a csoportdeklarációt lezáró zárójel. Természetesen a csoportba több gép is felvihető nem csak egy.

Globális opciók felülbírálása

Gépenként felülbírálhatjuk a globálisan megadott opciókat, ilyenkor a gép leírásánál kell megadni a kívánt opciókat.

```
group {
  host muszashi-netbook {
      hardware ethernet bc:ae:c5:a8:d9:c1;
      fixed-address 10.0.0.11;
     option routers 10.0.0.2;
      }
  }
```

Jelen példában a fenti gép már nem a mindenki számára megadott 10.0.0.1-et használja alapértelmezett átjárónak, hanem a 10.0.0.2-t.

Ebben a példában egyszerre él az ismeretlen kliensek (nem definiáltuk, milyen IP-címet kapjon) számára a cím automatikus osztása a megadott tartományból (range), és a csoportban (group) fel-

sorolt gépek számára a statikus IP adása. ***Ilyenkor nagyon figyeljünk arra, hogy statikus IP-t ne a kiosztható tartomány címeiből adjunk.***

Csak fix IP-címek

Amennyiben arra lenne szükségünk, hogy csakis a szerveren felsorolt gépek kapjanak IP-címet, akkor töröljük a range sort a hálózat deklarálásánál. Tehát akkor így kell kinéznie a hálózatunk megadásának:

```
subnet 10.0.0.0 netmask 255.0.0.0 {
  }
```

Innentől csak az kap címet, akiket a deklaráltunk a szerveren.

Amennyiben sok klienst szeretnénk felvinni, átláthatóbbá válik a konfigurációs állományunk, ha azokat nem itt soroljuk fel, hanem egy külön állományban, és itt csak megadjuk, hogy melyik fájlban találhatóak. Ez az alábbi módon tudjuk megtenni:

```
include "/etc/dhcp/kliensek";
```

Teljes példa konfigurációs állományok

/etc/dhcp/dhcpd.conf

```
ddns-update-style none;
  option domain-name "lok";
  Default-lease-time 86400;
  max-lease-time 604800;

  subnet 10.0.0.0 netmask 255.0.0.0 {
        range 10.0.2.100 10.0.2.120;
  }

  option domain-name-servers 10.0.0.1,8.8.8.8;
  option subnet-mask 255.0.0.0;
  option broadcast-address 10.255.255.255;
  option routers 10.0.0.1;
  option netbios-name-servers 10.0.0.1;
  option smtp-server 10.0.0.1;
  option pop-server 10.0.0.1;

  include "/etc/dhcp/kliensek";
```

/etc/dhcp/kliensek

```
group {

          host titkar-pc {
        hardware ethernet bc:ae:c5:8a:d9:c1;
        fixed-address 10.0.0.10;
          }

         host muszashi-netbook {
        hardware ethernet 08:00:27:b7:cd:e7;
        fixed-address 10.0.0.11;
```

```
        option routers 10.0.0.2;
          }

  }
```

DNSMasq telepítése, konfigurálása

Telepítés, alap beállítások

A DNSMasq-ot a következő paranccsal tudjuk telepíteni a gépünkre:

```
apt-get install dnsmasq
```

A telepítés során két konfigurációs állomány jön létre. Az egyik az /etc/default/dnsmasq, a másik az /etc/dnsmasq.conf néven jön létre.

Az /etc/default/dnsmasq konfigurációs állománnyal jelen esetben semmi tennivalónk nincs, hagyhatjuk az alapértelmezett beállításokon, itt két érték érdekes a számunka:

```
ENABLED=1
```

A fenti sor mondja meg, hogy elinduljon-e automatikusan a DNSMasq szolgáltatás. Ameddig az értéke 1, addig el fog indulni.

```
#IGNORE_RESOLVCONF=yes
```

Amíg ez a sor ki van kommentezve, addig veszi figyelembe a RESOLVCONF fájl tartalmát, egyébként sajátot kell megadni.

Mindkét beállítást érdemes így hagyni, hacsak nem direkt akarunk mást. Az összes további beállítást az /etc/dnsmasq.conf fájlon hajtjuk végre.

Konfiguráció

Az /etc/dnsmasq.conf fájlban is mindössze pár alapbeállítást kell módosítanunk a testreszabáshoz.

```
Port=53
  interface=eth1,lo
  except-interface=eth0
  listen-address=127.0.0.1,10.0.0.1
  bind-interfaces
```

A fenti beállításokat elvégezve elérjük, hogy mint DNS gyorsító kiszolgáló az 53-as porton válaszoljon a bejövő kérésekre, és csak a visszacsatoló eszközön (lo) és a második hálózati kártyán válaszoljon. Ez a beállítás akkor lehet fontos, ha több hálózati kártyánk is van a gépben, így el tudjuk érni, hogy csak a megadott hálózatok felé nyújtson szolgáltatást. Az except-interface pedig azt mondja meg, hogy mely csatolókon **ne** nyújtson szolgáltatást. Jelen esetben ez az eth0 eszköz, amely a külső csatolókártyánk. Látható ez pont ellentéte az interface-nek, és a kettő közül az egyik megadása elégséges is a megfelelő működéshez, de ha mindkét értéket helyesen adjuk meg, akkor biztos csak ott fog ténylegesen működni ahol ezt szeretnénk. (Egy kis paranoia sosem árt a rendszergazdának.) Tovább lehet szigorítani, ha megadjuk a listen-address alatt, hogy mely címeken válaszoljon: a csatolók meghatározásán túl ezzel még jobban körbe tudjuk határolni, mit is szeret-

nénk. A bind-interfaces opció lesz az, aminek köszönhetően a fenti interface és except-interface beállítások ténylegesen megtörténnek, ezért ezt ne hagyjuk el.

```
Domain-needed
  Bogus-priv
```

A fenti sorok hatására nem fog továbbítani a DNS-kiszolgálók felé olyan címeket, amelyek nem tartalmaznak tartománynevet (domain-needed) hiszen az a belső hálózathoz tartozik; és olyat sem, amelynek a címe nem routolható tartományba esik.

A fenti beállításokkal már van egy működő DNS-cache a gépünkön, amely csak a belső hálózat kéréseire válaszol.

További konfigurációs ötletek

```
cache-size=600
```

Az alapértelmezett cache méretét nyugodtan növelhetjük, ha hálózatunk forgalma ezt igényli.

```
domain=lok.local
  expand-hosts
```

Amennyiben az /etc/hosts fájlban vettünk fel gépnév–IP-cím párosokat, akkor azokat a DNS-cache kliensei felé a DNSMasq automatikusan szolgáltatja az ott megadott néven. Így gyakorlatilag egy kis hálózat esetén kiválthatjuk a „rendes” DNS-szervert. A fenti két beállításra csak akkor van szükségünk, ha azt szeretnénk, hogy minden általa ismert belső hálózathoz tartozó gépnévhez hozzáillesszen egy tartományvégződést. Ezt adjuk meg az expand-hosts bekapcsolásával. A „domain” sorban pedig azt a nevet adjuk meg, amit szeretnénk, hogy tartományi névként hozzáillesszen. Ha ezek a sorok nem szerepelnek, akkor csak simán az /etc/hosts fájlban felsorolt néven oldja fel a meghatározott gépek neveit.

DNSMasq mint DHCP-szerver

A DNSMasq képes DHCP-kiszolgálóként is működni, azonban DHCP-szerverként már az ISC-DHCP-t tárgyaltuk, így ezt a funkciót nem részletezzük. Amennyiben azt szeretnénk, hogy a DNSMasq DHCP-szerverként működjön, úgy az /etc/dnsmasq.conf fájlban vegyük ki a megjegyzésjelet a **dhcp-range** sor elől, és írjuk oda a kívánt IP-tartományt és a címbérleti időt. Az alábbi példában a 10.0.0.100-tól a 10.0.0..150-ig oszt IP-címeket a hozzá forduló ügyfeleknek

```
dhcp-range=10.0.0.100,10.0.0.150,255.0.0.0,12h
```

Teljes példa konfigurációs állományok:

/etc/resolv.conf

```
nameserver 8.8.4.4
  nameserver 8.8.8.8
```

Itt semmit nem módosítottunk, csak arra kell ügyelni, hogy létező és működő DNS-szerverek címei legyenek beírva.

/etc/default/dnsmasq

```
#DOMAIN_SUFFIX=`dnsdomainname`
  #DNSMASQ_OPTS="--conf-file=/etc/dnsmasq.alt"
```

```
  ENABLED=1
  CONFIG_DIR=/etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg-new
  #IGNORE_RESOLVCONF=yes
```

Itt sem végeztünk módosítást, mindössze ellenőrizzük, hogy elindul-e a szolgáltatás, és használja-e a resolvconf-ot.

/etc/dnsmasq.conf

```
port=53
  interface=eth1,lo
  except-interface=eth0
  listen-address=127.0.0.1,10.0.0.1
  bind-interfaces

  domain-needed
  bogus-priv

  cache-size=600

  domain=lok.local
  expand-hosts
```

Itt láthatóak az elvégzett módosítások.

/etc/hosts

```
127.0.0.1     localhost
  127.0.1.1   szerver4.lok.local      szerver4
  # The following lines are desirable for IPv6 capable hosts
  ::1     localhost ip6-localhost ip6-loopback
  ff02::1 ip6-allnodes
  ff02::2 ip6-allrouters

  # Felsorolt nevek
  10.0.0.21   nyomtato1
  10.0.0.22   nyomtato2
  10.0.0.1    intranet
```

A kezdeti sorokon túl, amelyek telepítéskor kerültek ide, a f*elsorolt nevek* után írtuk azokat a név–IP párosokat, melyekről szeretnénk, hogy a helyi hálózatunk számára DNS-szerverként névfeloldást végezzen a DNSMasq. A nyomtato1, nyomtato2, intranet neveket ebben a formában, és az intranet.lok.local, nyomtato1.lok.local néven is feloldja jelen konfigurációban.

Webszerver feladata, telepítése és üzemeltetése

Egy átlagos webes kiszolgáló a legtöbb esetben egy úgynevezett LAMP környezetből áll, amelynek részei az operációs rendszer, a webszerver, valamilyen szerveroldali szerver szkriptnyelv, mint például a PHP és az adatbázis-kezelő. Egész pontosan így kapjuk meg a LAMP rövidítés betűit. A legtöbb statisztikai mérés szerint a Linux+Apache+MySQL+PHP környezet a leginkább elterjedt (lásd az adatbázis fejezetet). Egy átlagosan feltelepített környezetben tehát a fenti megoldás segítségével kialakíthatunk dinamikus és statikus, szabad szoftverekkel működtetett oldalakat. Ez a fejezet segít a webszerverek és a hozzájuk szorosan kapcsolódó szolgáltatások kiválasztásában. Ilyen például az FTP szerver vagy az SFTP szerver, hiszen a legtöbb esetben az oldalak karbantartására ma is ezt a megoldást használják azok karbantartói, természetesen a számos egyéb lehetőség mellett. Ez a fejezet a webes kiszolgálás köré csoportosítható eszközöket ismerteti.

Webszerverek

Apache

Az Apache hosszú évek óta a legnépszerűbb webszerver a független kutatások szerint. Az Apache projektet az ASF alapítvány keltette életre, és többek között a webszervert is ők fejlesztik, tartják karban. Az alapítvány védőernyője alatt tevékenykedik számtalan szabad szoftveres programozó, akik az 1999-es indulás óta fejlesztik a webszervert. A kezdetekben a cél az volt, hogy az NCSA HTTPd bővítésén dolgozzanak, majd 1999. június 1-jén hivatalosan is létrejött az ASF. Az Apache egy igen kifinomult, moduláris és rugalmasan bővíthető webszerver, amely kompatibilis a HTTP/1.1 (RFC 2616) protokollal. A projekt nyilvánosságra bocsátása után dinamikusan fejlődött, és hamar valós alternatívát kínált az akkor előnyös pozícióban lévő Netscape Communications Corporation webszerverrel szemben. Azóta is folyamatosan fejlődik, és gyakorlatilag uralja a piacot. Moduláris felépítésének köszönhetően mi magunk dönthetünk arról, hogy az alap telepítés után milyen funkciókat akarunk használni, például az SSL vagy a mode_rewrite modult. Megfelelő hardvertámogatás mellett jól skálázható, kis- és nagyvállalati megoldásokra is. A legtöbb szkriptnyelvet támogatja, mint a PHP, Perl, Python, Ruby on Rails. Több ismert cég is felhasználja az Apache-ot termékeiben: Oracle, IBM WebSphere, OSX beépített webszerverként. Számos router, gateway, NAS megoldás szerves része. Még zárt szoftverként is forgalmazható, mert ezt nem akadályozza meg a nem copyleft Apache szabad szoftver licenc. Telepítése Ubuntu LTS alatt is igen egyszerű, az:

```
apt-get install apache2
```

parancs kiadásával az alap függőségek kezelése mellett fog települni a webszerver. Gyakorlatilag a parancs lefuttatása után van egy teljesen alap szinten működő web szerverünk, amely létrehozza a /var/www könyvtárban az alap „It works!” oldalt, és innentől kezdve a webszerver alap beállítások szerint a 80-as porton fogadja a kéréseket. Mint oly sok hálózati kiszolgáló szolgáltatást, az Apache-ot sem érdemes alapbeállításokkal használni, hiszen egyrészt nem túl biztonságos, valamint jellemzően több tartományt vagy oldalt szolgál ki az Apache, így mindenféleképpen nézzük

meg, milyen beállításokat érdemes átállítanunk. A konfigurációs fájlok az /etc/apache2/ könyvtárban találhatóak.

```
apache2.conf httpd.conf   mods-enabled/     sites-enabled/ conf.d/        magic
  ports.conf        ssl/ envvars       mods-available/  sites-available/
```

Az apache2.conf fájl

Az ismerkedést kezdjük az apache2.conf fájllal, amelyben a legfontosabb központi paraméterek megtalálhatóak, itt paraméterezhetünk az Apache alapvető viselkedésével kapcsolatban sok mindent, valamint igazából ide van belinkelve a többi konfigurációs fájl is, azaz az Apache innen olvassa be például a mods/site-enabled konfigokat is. Nézzük, mihez érdemes hozzányúlni és miért:

```
Timeout 300
```

Másodpercben kifejezve megmondjuk, hogy az Apache mennyi időt várjon mielőtt Timeout hibával lezárja a kapcsolatot. Főként nagy forgalmú, vagy I/O terhelt szerverek esetében lehet ez érdekes. Alapbeállításon célszerű hagyni, illetve ha szükséges változtatni (általában lefele irányban, azaz némileg csökkenteni) akkor minden esetben tesztelni érdemes, hogy az adott terheltség és az adott kiszolgálási stílus mellett számunkra mi a megfelelő: nem mindegy, hogy egy nagy terhelésű fórum szervert paraméterezünk, vagy egy 200 MB-os letöltőközpontot.

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

Ezen 3 paraméter segítségével állítható be, hogy a szerver egy TCP kapcsolat nyitva tartása mellett több kérést is kiszolgál. Ilyen eset, amikor egy oldalon sok kép van, és az Apache ilyen beállítások mellett csak egy TCP kapcsolatot használ fel arra, hogy esetleg az összes adatot a kliens felé továbbítsa. Az értékeket ezredmásodpercben tudjuk megadni. Megint csak: nagy terhelés esetén az oldal felépítése és a felhasználók szokásai alapján érdemes próbálkozni a ki/be kapcsolásával, és Munin segítségével folyamatosan monitorozni a végeredményt.

```
<IfModule mpm_prefork_module>
      StartServers          5
      MinSpareServers       5
      MaxSpareServers      10
      MaxClients          150
      MaxRequestsPerChild   0
  </IfModule>
```

Alap telepítés szerint az mpm_prefork_module-t használjuk, hiszen a libapache2-mod-php5 kiterjesztés ezzel kompatibilis. Amit érdemes tudni róla, hogy ebben az esetben a kéréseket egy szál szolgálja ki, és a memóriaigénye is magasabb a workerhez képest. Beállítása és üzemben tartása átlagos esetben egyszerűbb, és javasolt is a prefork+libapache2-mod-php esetén. Érdemes tudni róla, hogy ha mégis a workert akarnánk választani, akkor lehetőségünk lenne a mod_fcgi-vel PHP-futtatásra. Nézzük a paramétereket:

StartServers: meghatározza, hogy az Apache indulásakor hány szerverfolyamat induljon el. Általában érdemes az alapértelmezett beállításon hagyni, azaz 5-ön.

MinSpareServers, MaxSpareServers: a minimum és maximum SpareServer opciókkal szabályozhatjuk, hogy hány tartalék kiszolgáló fusson a háttérben, amelyek a várakoznak az új csatlakozók kiszolgálására. Így ha a szabad folyamatok száma eléri a MaxSpareServers értéket, akkor az Apache le fogja kapcsolni a feleslegesen futó folyamatokat.

MaxClients: meghatározhatjuk, hogy mennyi klienst szolgáljon ki az Apache egy időben maximum. Tipikusan ha egy kliensnek sokat kell várakozni az oldal betöltéséig, akkor vélhetőleg ez a szám túl alacsony.

MaxRequestsPerChild: ezzel szabályozzuk, hogy egy gyermek folyamat összesen mennyi kérést szolgálhat ki. Alapértéke nulla, de néha – főleg memóriaszivárgás esetén – érdemes ezt az értéket terheléstől függően 5000-10 000 közé állítani, de csak teszt jelleggel.

```
AccessFileName .htaccess
<Files ~ "^\.ht">
      Order allow,deny
      Deny from all
      Satisfy all
  </Files>
```

Rögzítjük, hogy a .htaccess fájlt hogyan ismerje fel a szerver, valamint megmondjuk, hogy a .htaccess és a .htpasswd fájlok kliensek által való lekérhetőségét akadályozzuk meg. A .htpasswd fájlokat amúgy sem praktikus publikus könyvtárba elhelyezni, de erről később még bővebben értekezünk.

```
HostnameLookups Off
```

A direktíva segítségével megmondhatjuk, hogy a kliensek IP-címének rögzítését hogyan akarjuk látni a naplóban. Paraméterei: On|Off|Double. ON esetén az IP-hez kapcsolódó reverse érték is rögzítve lesz. Érdemes azonban figyelembe venni, hogy akár már egy kis látogatottságú weboldalon is rengeteg nslookup-ot eredményezhet, azaz nem véletlen, hogy az alapértelmezett paramétere az OFF.

```
ErrorLog ${APACHE_LOG_DIR}/error.log
  LogLevel warn
  LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
  Agent}i\"" vhost_combined
  LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\""
  combined
  LogFormat "%h %l %u %t \"%r\" %>s %O" common
  LogFormat "%{Referer}i -> %U" referer
  LogFormat "%{User-agent}i" agent
```

Itt határozzuk meg a naplózással kapcsolatos formai és fájl igényeinket. Azaz, hol legyen a központi error.log. Ebbe a fájlba fogja rögzíteni az Apache a működése során fellépő hibákat, ide fognak kerülni (ha másként nem rendelkezünk) a PHP-ból eredő hibák, illetve a nem létező fájlok és egyéb természetű hibák. Ezért célszerű oldalanként definiálni egy error.log fájlt. A többi paraméter segítségével pedig a látogatói kliensekről megszerezhető információk rögzítési formátumát állítjuk be. Mivel alapesetben a felhasználók gépei és böngészői túlságosan közlékenyek (User Agent), ezért ha nincs szükségünk minden információra, itt tudjuk szűkíteni a kívánt információhalmazt.

```
Include ports.conf
```

Itt történik a fejezet elején felsorolt konfigurációs fájlok meghívása, azaz gyakorlatilag ezek is az apache2.conf részét képezik, csak egy strukturált módon.

A ports.conf fájl

A következő konfigurációs fájl tehát a ports.conf, amelyben azt rögzítjük, hogy a webszerver milyen IP-címeken és milyen portokon hallgasson a világ felé:

```
NameVirtualHost *:80
NameVirtualHost *:443
```

Jellemzően egy webszerver nem csak egy tartományt szolgál ki, hanem többet. Ebben az esetben be kell kapcsolnunk a NameVirtualHost direktívát, amely azonban a 2.3.11-es verzió óta szükségtelenné vált, mivel már az Apache úgy értelmezi, hogy bármelyik IP vagy port több tartományt szolgál ki. Itt jelenleg azért szerepel, mivel a jelenleg használatos Ubuntu LTS verzióban még kötelező direktíva.

```
Listen 80
<IfModule mod_ssl.c>
  Listen 443
</IfModule>
```

Ezzel megmondjuk a webszervernek, hogy a standard 80-as és 443-as portokon működjön. A 80-as port természetesen az alap, a 443-ashoz viszont SSL tanúsítványt kell készíteni és be is kell konfigurálni az Apache számára, amelyet később teszünk meg, ezért van feltételek között. Azaz ha az SSL modult bekapcsoltuk, akkor a Listen paraméter is élni fog, ha nem akkor természetesen nem. Megadható még így is:

```
Listen 192.168.1.1:80
```

A security fájl

A következő fájl, amelyet az Apache az /etc/apache2/conf.d-ből olvas fel a security. Nevéből adódóan a biztonsági beállítások egy jó részét tartalmazza:

```
ServerTokens Prod (Full | OS | Minimal | Minor | Major | Prod)
```

Meghatározza, hogy a webszerver milyen fejlécet adjon vissza a kliensek felé. Értelemszerűen itt is igaz az, hogy ha nem tesztelés jelleggel szükséges a teljes információ visszaadása, akkor a lehető legkevesebbet kell magunkról elárulnunk, azaz érdemes Prod-ra állítani az értéket, amely így csak a Product értéket fogja elárulni, azaz Apache lesz.

Nézzük meg mi történik, ha Full értéken van:

```
telnet localhost 80
  Trying 127.0.0.1...
  Connected to localhost.
  Escape character is '^]'.
  HEAD // HTTP/1.0 \n\n

  HTTP/1.1 401 Authorization Required
  Date: Mon, 13 May 2013 11:01:07 GMT
  Server: Apache/2.2.22 (Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1
  WWW-Authenticate: Basic realm="ByPassword"
  Vary: Accept-Encoding
  Connection: close
  Content-Type: text/html; charset=iso-8859-1

  Connection closed by foreign host.
  ServerSignature Off
```

Bekapcsolva (jellemzően hiba esetén van így) verzió- és egyén-szenzitív információt (signature) szolgáltat, természetesen nem hibakereső üzemmódban ezt is érdemes kikapcsolva tartani, főként biztonsági megfontolásból.

```
TraceEnable Off
```

Főként teszt és diagnosztikai esetekben érdemes bekapcsolva tartani.

A charset fájl

Ugyancsak a conf.d/ alatt található a charset fájl, amely ma már igen egyszerű, hiszen jellemzően UTF–8 kódolás támogatásával dolgozunk:

```
AddDefaultCharset UTF-8
```

Itt azonban szükség esetén megadható egyéb kódolás külön támogatása is.

Az Apache kiterjesztése

A következő fontosabb lépés az Apache kiterjesztési lehetőségeinek megismerése, azaz a hozzá kapcsolható modulszerkezet, illetve azok ki-be kapcsolása. Ha például PHP-t telepítünk a rendszerre – mint ahogy azt a fejezet vége felé a PHP részletezésénél fogunk is – az Ubuntu telepítőrendszere, miután feltelepítette a szükséges függőségeket, köztük az apache2 modult is, az /etc/apache2/mods-available alá fogja elhelyezni a PHP modulfájl helyét és a betöltéshez szükséges konfigurációt tartalmazó fájlt, amely így fog kinézni:

```
cat /etc/apache2/mods-available/php5.load
   LoadModule php5_module /usr/lib/apache2/modules/libphp5.so
cat /etc/apache2/mods-available/php5.conf
   <IfModule mod_php5.c>
      <FilesMatch "\.ph(p3?|tml)$">
          SetHandler application/x-httpd-php
      </FilesMatch>
      <FilesMatch "\.phps$">
          SetHandler application/x-httpd-php-source
      </FilesMatch>
      # To re-enable php in user directories comment the following lines
      # (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
      # prevents .htaccess files from disabling it.
      <IfModule mod_userdir.c>
          <Directory /home/*/public_html>
              php_admin_value engine Off
          </Directory>
      </IfModule>
  </IfModule>
```

A PHP-vel nincs teendőnk, hiszen azt majd a telepítőrendszer elintézi, azonban az /etc/apache2/mods-available könyvtárban van sok olyan modul, amely telepítve van, működésre kész, de jelenleg nem használja a webszerver. Az egyik ilyen például az SSL, amelyet egyszerűen az Apache számára használhatóvá tehetünk, csak létre kell hoznunk egy szimbolikus linket az /etc/apache2/mods-enabled/ssl.load -> /etc/apache2/mods-available/ssl.load között, azaz a mods-enabled alá linkeljük a modes-available/ alatt lévő ssl.log és ssl.conf fájlokat, majd az apache2ctl configtest és az apache2ctl restart parancsok segítségével érvényesítjük. A ports.conf-ban rögzítettek szerint így már fel fogja ismerni az Apache, hogy az SSL modul be van töltve. Az SSL további konfigurációját pedig a normál Virtualhost konfigurálás után részletezzük.

Egy tartomány kiszolgálásának beállítása

Nézzük, hogyan tudunk bekonfigurálni egy www.iskola.hu és iskola.hu tartományt. Első lépésben szükséges, hogy a DNS szerverünkben beállítsuk (vagy kérjük annak karbantartójától, hogy a www.iskola.hu a szerverünk IP-címére mutasson – „A” rekord), valamint célszerű az Origin (@)-t, azaz az iskola.hu (www nélküli) címet ugyanerre az IP-re irányítani. Ha ezzel megvagyunk, akkor

az /etc/apache2/sites-available könyvtárba helyezzük el a soron következő virtualhost leírófájlját. Ha ez lesz az első, akkor célszerű (a nullás foglalt, az az alapértelmezett) a 001-iskola.hu nevet adni neki. Az egyszerűség kedvéért, mivel itt egyszerre értelmezhető a konfiguráció, minden sor után #-tel kezdve olvasható a magyarázat, értelemszerűen ezeket nem kell a végleges fájlba másolni:

```
<VirtualHost *:80>
   #Itt határozzuk meg, hogy milyen IP-címekre és portokra vonatkozzon ez a
   virtualhost. Mi most a csillag karaktert használtuk, hogy ne okozzon gondot,
   ha a gépnek több IP-címe van, vagy le kell cserélni. Megadhatunk IP-címet is,
   például: <VirtualHost 192.168.1.1:80>

ServerAdmin webmaster@iskola.hu
   #Meghatározzuk a virtualhost által hirdetett rendszergazdai címet
ServerName www.iskola.hu
ServerAlias iskola.hu
   #A ServerName direktíva mondja meg, hogy milyen DNS névre hallgat majd, a
   ServerAlias-hoz pedig felvehetjük a szükséges többit, jelen esetben a www
   nélküli címet, de lehetne itt a mail.iskola.hu is.
DocumentRoot /var/www/iskola/
   #A DocumentRoot direktívával meghatározzuk, hogy honnan kezdődik a
   virtualhost nyilvános könyvtárszerkezete. Ide rakjuk az index.html, index.php
   stb. fájljainkat.

<Directory />
Options FollowSymLinks Indexes
AllowOverride AuthConfig All
</Directory>
#Az adott könyvtárra meghatározzuk az Options segítségével, hogy ha nincs
   index.html vagy index.php fájl az adott könyvtárban, akkor kilistázza-e a
   könyvtár fájljait (Indexes opció), illetve beállítjuk, hogy kövesse a
   szimbolikus linkeket. Majd az AllowOverride opcióval lehetővé tesszük, hogy
   ha .htaccess fájl található egy könyvtárban, akkor az abban lévő
   paramétereket figyelembe vegye, mintha csak például az Options-ök között
   szerepelt volna. Ez a paraméter barátunk és ellenségünk is lehet. Fontos,
   hogy akkor engedjük All opciókkal, ha erre a PHP programozóknak egyedileg
   szükségük van és megbízunk bennük, minden más esetben érdemes
   virtualhostonként és könyvtáranként egyénileg szabályozni, engedni vagy
   tiltani bizonyos opciókat. Az AuthConfig opcióval pedig a .htaccess-ből
   megvalósított felhasználó/jelszó bekérését engedélyezzük.
<Directory /var/www/>
   Options FollowSymLinks MultiViews
   AllowOverride Authconfig
    Order allow,deny
   allow from all
</Directory>
#Mint látható a /var/www könyvtárra, amely a DocumentRoot alatt helyezkedik el,
   részben eltérő hozzáférési beállításokat eszközöltünk. Tiltottuk az Indexes
   beállítást, azaz nem engedjük a listázást, illetve hiányzik az All paraméter,
   azaz ott már nem dönthet a .htaccess-ben utólag beállított felülíró
   paraméter. Az egészre akkor lehet szükség például, ha a weboldalunknak
   szüksége lehet a DocumentRoot-on kívülről írni vagy olvasni, ilyen eset
   amikor 2 oldal egy szerveren részben közös PHP kódot használ, vagy csak
   biztonsági okokból elkülönítünk bizonyos dolgokat a DocumentRoot-tól
   szeparálva.

   ErrorLog ${APACHE_LOG_DIR}/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/iskola.hu-access.log combined
```

```
#A korábban már tárgyalt naplózási lehetőségeket rögzítjük, azaz hol legyen a
   site error.log fájlja, milyen részletesen legyen a napló, illetve a sima
   hozzáférési napló hol tárolódjon.

   </VirtualHost>
   #itt ér véget a virtualhostunk.
```

Ha elmentettük, akkor következő lépésként be kell linkelnünk most létrehozott fájlt a sites-enabled könyvtárba, hogy az Apache tudja, most már használnia kell:

```
ln -s /etc/apache2/sites-available/001-iskola.hu /etc/apache2/sites-enabled/001-iskola.hu
```

Majd futtassuk le a tesztet, hogy nem írtunk-e el valamit, vagy véletlenül nem hagytunk ki esetleg egy lezáró </Virtualhost> címkét:

```
sudo apache2ctl configtest
```

Ha a válasz a következő:

```
Syntax OK
```

Akkor minden rendben van, ebben az esetben kiadhatjuk a sudo apache2ctl restart parancsot. Ha az is hiba nélkül futott le, akkor az iskola.hu és a www.iskola.hu tartományokat is ki fogja szolgálni az így beállított Apache.

Egy biztonságos tartomány beállítása

Nézzük, mi a teendő, ha HTTPS alatt szeretnénk az ugyancsak a példában szereplő www.iskola.hu és iskola.hu oldalakat konfigurálni, értelemszerűen itt most csak az előző normál 80-as porton működő Apache-hoz viszonyított különbséget fogjuk taglalni. Azt tehát már beállítottuk korábban, hogy a mods-enabled könyvtárba be legyen linkelve az SSL config és load fájl is, valamint az Apache port szinten is tudja, hogy ha a modul be van töltve, akkor a 443-as porton szükséges figyelnie, azért másoljuk az iskola.hu konfigurációs fájlját egy új fájlba, amelyet hívhatunk /etc/apache2/sites-available/002-iskola.huSSL-nek. A következő változtatásokat eszközöljük:

```
A Virtualhost direktíva értelemszerűen ne 80-as hanem 443-as legyen:
   <VirtualHost *:443>
```

A szerver név és alias meghatározás után, de még a DocumentRoot előtt a következő sorokat helyezzük el:

```
SSLEngine on
   SSLCertificateFile /etc/apache2/ssl/iskoal.hu.crt
   SSLCertificateKeyFile /etc/apache2/ssl/iskola.hu.private.key
```

Értelemszerűen előtte az SSL fejezetben (a fejezet jelenleg készítés alatt van, amint elkészülünk a weboldalunkon megtalálható lesz) leírtak szerint készítsük el a kulcspárokat. Ezek után ne felejtsük belinkelni a sima iskola.hu-hoz hasonlóan az iskola.huSSL fájlt ugyanúgy a sites-enabled könyvtárba, majd az

```
apache2ctl configtest
```

parancs futtatásával győződjünk meg róla, hogy semmit nem írtunk el, és csak ezek után indítsuk újra az Apache-ot, amelyet ezúttal az SSL konfiguráció bővülése miatt egy teljes értékű restart kiadásával érdemes megejteni:

```
service apache2 restart
```

Ha mindent jól csináltunk, és az error.log fájlban sincs jelezve semmi hiba, akkor a 443-as porton a https://www.iskola.hu oldalon meg fog jelenni pontosan ugyanaz a tartalom, mint a sima is-

kola.hu oldalon. Mivel a HTTP és a HTTPS virtualhost konfigurációja nem egy fájlba lett belerakva, ezért figyeljünk oda arra, hogy később amikor változtatni kell a HTTP konfiguráción, akár egy PHP-beállítást, akár egy log vagy Directory bejegyzést, akkor ezt a HTTPS fájlba is vezessük át.

Lighttpd

A Lighttpd egy kicsi és rendkívül gyors alternatív webszerver motor. A fejlesztését 2003-ban teljesen az alap koncepció lefektetésével kezdték, a C10K probléma köré építve. Fő jellemzője, hogy kifejezetten biztonságos és rendkívül gyors, az erőforrásokat teljes mértékben kíméli, így például egy statikus tartalom kiszolgálásához jóval kevesebb RAM- és processzorkapacitást igényel. A mostanában igen elterjedt bérelhető VPS megoldásokban gyakran használják, a szűkös erőforrások optimális kihasználása érdekében. Támogatja a FastCGI, CGI, Auth, Output-Compression, URL-Rewriting, SSL opciókat és még számos más megoldást is. Sok esetben használják kirakat webszervernek, azaz felépítenek egy LAMP környezetet, amely csak a dinamikus tartalmat szolgálja és a statikus tartalmat egyfajta terheléscsökkentő pajzsként ugyanezen a gépen a Lighttpd veszi át. De ugyanezt a megoldást használják fordítva is, vagy csak képek kiszolgálására. Jelenleg olyan oldalak használják kiszolgálásra mint a YouTube, Wikipedia vagy a meebo.

Nginx

A Nginx elveiben nagyon hasonló a Lighttpd-hez. 2004-ben Igor Sziszojev, egy orosz rendszer- és szoftvermérnök publikálta az első verziót belőle. Elképesztő terhelhetőség jellemzi, pontosan ugyanazon szempontok mentén, mint a Lighttpd-t. Felépítése moduláris, és elképesztően sok funkciót támogat. Felhasználása nagyon hasonló a Lighttpd-éhez, főként cache proxy és terhelés-elosztó funkciókat valósítanak meg vele. Gyakori felhasználás, hogy a statikus képeket kiszervezik egy külön Nginx szerver alá, ezzel sokszorozva meg a hardver lehetőségeit. A szakmai fórumok tele vannak ajánlásokkal és gyakorlati tapasztalatokkal, hogy milyen feladatra melyik szervert érdemes és lehet használni. Mindenképpen érdemes átolvasni ezeket egy bonyolultabb rendszer esetén, mert nem egyértelmű, hogy hosszú távon melyik a jobb. Jelenleg a Nginx a világ második legtöbbet futtatott webszervere (a saját dokumentációja szerint).

Webszerver üzemeltetéséhez szükséges komponensek

PHP

A PHP (Personal Home Page Tools) egy szerver oldali szkriptnyelv, amely egyike az első olyan szkriptnyelveknek, amelyet HTML kódba lehet ágyazni külső fájl használata helyett. A webszerverhez kapcsolódó modulként egy PHP-feldolgozó értelmezi a kódot. Az évek során akkora felhasználói és fejlesztői bázisra talált, hogy Rasmus Lerdorf az alkotója a The PHP Group-ra bízta a PHP-t. Mára a legelterjedtebb szkriptnyelv lett, amely messze a legtöbb lehetőséget támogatja modulrendszerével, mint például adatbázis-kezelés számtalan változata, képmanipuláció, cache lehetőségek, távoli szerver kapcsolatok stb. A kezdetekben csak egy szkript gyűjtemény volt, de mára iparági standard lett, amelyet számtalan területen használnak fel. A kezdeti „Personal Home Page Tools” megnevezést megváltoztatták, és ma már a PHP: Hypertext Preprocessor elnevezést használják. Lehetőség van nem csak webszerveren keresztüli használatára, hanem kiszolgáló olda-

li parancssori munkára is, ezzel nagyrészt átvéve az előtte egyeduralkodó shell és Perl szkriptes megoldásokat, hiszen ugyan úgy crontab-ból is ütemezhető a PHP. A PHP biztonsága híresen kétes, pedig főként nem a PHP-t fejlesztők által elkövetett tervezési hibák jellemzőek, hanem inkább a PHP-t használó fejlesztők nem tartják be a megfelelő direktívákat, és alapvető biztonsági szabályokat (bevitelkezelés stb.). Sokáig a PHP beépített lehetősége volt a safe_mode és a köré csoportosuló egyéb függvények. Sajnos vagy szerencsére az 5.3-as verzió kiadásával ezek kivezetése fokozatosan megkezdődött, így az üzemeltetőknek és a programozóknak kell más megoldások után nézni. A safe_mode megoldás egyébként egy igen kötött programozói viselkedést követelt meg, viszont nagyobb látszólagos biztonságot nyújtott, mint amilyen az a valóságban volt.

A PHP telepítése

Telepítése igen egyszerű, ugyanakkor sokan nem fordítanak elég figyelmet a telepítés utáni részletes beállítására, az Apache-csal való összehangolásra. Ugyanis alapbeállításokkal a PHP igen megengedő üzemmódban fut, amelyen érdemes szigorítani. Az alapszintű telepítése tehát Ubuntu LTS rendszeren:

```
apt-get install libapache2-mod-php5 php5
```

A php5 ezzel automatikusan települ, és a telepítőrendszer be is linkeli a PHP moduljait az Apache megfelelő könyvtárába, majd újra is indítja az Apache-ot, azaz a telepítés után a PHP rendszer működőképes. Ekkor kell elkezdenünk a finomhangolást:

A PHP Apache-hoz kapcsolódó beállításait az /etc/php5/apache2/php.ini fájlban találhatjuk, a fontosabb beállítások, amelyeket érdemes módosítani a telepítés után:

```
engine = On
```

Ezzel ki-be kapcsolhatjuk globális szinten a PHP-t. Ezt megtehetjük virtualhostonként is az Apache-ból, úgy hogy abba a virtualhostba, amelyben le szeretnénk tiltani a PHP-t, beírjuk a következőt: php_flag engine off

Ennek főként akkor van jelentősége, ha egy webhelyet átmenetileg karbantartó üzemmódba teszünk, például egy nagyobb átállás időtartamára azt akarjuk, hogy a látogatók egy index.html-ből tudomást szerezzenek róla, hogy az átállás meddig fog tartani, de ne érjék el a könyvjelzőzött direkt linkeken a PHP tartalmat.

```
safe_mode =Off
```

A Safe Mode egy remek próbálkozás arra, hogy rákényszerítse a programozókat a játékszabályok betartására. Sajnálatosan azonban a PHP 5.2-es változatától már nem támogatott, de még részben használható. Természetesen a Safe Mode nem kínált jó megoldást a biztonsági hibás kódokra és részben olyan érzetet kellett az üzemeltetőkben, hogy a PHP-n keresztül az esetleges támadók nem tudnak majd kitörni a DocumentRoot-ból. Azaz sok esetben illúzió volt, de sok esetben egy fajta alternatíva volt arra az esetre, ha egy webszerveren eltérő tulajdonosú, vagy fejlesztőjű oldalak futottak. Ilyenkor egy bizonyos szeparációs biztonságot jelentett az üzemeltetőnek és igen nagy kötöttséget a fejlesztőknek. Mivel használata a továbbiakban nem tanácsos, itt eltekintünk a részletezésétől.

```
disable_functions =
    pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,p
    cntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_err
    or,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,
    pcntl_setpriority
```

A disable_functions egy remek változó arra az eshetőségre, ha szeretnénk tiltani számos programozók által kedvelt, de általunk nem szükségesnek vélt (vagy éppen biztonsági megfontolásból) funkciót. Erre remek példa lehet a phpinfo függvény, amely sok esetben a fejlesztéshez elengedhetetlen, de véletlen hanyagságból kint felejtett phpinfo kimenete gyakorlatilag a szerver összes lényeges és támadható paraméterét kiszolgáltatja, azaz érdemes a fejlesztési szakaszt követően tiltani. A disable_classes is hasonló opció.

```
expose_php = Off
```

Nagyon hasonló az Apache-nál már alkalmazott beállításhoz, azaz a HTTP Header-ben megjelenő tényleges PHP fő- és alverziót lehet vele elrejteni. Alapvetően érdemes Off értéket megadni.

```
max_execution_time = 30
```

Meghatározhatjuk, hogy minden egyes PHP folyamat maximum mennyi időt futhat, mielőtt az elemző kilövi. Sok esetben a 30 másodperc kevés szokott lenni, ezért érdemes a saját igényeinkhez igazítani, de figyelve arra, hogy maximum annyit adjunk, amennyi ténylegesen szükséges.

```
max_input_time = 60
```

Az előző opcióhoz hasonlóan másodpercben határozzuk meg, hogy az INPUT meddig tarthat.

```
memory_limit = 128M
```

Ezzel meghatározzuk, hogy a PHP folyamatok mennyi tényleges memóriát fogyaszthatnak a rendszerünkön. Sok esetben a 128MB kevés, szintén kis lépcsőkben emeljük, és arra a tényleges maximumra ami még éppen elég, de a rendszert sem lehet kikészíteni még vele. A jó beállításhoz a Munin grafikonjait és a top, htop parancsokat is segítségül hívhatjuk.

```
error_reporting = E_ALL & ~E_NOTICE
```

Ezzel meghatározhatjuk, hogy a hibanaplózás milyen szinteken legyen bekapcsolva. Fejlesztési időszakban az alap beállítást érdemes meghagyni, később pedig szigorítani kell rajta, mivel túl sok információt árulhatunk el így egy esetleges támadó számára.

```
display_errors = Off
```

Ezzel engedhetjük vagy tilthatjuk, hogy a felhasználó a PHP szkript által okozott hibát megjelenítheti-e. Megint csak fejlesztési szakaszban, amikor az oldal nem publikus (például jelszóval védett), akkor érdemes bekapcsolva tartani, de később kikapcsolni és inkább naplófájlba terelni az ilyen üzeneteket.

```
log_errors = On
```

Ennek használatával megmondhatjuk, hogy az error.log-ban szeretnénk inkább látni a PHP szkriptek hibaüzeneteit.

```
file_uploads = On
upload_max_filesize = 20M
```

Ezekkel a beállításokkal engedhetjük vagy tilthatjuk a fájlfeltöltéseket PHP-n keresztül, valamint meghatározhatjuk azok maximális méretét. Érdemes összehangolni a max_execution_time és a max_input_time = 60 beállítással, mivel ha engedünk több száz MB adatfeltöltést, akkor az vélhetőleg nem fog beleférni az alap beállításoknál meghatározott 30/60-as keretbe, figyelembe véve az itthoni aszimmetrikus internetsebességeket (azaz a feltöltés a legtöbb esetben lassabb mint a letöltési irány).

```
allow_url_fopen = Off
```

A PHP szkript számára tilthatjuk vagy engedhetjük például a HTTP vagy FTP távoli fopen hívást. Ha nincs rá szükségünk (a programozók nem igénylik), akkor érdemes Off értékre állítani.

A php.ini fájlban, illetve az Apache virtualhostonként állítható változói még számos lehetőséget kínálnak számunkra, amelyeket most nem részletezünk.

Phpmyadmin

A Phpmyadmin egy nyílt forrású, PHP nyelven írt eszköz, amely képes webes felületen a MySQL adatbázis-kezelő majdnem teljes körű menedzselésére. Használatával lehetséges adatbázist kezelni és törölni, illetve a különböző egyéb adatbázissal kapcsolatos műveletek nagy részét is elérhetjük ezen a felületen. Híres a biztonsági incidenseiről, ezért kifejezetten fontos, hogy ne alapbeállításokkal használjuk. A szerverem-neve.hu/phpmyadmin alap hivatkozást ne hagyjuk meg neki (nagyon könnyen megváltoztatható, hiszen a legtöbb esetben a telepítés a disztribúció csomagkezelőjével zajlik. Így ebben az esetben csak egy Apache Alias beállítás módosítása szükséges. Ha pedig letöltjük, akkor ne olyan könyvtárba rakjuk be, ami a neve.), illetve érdemes eleve HTTPS felület alatt engedélyezni a használatát csak, valamint egy .htaccess fájlal pluszban védeni. A legtöbb esetben használata szükségszerű, hiszen a távoli MySQL port engedélyezése a programozó számára, ha lehet még kevésbé ajánlatos, mint a phpmyadmin maga.

OpenSSL

Az OpenSSL egy nyílt forrású programcsomag- és könyvtár-gyűjtemény, amely segítségével nagyon sokfajta kriptográfiai műveletet végezhetünk el. A legtöbb Linux disztribúció alaptelepítésének része, valamint Windows alatt is elérhető. Az Apache számára tudunk a segítségével egy önaláírt kulcsot készíteni, amelyet a mode_ssl bekapcsolása után megfelelően befűzve lehallgatással szemben ellenállóbbá tehetjük a hálózati kapcsolatot. Azaz a gyakorlatban egy HTTPS felületet hozhatunk létre, illetve ugyanezt az elkészített kulcsot akár az FTP kapcsolat biztonságosabbá tételére is felhasználhatjuk később.

Weboldalak: webes CMS

Az internet mára a mindennapok részévé vált. A saját weboldal mára a bizalom alapja lett, és ha az emberek valamit nem találnak meg az interneten, akkor már a létezésében, de legalábbis legitimitásában is kételkedni kezdenek. Ez azonban új típusú kihívás elé állította a vállalkozásokat és intézményeket: egy weboldal elkészítése és üzemeltetése sokszor a fő tevékenységtől nagyon eltérő kompetenciákat követel. Ráadásul általában nem csak egyszeri feladat, hiszen egy weboldal folyamatos karbantartást és frissítést igényel. A tartalomkezelő rendszerek ezt a feladatot egyszerűsítik le. Nem kell többé minden cégnek saját weblapjához kiszolgáló szoftver keretrendszert fejlesztenie, mivel számos kiváló szabad szoftver megoldás létezik erre a feladatra.

A tartalomkezelő rendszer feladata, hogy lehetővé tegye tartalmak létrehozását, kezelését és rendszerezését – vagyis mindazt, ami egy honlap üzemeltetésének legfontosabb része. A legismertebb webes tartalomkezelők a WordPress, a Drupal és a Joomla mind szabad szoftverek. Ezek közül a Wordpresshez és a Drupalhoz is elérhető egy-egy kiváló kézikönyv. Ezek az alábbi honlapokon érhetők el:

WordPress könyv: http://szabadszoftver.kormany.hu/sajat-oktatasi-anyagok/

Drupal könyv: http://nagygusztav.hu/drupal-7-alapismeretek

Ezen könyvek elolvasása után az olvasó képes lesz arra, hogy igényei szerint testre szabjon egy WordPress vagy Drupal alapú weboldalt, azon tartalmakat publikáljon, kezelje a felhasználókat és akár online közösséget építsen. Ehhez pedig semmilyen különösebb előképzettséget nem felételez: aki a mindennapokban számítógépet használ, közösségi oldalakat látogat, levelezik és boldogul egy egyszerű szövegszerkesztővel, annak ezeknek a tartalomkezelőknek a használata sem jelent majd különösebb kihívást.

FTP-szerver feladata, telepítése és üzemeltetése

A weboldalak adminisztrálására számtalan megoldás létezik, egyszerűbb esetben a tartalomkezelő felület kínál erre lehetőséget valamilyen bővítmény segítségével, vagy eleve elégséges az a funkcionalitás (cikkek és képek közzététele stb.) amely adott, plusz a phpmyadmin. Ha ennél többet szeretnénk, például hozzáférni a PHP és egyéb fájlokhoz közvetlen módon, akkor kerülnek képbe a különböző hozzáférést biztosító szolgáltatások. A legtöbb esetben még mindig az FTP protokollt[101] használjuk erre, de sokan az OpenSSH beépített megoldását az SFTP-t, vagy az SCP-t alkalmazzák. Elterjedt megoldás volt régebben az scp-only shell használata is, de szerencsére ezt az SFTP már jobbára kiszorította. Manapság még inkább fontosabb alapkövetelmény, hogy akármilyen megoldást is használunk, a lehetőség a szeparációra meglegyen. Azaz mindenki csak a saját könyvtárát láthassa, amikor becsatlakozik a kliensével, illetve mi adhassunk meg kivételeket. Ugyanígy alapkövetelmény ma már az is, hogy ha FTP-t használunk, akkor képes legyen SSL vagy TLS alatt működni, és ne csak az azonosítás menjen az SSL réteg alatt, hanem az adatkapcsolat többi része is. Ezért most olyan klienseket fogunk bemutatni, ahol ezek a dolgok megvalósíthatóak. Fontos ugyanakkor megemlíteni, hogy ma egy mindenki számára hozzáférhető FTP kapcsolatot engedélyezni nagyon nagy merészség. Ma már a fejlesztők jelentős része amúgy sem akar FTP kapcsolaton keresztül fejleszteni, hanem valamilyen verzió követő rendszert használ (subversion, git stb.). Az FTP szerverek biztonsága igen csak kétes, éppen ezért most két megoldást fogunk ajánlani. Ha tehát egy olyan FTP szerver megoldásra van szükségünk, amikor is sok funkcionalitást akarunk elérni, akkor a Pure-FTPd-t javasoljuk használatra, mindenképpen tűzfal mögött. Azaz a gép 21-es portja ne legyen bárki számára elérhető az interneten, hanem szabályozzuk le akár port kopogtatással, fix IP-vel, vagy dinamikus DNS + tűzfal szabályokkal. Ha pedig egy anonymous szervert szeretnénk felállítani, amely csak a mi publikus állományainkat szolgálja ki, akkor használjuk a jóval biztonságosabb vsftpd-t, csak olvasható üzemmódban, nyitott 21-es porttal.

Pure-FTPd[102]

Ahogyan a weboldal első kiemelt része is említi „Security first”, azaz a megalkotóknak a legfontosabb a biztonság volt. Támogatja a chroot-ot[103], a virtuális szervereket, az SSL/TLS kapcsolatot vegyes és kizárt üzemben is. Azaz beállítható, hogy csak az azonosítás vagy az azonosítás és az adat is titkosított réteg alatt menjen. Könnyen konfigurálható és virtuális felhasználói adatbázisával együtt egészen nagy rendszerek kialakítására is kényelmesen használható. Segítségével akár a www-data felhasználó nevében felvett virtuális felhasználókat tudunk létrehozni, így biztosítva, hogy a safe mode-ban futó PHP szkriptek se álljanak le bizonyos esetekben a futtató környezet (Apache esetén a www-data) és a fájljogosultságok eltérése miatt (safe mode gid). Szabályozható vele az anonymous és a tényleges felhasználók sávszélessége is, ezzel biztosítva, hogy egy-egy nagyobb feltöltési sávszélességgel rendelkező felhasználó ne tudja a végletekig leterhelni a szervert,

101. http://hu.wikipedia.org/wiki/File_Transfer_Protocol
102. http://www.pureftpd.org/project/pure-ftpd
103. http://hu.wikipedia.org/wiki/Chroot

valamint megmondható az is, hogy a Pure-FTPd mekkora terhelés[104] mellett szolgáljon még ki. Támogatja az FXP (server-to-server) protokollt, amely segítségével 2 szerver között tudunk nagyobb mennyiségű adatot úgy mozgatni, hogy az otthoni kis sávszélességű vonalunkon nem megy át az adat, ezzel is nagyban gyorsítva az adatátvitelt. Nagyon hasznos funkció továbbá, hogy a Passive Port Range opcióval meghatározható azon portok tartománya, ahol a passzív mód esetén kiszolgál. Így rögzíthetjük ezen portokat egy egyszerű Iptables szabály segítségével, ezzel is megkönnyítve a másik oldalról jövő NAT mögüli kapcsolatok dolgát.

Telepíteni a már megszokott apt-get install pure-ftpd paranccsal lehet, ez telepíteni fogja a pure-ftpd-common csomagot is mint függőséget. A telepítés után a konfigurációs fájlokat az /etc/pure-ftpd/conf könyvtárban találjuk. A Pure-FTPd-t praktikusan ebbe a conf könyvtárba elhelyezett szöveges fájlok segítségével (és a fájlokba beleírt opció állapotával[105]) tudjuk finomhangolni, ez a gyakorlatban így néz ki (a conf könyvtár tartalma):

```
/etc/pure-ftpd/conf# ls -f
  ..                  AnonymousCantUpload         MaxLoad
  MaxClientsNumber    PureDB                      FSCharset
  Daemonize           BrokenClientsCompatibility
  MaxIdleTime         DisplayDotFiles             MaxClientsPerIP
  AllowAnonymousFXP   AnonymousCanCreateDirs      AnonymousBandwidth
  PAMAuthentication   AltLog                      UserBandwidth
  TLS                 NoAnonymous                 AllowUserFXP
  UnixAuthentication  ChrootEveryone              MinUID
```

Nézzük egyenként az opciók jelentését és a fájlok tartalmát:

NoAnonymous [yes/no]

Ezzel engedhetjük vagy tilthatjuk az anonymous (névtelen) felhasználó bejelentkezését.

AnonymousCantUpload [yes/no]

Ezzel engedhetjük vagy tilthatjuk az anonymous felhasználónak a fájlfeltöltést. Ha megtiltjuk, akkor csak letöltheti az anonymous módban elérhető fájlokat és könyvtárakat. Alapesetben állítsuk no-ra a fájl tartalmát.

AnonymousCanCreateDirs [yes/no]

Ezzel engedhetjük vagy tilthatjuk szintén a vendég felhasználó számára a könyvtárak létrehozását, alapesetben állítsuk no-ra.

AnonymousBandwidth [KB/Sec]

Ezzel beállíthatjuk, hogy a vendég felhasználó mekkora sávszélességgel forgalmazhat. Ha például egy népszerű szabad szoftvert teszünk ki anonymous FTP-re, akkor mindenképpen érdemes ezt az értéket egy olyan tapasztalati úton kipróbált maximumra hangolni, amely még nem veszélyezteti a lemez I/O műveletekből fakadóan a gép stabil üzemét, még akkor sem ha nagyon nagy az érdeklődés az adott fájl iránt.

AllowAnonymousFXP [yes/no]

Ezzel engedhetjük vagy tilthatjuk, hogy a vendég felhasználó az FXP[106] protokoll szerint, a kliens kihagyásával 2 szerver között mozgathasson direktben adatot.

104. http://hu.wikipedia.org/wiki/Load_(sz%C3%A1m%C3%ADt%C3%A1stechnika)
105. http://download.pureftpd.org/pure-ftpd/doc/README
106. http://en.wikipedia.org/wiki/File_eXchange_Protocol

AllowUserFXP [yes/no]

Ez az előző opció normál, azaz beazonosított felhasználókra szánt változata.

AltLog [clf:/var/log/pure-ftpd/transfer.log]

Ezzel beállíthatjuk az átviteli napló pontos helyét. Alapesetben a /var/log/pure-ftpd/transfer.log fájlban érdemes tárolni a felhasználók aktivitását.

ChrootEveryone [yes/no]

Ezen opció segítségével beállíthatjuk, hogy minden felhasználó csak a számára létrehozott home környezetet láthassa, gyakorlatilag / (gyökér) környezetként. Ez különösen hasznos, ha több esetleg „idegen" felhasználót vagyunk kénytelenek beengedni a szerverre. A szerver biztonsága érdekében ez alapbeállítás kellene, hogy legyen, ezért állítsuk Yes-re.

Daemonize [yes/no]

Ezzel adhatjuk meg, hogy a Pure-FTPd külön démonként vagy az inetd[107] részeként fusson. Javasolt démon módban futtatni.

DisplayDotFiles [yes/no]

Mutassa-e a ponttal kezdődő nevű, rejtett fájlokat, mint például a .htaccess-t. Javasolt ezt kikapcsolni, azaz no-ra állítani, és csak akkor engedni a klienseknek, hogy lássák például a .htaccess-t, ha erre ténylegesen szükségük van.

FSCharset [UTF-8]

Ezzel beállítjuk az alapértelmezett karakterkódolást, amely ma már praktikusan az UTF-8.

MaxClientsNumber [szám]

Ezzel meghatározhatjuk, maximum mennyi felhasználót enged be egy időben a rendszer. Nagy terheltségű rendszer esetén érdemes alulról felfele haladva tesztelve beállítani, hogy a túlterheltség ne akadályozza a szerver működését.

MaxClientsPerIP [szám]

Ezzel meghatározhatjuk, hány klienst engedjen egy adott IP-címről. Tekintve, hogy manapság már az otthoni hálózatok is NAT mögött vannak, távolról 1 IP-nek látszanak, ami azt jelenti, hogy akár egy egész alhálózat lehet 1 db IP-cím mögött (amely azért az otthoni felhasználókra nem jellemző), praktikusan érdemes 1-5 közé tenni ezt a számot. Ha sok kérést kapunk Proxy vagy nagyvállalati tűzfal gépek mögül, akkor érdemes növelni ezt a számot.

MaxIdleTime [perc]

Ezzel percben megadhatjuk, mennyi inaktivitás után szakítsa meg a kapcsolatot a szerver a klienssel. Mivel ezt az opciót a legtöbb klienssel felül lehet bírálni, ezért nem érdemes túl nagy számot választani, praktikusan 5-15 perc közötti értéket adjunk meg.

MaxLoad [szám]

A túlterhelés megakadályozására alkották meg ezt a remek opciót, amely nem engedi a Pure-FTPd-nek, hogy egy bizonyos rendszerterheltség (load) felett további erőforrásokat emésszen fel. Általánosságban elmondható, hogy a load[108] egy olyan iránymutató szám a Unix/Linux rendszerekben, amely esetében az 1-es alatti érték jelképezi az üzemszerű erőforrás-felhasználást. Ez nem

107. http://manpages.ubuntu.com/manpages/precise/man8/inetd.8.html
108. http://en.wikipedia.org/wiki/Load_(computing)

azt jelenti, hogy a MaxLoad értékének 1-et kellene beírnunk, de érdemes a top vagy a htop program segítségével monitorozni az FTP szerver és a szerver terheltségét, és egy olyan értéket meghatároznunk, amely a felhasználók viselkedése és a hardver tűréshatárán belül egy ésszerű érték. Ez jelenti azt is, hogy ekkor a lemez I/O várakozás még nem okoz nagyobb loadot és a memóriafelhasználás sem terelődik át a swap tartományba. A példa kedvéért, ez egy nagyobb terheltségű FTP szerveren jelenleg 12-es érték, de ez teljesen szubjektív és egyedi mérésekre alapozott.

TLS [0-3]

Ezzel engedhetjük vagy tilthatjuk, és meghatározhatjuk az SSL használatát a következőképpen:

0: az SSL/TLS réteg tiltva van.

1: engedjük az SSL/TLS-t és a sima titkosításmentes kapcsolatot is

2: visszautasítja a nem SSL/TLS mechanizmussal kezdődő azonosítási kísérleteket, anonymous kapcsolat esetében is.

3: visszautasítja a titkosításmentes kapcsolatot és SSL/TLS alapú adatkapcsolatot épít ki.

Ugyanakkor az adatkapcsolati réteg esetében is kényszeríti az SSL/TLS kapcsolatot.

A 2-es és 3-as opció esetében első lépésben létre kell hoznunk az SSL kulcsot és bemásolnunk a következőképpen:

```
mkdir -p /etc/ssl/private
  openssl req -x509 -nodes -newkey rsa:4096 -keyout \
   /etc/ssl/private/pure-ftpd.pem \
   -out /etc/ssl/private/pure-ftpd.pem
  chmod 600 /etc/ssl/private/*.pem
```

Az Ubuntu LTS-ben lévő Pure-FTPd TLS/SSL támogatással kerül csomagolásra, így az SSL kulcs elkészítése, valamint az opció bekapcsolása után további teendőre nincs szükség.

UserBandwidth [KB/s]

Akárcsak az anonymous esetében, itt is KB/Sec-ben adhatjuk meg az azonosított felhasználó maximális átviteli képességét. Vigyázzunk: az esetlegesen ugyanabban a hálózatban lévő felhasználók akár gigabites forgalmat is generálhatnak.

MinUID [uid]

Ezzel az opcióval adhatjuk meg az azonosítás során, hogy mely felhasználók csatlakozhatnak, hiszen jellemzően a sima felhasználói szint 1000-es UID-nál kezdődik, így a root és a többi 1000 alatti felhasználó még a jó jelszó megadása után sem tud csatlakozni, amely kifejezetten kívánatos óvintézkedés. Így tartalma legyen 1000.

PassivePortRange [szám szám]

Ez egy rendkívül fontos opció, mivel jelen esetben kényszeríthetjük, hogy az FTP forgalom passzív üzemmód[109] esetén milyen tartományban üzemeljen, így a tűzfalszabályokat is könnyedén mögé igazíthatjuk a következő módon:

A fájl tartalma legyen „42 000 42 100”, sima szóközzel elválasztva, majd pedig helyezzünk el a tűzfalunkban egy ehhez hasonló szabályt:

```
#FTP Passive Port + SSL
```

109. http://en.wikipedia.org/wiki/File_Transfer_Protocol

```
  $IPTABLES -A INPUT     -p tcp --sport 20 -m state --state ESTABLISHED,RELATED
  -j ACCEPT
  $IPTABLES -A OUTPUT -p tcp --dport 20 -m state --state ESTABLISHED -j ACCEPT
  $IPTABLES -A tcp_packets -i eth0 -p tcp -m tcp --dport 42000:42100 -j ACCEPT
```

Ezek után a kliensen az SSL/TLS és a passzív mód kapcsolókat kell bekapcsoltatni a felhasználóval.

A Pure-FTPd és a virtuális felhasználók

Vannak olyan esetek, amikor (pl. kvóta[110] miatt) szeretnénk virtuális felhasználókkal dolgozni. Vagy csak egyszerűen szeretnénk, hogy ha a /var/www alatt lévő, akár több száz egyenként más és más személyhez tartozó egyedi virtuális hostok (önálló weboldalak) mind a www-data felhasználó jogaival futnának, de csak virtuális felhasználók érhetnék el, mert nem akarunk nekik más (például IMAP vagy POP3) szolgáltatásokat is nyújtani, csak FTP elérést a saját weboldalaikhoz, és azt is meghatározott lemezkapacitással. Ebben az esetben egy remek lehetőség a virtuális felhasználók használata, nézzük hogyan is kell ezt beállítani. Először is állítsuk be a fenti leírás szerint a Pure-FTPd-t, majd adjuk ki a következő utasításokat:

```
cd /etc/pure-ftpd/conf
  echo 'no' > PAMAuthentication
  echo 'no' > UnixAuthentication
  echo '/etc/pure-ftpd/pureftpd.pdb' > PureDB
  ln -s ../conf/PureDB /etc/pure-ftpd/auth/50pure
```

Ezekkel beállítjuk, hogy tudjon a saját adatbázisa szerint is azonosítani, egyben tiltjuk, hogy a lokális felhasználókat azonosítsa.

A virtuális felhasználók saját könyvtárai a /var/www/ alatt lesznek létrehozva, például:

```
mkdir /var/www/www-site1.hu
```

Ezzel létrehozzuk a site1.hu gyökérkönyvtárát.

```
chown www-data:www-data -R /var/www/www-site1.hu
```

Majd átadjuk a www-data tulajdonosnak a könyvtár jogait.

```
pure-pw useradd site1hu -u www-data -g www-data -d /var/www/www-site1.hu -N 100
```

Létrehozzuk a virtuális felhasználót 100 MB lemezkvótával és a feltett kérdésre 2x beírjuk az új jelszavát.

```
pure-pw mkdb
```

A Pure-Ftpd adatbázisát frissítjük, hogy tudjon az elkészített felhasználóról.

A már elkészített felhasználó tulajdonságait a következőképpen listázhatjuk:

```
pure-pw show site1hu
```

Az összes felhasználót megnézhetjük a *pure-pw list* parancs segítségével, ha pedig jelszót kell változtatnunk egy már létrehozott felhasználónak, azt *pure-pw passwd site1hu* parancs segítségével tudjuk megtenni, de minden változtatás utána az adatbázist is frissíteni kell a már ismertetett pure-pw mkdb parancs segítségével. Ezek után már nyugodtan tesztelhetjük a *service pure-ftpd restart* kiadása után.

110. http://en.wikipedia.org/wiki/Disk_quota

Természetesen számtalan egyéb[111] opciója is létezik még a Pure-FTPd-nek, azonban a legfontosabb és legnépszerűbb opciókat felsoroltuk.

Vsftpd[112]

A projekt weboldala szerint a Vsftpd valószínűleg a legbiztonságosabb és leggyorsabb FTP szerver a Unix rendszerű gépek esetében. Az biztos, hogy akárcsak a Pure-FTPd esetében számtalan biztonsági és működést segítő opcióval rendelkezik[113]. Ugyancsak támogatja a virtuális felhasználókat és az SSL/TLS-t is. Érdekessége, hogy a legnagyobb Linux terjesztések majdnem kivétel nélkül Vsftpd-t használnak a nagy terheltségű ISO és egyéb kiszolgálásaikra. Ahogyan a bevezetőben ajánlottuk, a Vsftp remek megoldás pl. a publikus fejlesztéseink kiszolgálására, publikálásra. Ha tehát a cél az, hogy mi (pl. SCP-vel) felmásoljuk a Vsftp rootjába az állományainkat (changelog, tgz stb.), majd ezeket az FTP szolgálja ki elvihető (readonly) üzemmódban, akkor következőképpen érdemes eljárni:

```
sudo apt-get install vsftpd
```

Majd szerkesszük a /etc/vsftpd.conf állományt:

```
anonymous_enable=YES
```

Megengedjük a vendég csatlakozásokat.

```
guest_username=ftp
```

Beállítjuk az anonymous felhasználót.

```
anon_root=/var/ftp
```

Beállítjuk az anonymous document root-ot.

```
local_enable=NO
```

Megtiltjuk a lokális felhasználók belépését.

```
write_enable=NO
```

Megtiltjuk az írást.

```
force_dot_files=YES
```

Engedjük, hogy a ponttal kezdődő állományokat is listázza.

```
hide_ids=YES
```

Megtiltjuk, hogy az owner és group adatokat elárulja.

```
max_per_ip=2
```

Limitáljuk, hogy egy adott IP-címről maximum 2 csatlakozás érkezhessen.

```
max_clients=20
```

Beállítjuk, hogy hány kliens csatlakozhat egy adott időben.

```
xferlog_enable=YES
```

Beállítjuk, hogy az alapértelmezett naplóállományba naplózzon (/var/log/vsftpd.log).

111. http://download.pureftpd.org/pure-ftpd/doc/README
112. https://security.appspot.com/vsftpd.html
113. https://security.appspot.com/vsftpd.html#features

```
listen_address=172.20.1.1
```

Beállítjuk, milyen IP-címen legyen aktív.

```
listen_port=21
```

És ott milyen porton.

Ha készen vagyunk, a *service vsftpd restart* parancs segítségével újraindítjuk, és tesztelhetünk is.

ProFTPD[114]

Talán az egyik legtöbb platformon futó, rendkívül széles körben konfigurálható FTP démon, amelynek hosszú múltja van. Sajnos azonban a hosszú múltban adódtak kellemetlen biztonsági incidensek is, talán ezért a projekt főoldalán csak a magas konfigurálhatóságot említik első sorban. Ennek ellenére a ProFTPD igen megbízható és sokoldalú FTP szerver megoldás.

Számtalan más szabad szoftveres FTP szerver megoldás létezik még, de ma ezek a leginkább elterjedtek. Amennyiben tájékozódni akarunk a többi megoldás tudását és elérhetőségét illetően, akkor a Wikipedia[115] egy remek táblázatban ezt összefoglalja számunkra.

SFTP[116]

Az SFTP alrendszer részben az SCP utódja, valójában az SSHD egy jól elkülönített alrendszere. Az SCP egy remek eljárás az SSH titkosított rétegein belül fájlátvitelre. Ugyanúgy támogatja az SSH összes biztonsági megoldását, mint a parancssori mód. Azaz érvényes rá az Allow Users direktíva, és a kényszerített kulcshasználat is, amikor a jelszavas azonosítást kikapcsoljuk, és RSA vagy DSA kulcsok segítségével végezzük el az azonosítást. De használható vele az OTP (one time password) azonosítás is. Gyengesége, hogy egy standard SSH szerver felépítése esetén az SCP jogot nem lehet jól elkülöníteni az SSH parancssori felület jogaitól, valamint nehéz chroot-ba kényszeríteni. Van ugyan rá egy megoldás az scponly shell[117], amikor is egy speciális chroot-ba kényszerített, nagyon limitált shell környezetet kap a felhasználó, amely segítségével a sima SSH-t már nem tudja használni, csak egy adott könyvtár fölötti részhez fér hozzá. Ezzel jellemzően az automatizált mentéseket lehet biztonságosabbá tenni. Az SFTP esetében azonban már az SSH konfigurációjában megmondhatjuk, hogy ki milyen séma szerint férhet és mihez. Teljesen biztonságos, mivel minden adat az SSH BINARY csatornáján keresztül megy, így továbbra is lehetőség van az összes biztonsági azonosítás és korlátozás használatára. Gyakorlatilag egy minimális odafigyeléssel ma talán ez a legmegfelelőbb megoldás a sima FTP kiváltására. Szerencsére a kliensek is számos platformon támogatják, így a fejlesztőknek is sok eszköz áll rendelkezésükre, sőt manapság már a kereskedelmi weblapfejlesztő és verziókövető rendszerek is támogatják ezt a módot. Részletesebb beállítási útmutatót a Távoli adminisztráció fejezet SFTP részében találunk.

114. http://proftpd.open-source-solution.org/
115. http://en.wikipedia.org/wiki/List_of_FTP_server_software
116. http://en.wikibooks.org/wiki/OpenSSH/Cookbook/SFTP
117. https://github.com/scponly/scponly/wiki

Webstatisztika-készítő szoftverek

AWStats[118]

Ez egy alapvetően Perl nyelven írt, igen részletes statisztikakészítő program. Szükséges hozzá CGI támogatás az Apache szerveren belül, éppen ezért egy kicsit komplikáltabb az üzemeltetése is egy sima HTML-t kezelő programnál. Viszont sokkal több és részletesebb lehetőséget kapunk. Tudása gyakorlatilag a kereskedelmi statisztikakészítő programok tudásával vetekszik. A pontos funkciók részletezése a honlapján[119] megtalálható. Üzemeltetésénél érdemes számításba venni, hogy CGI módban fut, ezért érdemes korlátozni és védeni esetleg https és htaccess használatával az elérhetőségét. Rendelkezik magyar nyelv támogatással is.

Webalizer[120]

Az egyik legrégebbi és leggyorsabb (hiszen C nyelven írták) naplóelemző szoftver. Beállítása és üzemeltetése is igen egyszerű. Az Awstats-szal ellentétben a Webalizer futtatását vagy emberi beavatkozással parancssorból (vagy tömegesen szkriptelve) vagy pedig cron-ból időzítve végezhetjük. Nincsen tehát gyakorlatilag támadási felülete, ugyanis sima HTML kódot + képeket publikál egy előre meghatározott könyvtárba. Honosított számtalan nyelvre, köztük a magyarra is. Tudása nagyon hasonló az Awstats-hoz, azonban a statikus tartalom kezelése miatt kevesebb extrát tud. Egyszerű felhasználású eszköznek igen kiváló. Bár a publikált területen a HTML oldalak támadási felületet nem kínálnak, azonban maga a statisztika adhat lehetőséget visszaélésre (például: egy azonosítás nélküli felület, amelyet sokszor használunk, benne lesz a publikus adatok között) ezért érdemes szintén https és htaccess védelemmel ellátni.

Piwik[121]

Egy PHP és MySQL alapokon nyugvó nyílt forráskódú, teljesen valós időben működő webforgalom-analizáló szoftver. A weboldal tanúsága szerint több mint 320.000 weboldal használja és alternatívái akarnak lenni a Google Analytics-nek. Jelenleg 46 nyelvet támogat, köztük a magyart is. Tudása összetettebb és szélesebb körű mint társaié, valamint nagy előnye, hogy az adatainkat nem szolgáltatjuk ki egy 3. félnek, azaz saját rendszerünkön belül tároljuk az adatokat.

BBclone[122]

Akár csak a Piwik, PHP alapokon nyugvó valós idejű statisztikaszoftver, ahol szintén nem kell kiszolgáltatni az adatainkat egy kereskedelmi cégnek. Nyílt forrású és számos nyelven támogatott, köztük magyarul is.

118. http://awstats.sourceforge.net/
119. http://awstats.sourceforge.net/#FEATURES
120. http://www.webalizer.org/
121. http://piwik.org/
122. http://bbclone.de/

Fájl szerver feladata, telepítése, üzemeltetése

Fájl és nyomtatószerver heterogén hálózatok részére

Miről is szól mindez?

Az alábbi írásnak nem célja a Samba összes lehetőségének bemutatása, ellenben remélhetően a segítségével könnyen és gyorsan lehet egy már működő rendszer üzemeltetéséig eljutni, illetve egy teljesen új rendszert létrehozni. A kevesebb néha több – ez volt az elv a készítés közben.

Történeti áttekintés

A Linux terjedésével jogos igényként merült fel, hogy alkalmas legyen Windows alapú számítógépek számára fájlszerver szolgáltatást nyújtani, ezzel kiváltva a Microsoft alapú szervereket. Ezt az igényt vette észre a Samba fejlesztői[123] csapata, és hozta létre a ma már széles körben ismert Samba szervert. 1993-ban jelent meg az első változat Nbserver 1.5 néven, ma pedig a Samba 4.1.1 verziónál tartunk. A fokozatosan teret hódító négyes sorozat mellett jelenleg rendkívül elterjedt a 3.6 is, amely sok működő kiszolgálón megtalálható. A Samba sok helyen bizonyított már, 15-20 gépes kis cégeknél és 100-as gépszám feletti iskolákban is évek óta üzemeltetnek vele megbízhatóan kiszolgálókat. Mivel a Linux kliensek is alkalmasak Windows-megosztások elérésére, így vegyes kliens környezetben is tökéletes választás lehet az eltérő operációs rendszert futtató ügyfelek kiszolgálására, összekapcsolására.

Samba verziók

3.6-os sorozat

A 3.6-os sorozat tulajdonképpen egy Windows NT 4.0 képességeivel rendelkezik, alkalmas fájl- és nyomtatókiszolgálóként működni Windows alapú számítógépek számára, továbbá WINS szolgáltatást és tartományi beléptetést végezni. A Windows XP Professional operációs rendszer gond nélkül beléptethető egy 3-as sorozatú Samba tartományba, a Windows7 viszont már igényel ehhez egy egy soros registry módosítást. Ez így 2013-ban talán kevésnek tűnik, és sokan a Samba szemé-

123. http://www.samba.org/

re is vetették, hogy nincs valós Active Directory[124] támogatás és GPO[125] sem benne, de ha megnézzük a valós igényeket a kis és közepes cégeknél, akkor azt látjuk, hogy valójában pár közös mappára, saját hálózati tárhelyre van igény, esetleg egy vándor profilra[126], hogy bármely géphez átülve ugyanaz a környezet fogadja a dolgozót. Ennél többre ritkán van igazán szükség. Ezeket az igényeket jelenleg bőven kiszolgálja a 3.6-os sorozat. A továbbiakban először a hármas sorozatú Sambát nézzük majd át, hogyan lehet vele létrehozni egy tartományt, megosztásokat készíteni, és ahhoz megfelelő jogosultságokat adni, majd utána ugyanezt a 4-es sorozattal is megtesszük. Miért? Mert jelenleg sok hálózatban ilyen működik, a Linux disztribúciók jelentős részében most is ez az alapértelmezett, a jelenlegi stabil Debian[127] verzió is ezt ajánlja használatra, és csak egy 4.0.0 béta csomagot tartalmaz az újabb verzióból.

4-es sorozat

A 4-es sorozat alkalmas már egy korszerű Microsoft szerver kiváltására is, rendelkezik AD, GPO és FSMO[128] támogatással is. Teljes értékű tagja lehet egy Windows Active Directory-nak (itt már nincs PDC, BDC megkülönböztetés[129]), és nincs szükség az újabb klienseken registry piszkálásra sem. Új szerver telepítésekor érdemes megfontolni, hogy rögtön a 4-es sorozatú Sambát telepítsük; meglévő régebbi telepítésekor pedig, ha az AD által hozott plusz szolgáltatásokat nem igénylik, bátran hagyhatjuk a korábbi 3-as sorozatot fent, később bármikor dönthetünk a migráció mellett.

Samba 3.6 telepítése, konfigurálása

A döntés

Mielőtt elkezdjük a telepítést, szükséges végiggondolni, hogy mit is szeretnénk a Sambával. Szeretnénk egyszerű jelszó nélküli megosztásokat, vagy esetleg név és jelszó párost igénylő megosztásokat szeretnénk készíteni? A Samba ugyanis mindegyikre képes, attól függően, mit állítunk be a konfigurációs állományában. A fentiek azonban már egy pár gépes hálózatban is kérdésesek, akár néhány gép esetén is érdemes lehet tartományvezérlőnek beállítani a Sambát. Az első legfontosabb érv e mellett, hogy így a megosztásokhoz való hozzáférést sokkal finomabban tudjuk szabályozni, a felhasználókat egy helyen tudjuk felvenni, és ha ezt a lehetőséget választjuk, akkor módunk van vándor profilokra is, azaz a felhasználók bármely munkaállomáson jelentkeznek be, a megszokott környezet várja őket. Ebbe beletartozik az asztal háttere és a rajta véletlenszerűen szétdobált dokumentumok, parancsikonok is, valamint a Dokumentumok mappa is a böngésző-előzményekkel együtt. Felhasználói szempontból ez egy igen fontos szempont. Mivel nem igényel a szerveren jelentős többletmunkát, ugyanakkor jelentős előnyökkel jár, így egy tartományt hozunk most létre. Szintén fontos, hogy a felhasználókat próbáljuk csoportokba rendezni aszerint, kiknek mihez szeretnénk jogosultságokat adni, illetve térképezzük fel előre milyen megosztásokra van szükség, és ezekhez a megosztásokhoz mely felhasználói csoportok milyen jogokkal férhetnek hozzá. A jó tervezés fél siker az új szerverünk létrehozásához.

124. http://hu.wikipedia.org/wiki/Active_Directory
125. http://en.wikipedia.org/wiki/Group_Policy
126. http://en.wikipedia.org/wiki/Roaming_user_profile
127. https://wiki.debian.org/DebianWheezy
128. http://hu.wikipedia.org/wiki/Flexible_single_master_operation
129. http://en.wikipedia.org/wiki/Primary_Domain_Controller

A példa

Jelen példában egy elképzelt iskola számára hozunk létre tartományvezérlőt. A következő felhasználói csoportokat hozzuk létre:

– diak (diákoknak)
– tanar (tanároknak)
– adminisztracio (adminisztratív dolgozóknak)
– admins (rendszergazdák csoportnak)

Ezen csoportokba tartozó felhasználók a következő megosztásokhoz férhetnek majd hozzá:

– publikalas (tanárok számára írható/olvasható, diákok számára olvasható, más nem éri el)
– adminiszt (adminisztratív dolgozók számára írható/olvasható, más nem éri el)
– adminiszt2 (adminisztratív dolgozók és tanárok is írhatják/olvashatják, más nem éri el)
– hr (csak a két HR-es éri el teljes joggal, más sehogy)
– rendszergazda (rendszergazda teljes joggal eléri, mindenki más csak olvasással)
– home (minden felhasználónak egy saját home könyvtár, amit csak ő ér el)

Fenti példa teljesen elégséges lesz a Samba beállításainak megismeréséhez.

Előkészületek:

A Sambát egy Debian Wheezy-re fogjuk telepíteni, ez most az aktuális stabil Debian rendszer. Az alaptelepítésre van szükség. Az első dolog, amit át kell gondolni az a particionálás, hogy legyen egy olyan partíciónk, amelyre a Samba dolgozni fog. Amennyiben a felhasználói saját (home) könyvtárakat el szeretnénk érni Linux alól, mert mondjuk a felhasználóknak van postafiókjuk, és a Maildir könyvtáraikat úgy is ott tároljuk, vagy esetleg később a felhasználók egy részének szeretnénk sftp vagy egyéb elérést biztosítani, akkor érdemes megfontolni, hogy a Samba a felhasználói saját könyvtárakat a rendes /home partíción tárolja. Ebben az esetben érdemes megfontolni, hogy a mentés lényegesen egyszerűbb lehet, ha minden fontos felhasználói adat egy helyen van, így érdemes a közös könyvtárakat is ugyanitt tárolni. Ezzel elérhetjük azt, hogy a /home partíció mentésével az összes felhasználói adat, a közös könyvtárak tartalma is mentésre kerül (sőt mi még a netlogon könyvtárat [130]is oda tesszük). Ezeket figyelembe véve mi most a /home partíción fogjuk az összes megosztást elkészíteni.

Érdemes a kvótát bekapcsolni, mivel ezzel elejét vehetjük, hogy a felhasználók túlságosan sok helyet foglaljanak a rendszeren, ezt az usrquota és a grpquota beszúrásával érhetjük el. Így néz ki a kérdéses partíció bejegyzés az /etc/fstab alatt:

```
/dev/cciss/c0d0p10 /home           xfs     nodev,nosuid,noexec,usrquota,grpquota
  0       2
```

Az, hogy milyen fájlrendszert használjunk, hitvita kérdése lehet, az eddigi tapasztalatok az XFS-el igen kedvezőek voltak.

A gépben most két hálózati kártya van, mivel kifelé is nyújt más szolgáltatásokat a kiszolgáló. (Nem ideális, de kisvállalati és iskolai környezetben jóval gyakoribb, mint azt gondolnánk.) A belső hálózat a 10.0.0.0/255.0.0.0. Feltételezzük, hogy a gépen van Internet-kapcsolat, így indulhat a telepítés.

130. http://www.thenetworkencyclopedia.com/d2.asp?ref=1387

Telepítés:

Írjuk be a következőket:

```
apt-get install samba
```

A kérdésre igennel válaszolva a következő csomagok kerülnek telepítésre: samba-common, samba-common-bin, tdb-tools.

Ha kiadjuk a ps ax parancsot már láthatjuk is, hogy fut a samba démon!

A telepítés során létrejött az alapértelmezett konfigurációs állomány az /etc/samba alatt smb.conf néven. A Samba összes beállítását ebben az /etc/samba/smb.conf állományban kell elvégeznünk!

Nagyon fontos, hogy a Sambában lévő Windowsos csoportokat meg kell feleltetni a Linuxos csoportoknak. Ezt a következő lépésekkel tehetjük meg:

Linux csoportok létrehozása:

```
groupadd admins
groupadd diak
groupadd tanar
groupadd adminisztracio
```

Windows csoportok megfeleltetése a **net** paranccsal:

```
net groupmap add ntgroups="Domain Admins" unixgroup=admins rid=512 type=d
net groupmap add ntgroups="Domain Users" unixgroup=users rid=513 type=d
net groupmap add ntgroups="diak" unixgroup=diak
net groupmap add ntgroups="tanar" unixgroup=tanar
net groupmap add ntgroups="adminisztracio" unixgroup=adminisztracio
```

Ha ezekkel megvagyunk, a telepítés késznek tekinthető, és jöhet a konfigurálás.

Konfigurálás

A továbbiakban az /etc/samba/smb.conf szöveges állományt szerkesztjük. Nézzük sorban a legfontosabb és minket érintő beállításokat.

Globális beállítások:

```
[global]
```

Ez vezeti be a globális beállításokat, kötelező a megléte!

```
interfaces = eth1 127.0.0.1/8
  bind interfaces only = yes
```

Méltatlanul elfeledett beállításpár. Többlábú gépen megadhatjuk mely interfészen fogadja a kéréseket a Samba, így biztosan nem fogja a külvilág felé elérhetővé tenni a szolgáltatásait.

```
hosts allow = 127.0.0.1/8 10.0.0.
```

Csak a localhost-ról és a 10.0.0.X címekről engedjük a hozzáférést. A localhost-ot mindenképp engednünk kell, különben nem működik a Samba, a 10.0.0.X pedig a saját belső hálózatunk gépeinek egy csoportja.

```
hosts deny = 0.0.0.0/0
```

Biztos ami biztos minden egyéb címről még külön megtagadjuk a hozzáférést.

```
admin users = administrator
```

Mi a tartományi rendszergazda neve

```
unix password sync = yes
  pam password change = yes
```

Erre a két beállításra van szükség, ha engedélyezni szeretnénk, hogy a Windowsos gépeiken a felhasználók módosíthassák a jelszavaikat.

```
netbios name = SZERVER3
```

Ez lesz a szerver neve, ez látszik a tallózólistában. Igen javasolt a gép host nevével megegyező nevet választani.

```
server string = KOZPONTI_LINUX_SZERVER
```

Ez jelenik meg leírásként a szerverről.

```
workgroup = LOK
```

Ez lesz a tartomány neve.

```
passdb backend = smbpasswd
```

Itt határozzuk meg milyen módon tárolja a jelszavakat. Több egyéb lehetőség is van,[131] mi a 2.2 Sambákkal való kompatibilitás miatt választottuk ezt (azért mert jó eséllyel találkozhatunk még arról frissített rendszerekkel is).

```
os level = 254
  preferred master = yes
  domain master = yes
  local master = yes
```

Ha azt szeretnénk, hogy mi legyünk a főtallózók a tartományban (és legtöbbször ezt szeretnénk) akkor a fenti beállításokra van szükség.[132]

```
security = user
```

Itt határozzuk meg a biztonsági módot[133] (share vagy user szinteken, összesen 5 van).

```
domain logons = yes
```

Mivel szeretnénk PDC (elsődleges tartományvezérlő) lenni, így szükséges beállítás.

```
encrypt passwords = true
```

Titkosított jelszavak megkövetelése

```
name resolve order = wins bcast host lmhosts
```

Hogyan történjen a névfeloldás. Amennyiben nem szeretnénk sok broadcast üzenetet a hálózatunkon, úgy a wins-et tegyük előre.

```
wins support = yes
```

Itt engedélyezzük, hogy szerverünk WINS[134] szerverként is működjön. Lényegesen csökken a broadcast üzenetek száma a hálózaton, ha ezt engedélyezzük.

131. http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/passdb.html
132. http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/NetworkBrowsing.html
133. http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/ServerType.html
134. http://hu.wikipedia.org/wiki/Windows_Internet_Name_Service

```
time server = yes
```

Időszerverként is működjünk a klienseknek.

```
display charset = UTF8
   unix charset = UTF8
   case sensitive = no
   default case = lower
   preserve case = yes
   short preserve case = yes
```

Ezekkel a beállításokkal érjük el, hogy az ékezetes karakterek mindenhol korrektül jelenjenek meg (Windows és Linux alatt is), és rendben legyen a kis- és nagybetűs fájlnevek eltérő kezeléséből adódó különbség.

```
add machine script = /usr/sbin/useradd -d /dev/null -g 100 -s /bin/false %u
```

Ennek a beállításnak köszönhetően nem kell a szerveren előre létrehozni a gépek fiókjait, hanem amikor tartományba léptetjük a gépeket, azok automatikusan elkészülnek.

```
logon path = \\%L\profiles\%U
   logon drive = Z:
   logon home = \\%L\%U\.profiles
```

Ezekkel a beállításokkal tudjuk bekapcsolni a vándor profilt, és hogy a Samba a felhasználói könyvtárat a Z meghajtóhoz rendelje.

```
logon path =
   logon drive = z:
   #logon home =
```

Így nézne ki a konfigurációs állomány idevágó része, ha nem szeretnénk vándor profilt használni.

```
Logon script = %G.bat
```

Itt adhatjuk meg, hogy bejelentkezéskor milyen parancsfájl fusson le a kliensgépeken. A %G-vel minden csoportnak eltérő parancsfájlt adhatunk meg. Rendkívül hasznos lehetőség. Természetesen lehet csak egy parancsfájl mindenkinek, de lehet akár felhasználónként külön-külön is.

```
load printers = no
```

Hatására nem töltődnek be a nyomtatók, azaz a Samba nem működik nyomtatókiszolgálóként.

```
printing = cups
   printcap name = cups
   load printers = yes
```

Ezekkel a beállításokkal tudjuk elérni, hogy a Samba kiajánlhassa a nyomtatóinkat. Ehhez azokat majd a megosztás részben még definiálnunk kell azon kívül, hogy a CUPS[135]-ban már megfelelően bekonfiguráltnak kell lenniük, amelynek webes felületén tökéletesen tudjuk majd azokat menedzselni is.

Ezzel a globális szekció legfontosabb beállításainak végére értünk, azonban egy hasznos opciót akár már itt be lehet állítani ez pedig a veto files. A veto files után megadott fájl- és könyvtárneveket a Samba elrejti a felhasználók elől. Hasznos lehet például akkor, ha a felhasználók levelezése Maildir formátumban a saját könyvtárukban van, mert így például nem tudják azt véletlenül vagy szándékosan törölni, mivel nem is látják azt. Használata:

135. http://www.cups.org/

```
veto files = /Maildir/
```

A másik hasznos lehetőség a naplózás szabályozása.

```
log file = /var/log/samba/log.%m
   max log size = 50
```

A fenti beállításokkal netbios nevenként külön naplóz minden gépet és 50Kb-onként kezd új naplófájlt.

Megosztások

Az smb.conf másik fontos része a megosztások. A megosztások neveit [kapcsos zárójelek] közé kell tenni. Vannak kötelező megosztások, és vannak, amiket tetszőlegesen készíthetünk el.

Kötött megosztások

```
[netlogon]
```

Az itt lévő parancsfájlokat futtatják le a kliensek bejelentkezéskor.

```
[profiles]
```

Ide kerülnek a felhasználói profilok (vándor profilok esetén).

```
[homes]
```

Itt definiáljuk a felhasználók saját könyvtárait

```
[printers] [print$]
```

Az első a globális nyomtatómegosztás, helyette használhatunk nyomtatónként külön is. A másodikra akkor van szükségünk, ha azt szeretnénk, hogy a nyomtatómeghajtók kliensre telepítése automatikusan megtörténjen.[136]

Megosztásaink

A megosztások felhasználók általi használatát, amennyiben elég jól átgondoltuk a csoportok jogait, némi segítséggel könnyen szabályozhatjuk. Most a konkrét megosztásokon keresztül fogjuk végigvenni a beállítási lehetőségeket. Nézzük, hát milyen megosztásokat definiálunk:

```
[netlogon]
   browseable = no
   path = /home/samba/netlogon
   read only = yes
   write list = @admins
```

A netlogon megosztás nem tallózható (nem látszik a tallózólistában), a Linuxon lévő fizikai könyvtár helye a „/home/samba/netlogon”, csak olvasható, ugyanakkor az admins csoport tagjai (tartományi rendszergazdák csoportja) mégis írhatnak bele.

```
[profiles]
   browseable = no
   path = /home/samba/profiles
   read only = no
   create mask = 0600
   directory mask = 0700
```

136. https://wiki.samba.org/index.php/Samba_as_a_print_server

A profiles megosztás nem tallózható, fizikailag a /home/samba/profiles a helye a szerveren, csak olvasható, az itt létrejövő fájlok 600, a könyvtárak 700 joggal jönnek létre (klasszikus Linux jogosultság szerint).

```
[homes]
  valid users = %S
  comment = Felhasznalo sajat konyvtara
  browseable = no
  writable = yes
  create mask = 0700
  directory mask = 0700
```

A homes megosztásból (felhasználók saját könyvtára) minden felhasználó csak a saját „saját könyvtárát” éri el. Felhasználó saját könyvtára lesz a megjegyzés, nem tallózható, írható, és 700 joggal jönnek benne létre a fájlok és a könyvtárak is.

```
[publikalas]
  valid users = @tanar @admins @diak
  path = /home/samba/publikalas
  comment = tanarok szamara publikalasra
  browseable = yes
  create mask = 0755
  directory mask = 0755
  writable = yes
```

A publikalas könyvtárba csak a tanar, admins és diak csoport tagja léphetnek be, tallózható, 755-ös joggal jönnek létre a könyvtárak és fájlok, valamint írható. (Ha Linuxon úgy néz ki fájl szinten a megosztás könyvtára, mint itt: drwxrwxr-x root tanar publikalas, akkor máris elértük, hogy a tanár csoport tud bele írni, a diákok pedig csak olvasni tudják.)

```
[adminiszt]
  valid users = @adminisztracio verocsek vorosd
  path = /home/samba/adminisztracio
  comment = adminisztrativ dolgozoknak, zart
  browseable = yes
  create mask = 0750
  directory mask = 0750
  force group = adminisztracio
  writable = yes
```

Az adminiszt megosztást elérhetik az adminisztracio csoportba tartozó felhasználók, de rajtuk kívül a verocsek és a vorosd nevű felhasználó is. Tallózható, 750-es joggal jönnek létre benne az objektumok, és minden fájl és könyvtár az adminisztracio csoport nevében jön létre. Írható megosztás. A force group-pal érjük el, hogy az egyébként tanár verocsek és vorosd felhasználók által létrehozott fájlok, könyvtárak olvashatóak legyenek az adminisztracio csoport tagjai számára, mivel minden általuk létrehozott fájl és könyvtár tulajdonoscsoportja az adminisztracio csoport lesz, hiába tartoznak ők a tanárok közé valójában. (A könyvtár Linux alól így néz ki: drwxrwx--- root adminisztracio adminisztracio.)

```
[adminiszt2]
  valid users = @adminisztracio @tanar
  path = /home/samba/adminisztrac2
  comment = adminisztrativ dolgozoknak, nyilt
  browseable = yes
  create mask = 0750
```

```
  directory mask = 0750
  writable = yes
  force group = adminisztracio
```

Az adminiszt2 megosztáshoz az adminisztracio és a tanar csoport tagjai csatlakozhatnak csak. Függetlenül attól, hogy tanár vagy adminisztratív dolgozó hozott létre valamit, az 750-es joggal az adminisztratív dolgozók csoport tulajdonában lesz. Így elértük, hogy a két külön csoport tudja olvasni egymás elmentett fájljait, írni pedig mindenki csak a sajátját. (Természetesen Linuxon az adminisztracio csoportnak rwx joga kell legyen a könyvtáron.)

```
[hr]
  valid users = balazs petra
  write list = balazs petra
  path = /home/samba/hr
  comment = hr-es dolgozoknak
  browseable = yes
  create mask = 0770
  directory mask = 0770
  writable = yes
  force group = adminisztracio
```

A hr megosztásba már csak 2 felhasználó léphet be, nem pedig az egész adminisztracio csoport. Ők ketten a balazs és a petra felhasználók. Minden, amit létrehoznak, az adminisztracio csoport jogával jön létre, ráadásul itt 770 a jog, tehát képesek egymását törölni is, nem csak olvasni.

```
[rendszerg]
  path = /home/samba/rendszergazda
  comment = rendszergazdai publikaciok
  browseable = yes
  create mask = 0755
  directory mask = 0755
  read list = @tanar @diakok @adminisztracio
  write list = @admins
  writable = yes
```

A rendszerg megosztáshoz hozzáférnek a diákok, a tanárok és az adminisztratív dolgozók is, azonban csak olvasási joggal, írási joga egyedül a rendszergazdák csoportnak van. (Linux alatt valami hasonlónak kell lennie a könyvtárnak: drwxrwxr-x administrator admins rendszergazda)

A megosztásoknál lehet tovább cifrázni, ha a fájlrendszerünk támogatja az ACL-eket, illetve néha jól jöhet az immutable vagy az append bit[137] beállítása is egy-egy könyvtáron.

További hasznos lehetőség lehet, ha szeretnénk lomtárakat létrehozni a hálózati megosztásokon, ezt megosztásonként lehet az alábbi példa szerint:

```
vfs objects = recycle
  recycle:repository = /home/samba/adminisztracio/kuka/%U
```

Példa nyomtatók megosztására

```
[anyomiga]
     comment = Kozpotni nyomtato adminisztrativ dolgozoknak igazgatasi csoport
     valid users = @adminisztracio @admins muszashi tami
```

137. http://en.wikipedia.org/wiki/Chattr

```
    path = /var/spool/samba
    printer = anyomiga
    writable = no
    printable = yes
    use client driver = yes
```

A szokásos értelmezés alapján a nyomtatóra csak a rendszergazda csoportba és az adminisztracio csoportba tartozók nyomtathatnak, rajtuk kívül még a muszashi és a tami felhasználó. Vegyük észre, hogy ez nem írható, ellenben nyomtatható megosztás. Ezen elv alapján tudunk további nyomtatókat is felvenni a rendszerbe. **Fontos, hogy a printer= után megadott név pontosan egyezzen a CUPS-ban konfigurált névvel! Az use client driver abban az esetben kell, ha kliens oldalon telepítjük a nyomtatók meghajtóprogramjait.** Nélküle csak nulla méretű fájlok sorakoznak majd a /var/spool/samba könyvtárban. Ha azt szeretnénk, hogy a szerverről automatikusan települjenek a meghajtóprogramok, akkor további lépéseket kell tennünk.[138]

A netlogon könyvtár szkriptjei

A netlogon könyvtárban elhelyezett szkriptek lefutnak a klienseken történő belépéskor. Jelenlegi beállítással minden csoportnak külön szkriptje van, ami azt jelenti, hogy a szkript neve megegyezik a felhasználói csoport nevével. Jelen esetben adminisztracio.bat, tanar.bat, diak.bat. Érdemes ezeket a fájlokat Windowson szerkeszteni, mivel a linuxos sorvégjel nem jó a windowsos klienseknek. Annak érdekében, hogy lássuk mikre lehet használni, nézzünk megint egy példát, a diak.bat tartalmát:

```
echo off
  echo "Csatlakozas a szerverhez..."
  echo "Mappolom a home konyvtarat a Z: meghajtohoz"
  net use z: /HOME /yes
  start \\szerver3\netlogon\shortcut.vbs
  echo "Ora szinkronizalasa a szerverhez"
  net time \\szerver3 /set /yes
  echo "Tovabbi konyvtarak rendszerhez adasa..."
  net use k: \\szerver3\publikalas /yes
  start \\szerver3\netlogon\shortcut5.vbs
  net use r: \\szerver3\rendszerg /yes
  start \\szerver3\netlogon\shortcut4.vbs
  echo "...tovabbi parancsikonok elhelyezese"
  start \\szerver3\netlogon\shortcut9.vbs
  start \\szerver3\netlogon\shortcut10.vbs
```

... és példaképp az egyik hivatkozott vbs fájlt is (szintén a netlogon könyvtárban van):

```
' VBScript.
  Dim Shell, DesktopPath, URL
  Set Shell = CreateObject("WScript.Shell")
  DesktopPath = Shell.SpecialFolders("Desktop")
  Set URL = Shell.CreateShortcut(DesktopPath & "\Rendszergazda.LNK")
  URL.TargetPath = "R:\"
  URL.Save
```

Látható, ez például kiteszi az asztalra a rendszergazda megosztás parancsikonját, amelyet a szkriptben R: meghajtóként felcsatoltunk. Ezzel könnyíthetjük meg felhasználóink és a saját életünket is.

138. https://wiki.samba.org/index.php/Samba_as_a_print_server

Felhasználók létrehozása

Először adjuk hozzá az administrator felhasználót a rendszerhez az alábbi parancsokkal:

```
useradd -m -s /bin/false -c "Samba Admin" -G admins administrator
smbpasswd -a administrator
```

A felhasználókat létrehozhatjuk egyesével is, de érdemes valamilyen szkriptet készíteni amivel könnyebbé tehetjük a munkát. Itt most láthatjuk egy régi rendszer felhasználólétrehozó szkriptjét átalakítva, amiben eredetileg volt e-mail fiók, csoportmunka hozzáférés létrehozása és kvóta is, de most csak a lecsupaszított Samba hozzáférések létrehozását csinálja. Arra mindenképpen jó, hogy ez alapján lehessen sajátot készíteni.

```
#!/bin/bash
  clear
  echo " ............ Felhasználó létrehozó script! ............"
  # felhasznaloi csoport konstansok
  export CSOPORT1="diak"
  export CSOPORT2="tanar"
  export CSOPORT3="adminisztracio"
  # Kvota konstansok
  export CSOP1K="300000"
  export CSOP2K="1024000"
  export CSOP3K="2048000"
  # felhasznaloi alapadatok
  export VNEV=""
  export KNEV=""
  export UNEV=""
  export KORNYEZETE="/bin/false"
  export PASSWORDJE=""
  export CSOPORTJA=""
  # Adatok bekérése:
  echo "Kérem a felhasználó vezetéknevét: "
  read VNEV
  echo " "
  echo "Kérem a felhasználó keresztnevét: "
  read KNEV
  echo " "
  echo "Kérem a felhasználó user-nevét: "
  read UNEV
  cat /etc/passwd | grep $UNEV
  PROBA=$?
  case "$PROBA" in
  1):
      echo "OK!";;
  0):
      echo "Sajnálom $USERNEV felhasználó MÁR létezik!!!";
      exit 1;;
  esac
  echo " "
  echo "Kérem a felhasználó jelszavát: "
  read PASSWORDJE
  echo " "
  echo " "
```

```
  echo "Neve: " $VNEV" "$KNEV "Userneve: " $UNEV  " jelszava: " $PASSWORDJE
  echo " "
  echo " "
  echo "Mely csoporthoz kíván felhasználót hozzáadni:"
  echo "1," $CSOPORT1
  echo "2," $CSOPORT2
  echo "3," $CSOPORT3
  echo ""
  read valasztas
  case "$valasztas" in
  1):
      export CSOPORTJA=$CSOPORT1;
      useradd -m -p $PASSWORDJE  -s $KORNYEZETE -c $UNEV -g $CSOPORTJA $UNEV;
      echo $UNEV:$PASSWORDJE|chpasswd -md5;
      (echo $PASSWORDJE; echo $PASSWORDJE)|smbpasswd -s -a $UNEV;;
  2):
      export CSOPORTJA=$CSOPORT2;
      useradd -m -p $PASSWORDJE -s $KORNYEZETE -c $UNEV -g $CSOPORTJA $UNEV;
      echo $UNEV:$PASSWORDJE|chpasswd -md5;
      (echo $PASSWORDJE; echo $PASSWORDJE)|smbpasswd -s -a $UNEV;;
  3):
      export CSOPORTJA=$CSOPORT3;
      useradd -m -p $PASSWORDJE -s $KORNYEZETE -c $UNEV -g $CSOPORTJA $UNEV;
      echo $UNEV:$PASSWORDJE|chpasswd -md5;
      (echo $PASSWORDJE; echo $PASSWORDJE)|smbpasswd -s -a $UNEV;;
  esac
  echo ""
  echo "A kiválasztott csoport: $CSOPORTJA"
  echo " "
  echo "A művelet végetért... "
```

A teljes /etc/samba/smb.conf

Itt pedig egyben látható az elkészült konfigurációs állományunk, az /etc/samba/smb.conf tartalma:

```
[global]
  interfaces = eth1 127.0.0.1/8
  bind interfaces only = yes
  hosts allow = 127.0.0.1/8 10.0.0.
  hosts deny = 0.0.0.0/0
  admin users = administrator
  unix password sync = yes
  pam password change = yes
  netbios name = SZERVER3
  server string = KOZPONTI_LINUX_SZERVER
  workgroup = LOK
  passdb backend = smbpasswd
  os level = 254
  preferred master = yes
  domain master = yes
  local master = yes
  security = user
  domain logons = yes
```

```
  encrypt passwords = true
  wins support = yes
  name resolve order = wins bcast host lmhosts
  time server = yes
  display charset = UTF8
  unix charset = UTF8
  case sensitive = no
  default case = lower
  preserve case = yes
  short preserve case = yes
  add machine script = /usr/sbin/useradd -d /dev/null -g 100 -s /bin/false %u
  logon path = \\%L\profiles\%U
  logon drive = Z:
  logon home = \\%L\%U\.profiles
  Logon script = %G.bat
  printing = cups
  printcap name = cups
  load printers = yes
  log file = /var/log/samba/log.%m
  max log size = 50

  [netlogon]
  browseable = no
  path = /home/samba/netlogon
  read only = yes
  write list = @admins

  [profiles]
  browseable = no
  path = /home/samba/profiles
  read only = no
  create mask = 0600
  directory mask = 0700

  [homes]
  valid users = %S
  comment = Felhasznalo sajat konyvtara
  browseable = no
  writable = yes
  create mask = 0700
  directory mask = 0700

  [publikalas]
  valid users = @tanar @admins @diak
  path = /home/samba/publikalas
  comment = tanarok szamara publikalasra
  browseable = yes
  create mask = 0755
  directory mask = 0755
  writable = yes
[adminiszt]
  valid users = @adminisztracio verocsek vorosd
  path = /home/samba/adminisztracio
```

```
comment = adminisztrativ dolgozoknak, zart
browseable = yes
create mask = 0750
directory mask = 0750
force group = adminisztracio
writable = yes

[adminiszt2]
valid users = @adminisztracio @tanar
path = /home/samba/adminisztrac2
comment = adminisztrativ dolgozoknak, nyilt
browseable = yes
create mask = 0750
directory mask = 0750
writable = yes
force group = adminisztracio

[hr]
valid users = balazs petra
write list = balazs petra
path = /home/samba/hr
comment = hr-es dolgozoknak
browseable = yes
create mask = 0770
directory mask = 0770
writable = yes
force group = adminisztracio

[rendszerg]
path = /home/samba/rendszergazda
comment = rendszergazdai publikaciok
browseable = yes
create mask = 0755
directory mask = 0755
read list = @tanar @diakok @adminisztracio
write list = @admins
writable = yes

[Deskjet_3510]
   comment = Kozpotni nyomtato adminisztrativ dolgozoknak igazgatasi csoport
   valid users = @adminisztracio @admins muszashi tami
   path = /var/spool/samba/
   printer name = Deskjet_3510
   writable = no
   printable = yes
   use client driver = yes
```

A konfigurációs állományunk helyességét ellenőrizhetjük a szerveren a testparm parancs kiadásával.

Samba használata

Most tehát már elkészült a konfigurációs állományunk, felvettük az administrator és az egyéb felhasználókat, így nincs más teendő, mint elindítani a Sambát.

Samba indítása

```
service samba start
```

vagy a régi módon az

```
/etc/init.d/samba start
```

segítségével tudjuk megtenni.

Kliensek tartományba léptetése

A windowsos gépeken jelentkezzünk be rendszergazda jogú felhasználóval, hogy a gép tartományba léptetését el tudjuk végezni. XP esetén nincs extra teendő, ugyanakkor Windows 7 alatt szükséges az alábbi registry beimportálása:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManWorkstation\Parameters]
"DNSNameResolutionRequired"=dword:00000000
"DomainCompatibilityMode"=dword:00000001
```

A tartományba léptetést a Vezérlőpult\Rendszer és Biztonság\Rendszer útvonalon a számítógépnév módosítása alatt tudjuk megtenni. A tagságot munkacsoportról tartományra kell állítani és be kell írni a tartomány nevét (LOK). Következő lépésként egy felugró ablakban meg kell adni a *tartományi rendszergazda* (administrator) nevét és jelszavát. Ha mindent jól csináltunk, sikeresen be kell tudni lépjünk a tartományba.

Hasznos segédeszközök

Pár eszközt érdemes lehet telepíteni a szerveren. Ezek a cifs-utils csomagban találhatóak. Telepítsük őket a szokásos módon!

```
apt-get install cif-utils
```

Hasznos segédeszközeink egy Samba szerveren:

- **smbclient**
  Sokcélú programocska, például a Samba elindítása után rögtön ellenőrizhetjük vele, hogy működik-e: smbclient -L localhost -U administrator
- **nmblookup**
  Névfeloldás ellenőrzésére tökéletes, például így: nmblookup -U szerver IP -A egy kliens IP
- **smbtree**
  Megjeleníti a hálózaton lévő összes aktív Netbios nevet
- **smbstatus**
  A legfontosabb eszköz: láthatjuk, mely felhasználók, mely gépeken, milyen megosztásokon mit csinálnak éppen.

Samba 4 telepítése, konfigurálása

Ebben a részben telepítünk egy Samba 4-es tartományt, gyakorlatilag ugyanazt a rendszert valósítjuk meg a négyes Samba segítségével, amit az előző oldalakon a 3.6-os ismertetésénél már egyszer létrehoztunk. Ami onnan már ismerős lehet, azt itt már csak megemlítjük, részletesen nem tárgyaljuk. Főként az eltéréseken lesz ebben a részben a hangsúly. *Mivel a Samba 4 még folyamatosan változik, így az itt leírt dolgok csak ebben a verzióban működnek biztosan!*

Samba 4 igényei

A Samba 4 az Active Directory miatt rengeteg komponens meglétét igényi, jóval összetettebb, mint a korábbi sorozatok. A következőkre van szüksége a megfelelő működésre:

– DNS szerver
– Kerberos
– NTP
– LDAP
– ACL és Extended Attributes

A fentieken kívül javasolt még a quota támogatás, és egy CUPS szerver telepítése is (nyomtatáshoz).

Ami szerencse, hogy a Samba 4 tartalmaz LDAP-ot, van belső DNS szervere is, Kerberos-t sem kell telepítenünk. Természetesen lehet külső eszközöket használni, például BIND lehet a DNS szerver, azonban egy alap telepítéshez tökéletesen jól használhatóak a Samba beépített megoldásai. NTP-szervert ellenben telepítenünk kell majd, mivel pontos idő nélkül a Kerberos nem működne helyesen (egészen pontosan arra van szükségünk, hogy a kliensek és a szerver órája járjon együtt, az lényegtelen ez miként viszonyul a valós időhöz). A fájlrendszer ACL és XATTR támogatására szükségünk van, mert ezen keresztül tudnak tárolódni a windowsos jogosultságok, CUPS-ot is telepítenünk kell, mivel nyomtatni szeretnénk és természetesen a quota is fontos lehet, ezért majd azt is beállítjuk az fstab fájlban.

Samba 4 beszerzése

A Samba 4 beszerzése rögtön kihívásokkal teli, mivel az általunk használt stabil Debian verzióban a 3.6 az alapértelmezett és ajánlott, 4-es sorozatból csak 4.0.0 beta van benne, amit még a készítők sem ajánlanak produktív környezetbe. Jelenleg 2 módon lehet hozzájutni:

– forrásból telepítve
– enterprisesamba oldalról letöltve a debian csomagokat

Mi az utóbbi megoldást választjuk, és a következő lépéseket követjük:

– Először el kell látogatni a **http://enterprisesamba.com/** oldalra, és ott regisztrálni szükséges. A regisztráció és a letöltés ingyenes. Miután regisztráltunk, kapunk egy jelszót, amivel majd le tudjuk tölteni a csomagokat.
– Állítsuk be az /etc/apt/sources.list fájlban, hogy az enterprisesamba oldal is a források között legyen, ezt a következő sor beszúrásával tudjuk megtenni:

```
deb https://USERNAME:ACCESSKEY@download.sernet.de/packages/samba/4.0/debian
```

```
  wheezy main
```

A USERNAME helyére a felhasználói nevet, az ACCESSKEY helyére a kapott jelszót írjuk be.

– Töltsük le és telepítsük a kulcsokat az alábbi parancsok begépelésével!

```
wget http://ftp.sernet.de/pub/sernet-samba-keyring_1.4_all.deb
  dpkg -i sernet-samba-keyring_1.4_all.deb
```

– Megfigyelhetjük, hogy a telepítési források https-en keresztül érhetőek el. Ahhoz, hogy az apt kezelni tudja ezt, előbb telepíteni kell az apt-transport-https csomagot. Tegyük meg:

```
apt-get install apt-transport-https
```

– Frissítsük a csomaglistát!

```
apt-get update
```

– Telepítsük a Sambát!

```
apt-get install sernet-samba-ad
```

– Amennyiben a Samba sikeresen települt, azt ellenőrizzük le a verzió lekérésével:

```
samba -V
```

Jó esetben, valami ilyet kapunk válaszul: *Version 4.0.10-SerNet-Debian-8.wheezy*. Tehát jelenleg a 4.0.10 a legfrissebb amihez hozzájuthattunk.

További előkészítő munkálatok

Mint feljebb már volt róla szó, szükségünk van egy NTP-szerverre. Ezt könnyedén telepíthetjük az alábbi parancs begépelésével:

```
apt-get install ntp
```

Ha szeretnénk (nem kötelező), akkor utána a pool-kat módosíthatjuk az /etc/ntp fájlban hu nevűekre így:

```
server 0.hu.pool.ntp.org
server 1.hu.pool.ntp.org
server 2.hu.pool.ntp.org
server 3.hu.pool.ntp.org
```

A helyes működést ellenőrizhetjük az alábbi paranccsal:

```
ntpq -p
```

Engedélyeznünk kell a Samba számára fenntartott partíciókon az ACL és Extended Attributes, illetve a quota támogatást is. Ehhez előbb telepíteni kell ezeket az alábbi paranccsal:

```
apt-get install acl attr
```

Ezután szerkesszük az /etc/fstab állományt az alábbi mintához hasonlóan:

```
/dev/sda3 /home ext4
  defaults,nodev,nosuid,usrquota,grpquota,user_xattr,acl,barrier=1      0
  2
```

Ebből az acl és a user_xattr megléte most a lényeges számunkra. A barrier=1 a Samba csapat ajánlása, mely szerint nélküle hirtelen áramszünetkor előfordult pár adatbázissérülés, szóval inkább tegyük be mi is.

Ellenőrizzük, hogy a hálózati beállítások rendben legyenek. Ez alatt az értendő, hogy az /etc/resolv.conf állományban legyen valamilyen működő DNS beállítva (ezt az értéket fogja majd a Samba bemásolni az /etc/samba/smb.conf állományba), és tanácsos, ha a gép hostneve megegyezik azzal a névvel, ami a tartomány neve lesz, mert az eltérő hostnév esetenként problémát okozhat.

Ha mindezekkel megvagyunk indulhat a Samba konfigurálása!

Konfigurálás

A samba-tool nevezetű programot kell használnunk az alapbeállítások elkészítéséhez. A parancs használható interaktív módon is (samba-tool domain provision –interactive), illetve úgy is hogy rögtön megadjuk neki a kívánt értékeket. Íme:

```
samba-tool domain provision --realm=lok.local --domain=lok --server-role=dc
  --adminpass=Tuj024b
```

Látható, hogy meg kell adni a teljes tartománynevet (lok.local), a tartománynevet (lok), a szerver szerepkörét (dc, azaz domain controller, tartományvezérlő) és az admin felhasználó jelszavát. Ez utóbbira külön figyeljünk, mert csak kellően bonyolult jelszót fogad el, kis- és nagybetű, valamint szám kombinációt. Gyenge jelszó esetén a folyamat, amely a parancs indítása után fog kezdődni, a vége felé hibával megszakad.

Ezzel el is készült a tartományvezérlőnk, létrejön egy alap smb.conf az alábbi tartalommal:

```
[global]
        workgroup = LOK
        realm = lok.local
        netbios name = SZERVER4
        server role = active directory domain controller
        dns forwarder = 8.8.4.4
  [netlogon]
        path = /var/lib/samba/sysvol/lok.local/scripts
        read only = No
  [sysvol]
        path = /var/lib/samba/sysvol
        read only = No
```

Láthatunk itt egy új megosztást sysvol néven: ez a csoportházirendekhez kell, nélküle nem lenne AD-vezérlő a Sambánk, illetve látható egy dns forwarder bejegyzés is, amely arra a DNS-szerverre mutat, ami telepítéskor be volt állítva az /etc/resolv.conf-ban, és felé irányítja a külső lekérdezéseket.

Utómunkálatként töröljük ki az összes futási szint könyvtárából (/etc/rc könyvtárak) az összes sernet-samba-nmbd, sernet-samba-smbd, sernet-samba-winbindd-re mutató linket. A sernet-samba-ad-t viszont feltétlen hagyjuk meg, mivel ezzel tudjuk indítani, leállítani a Samba 4 szerverünket.

Szerkesszük az /etc/default/ könyvtárban a **sernet-samba** állományt úgy, hogy a *samba_start_mode=”none”* helyett *„ad”* legyen:

```
SAMBA_START_MODE="ad"
```

Eljött az indítást, tesztelés ideje.

```
/etc/init.d/sernet-samba-ad restart
```

Újraindítjuk a Samba szervert.

```
ps ax
```

A paranccsal meggyőződhetünk, hogy fut-e (ilyen sorokat kell látnunk a kimenetben /usr/sbin/samba -D).

```
smbclient -L 127.0.0.1 -U%
```

A parancs kiadására látnunk kell, hogy mi a tartomány neve, milyen megosztások vannak stb. Ha a Samba jól fut, teszteljük le még a DNS-szolgáltatását is.

```
host -t SRV _ldap._tcp.lok.local. 10.0.0.1
```

A 10.0.0.1 a saját szerverünk IP címe, hiszen tőle várunk eredményt a lekérdezésre.

```
host fsf.hu 10.0.0.1
```

...és végül ellenőriztük, hogy külső címet is helyesen old fel.

Fontos, hogy a klienseken a Samba szerver címe legyen majd megadva DNS-szervernek! Ha minden jól működött, akkor tényleg elkészült a tartományvezérlő, mostantól be lehet léptetni a tartományba a gépeket.

További globális beállítások

Érdemes azonban még további beállításokkal bővíteni a [global] szekciót. Hogy mik az alapértelmezett beállítások, azt a hármas sorozattal ellentétben itt a samba-tool testparm -v paranccsal kapjuk meg.

Ezek a beállítások már a hármas Samba ismertetésekor említésre kerültek, így tudjuk szabályozni, hogy csak a belső interfészünkön kommunikáljon a Samba:

```
interfaces = eth1 lo
  bind interfaces only = yes
```

Az előző verziónál ismertetett host allow, host deny-t ne kapcsoljuk be, mert akkor látszólag működő, tallózható szerverünk lesz, ellenben mikor szeretnénk majd menedzselni windowsos kliensről a gépet (hamarosan), akkor nem fogja elérni a DC-t a megfelelő segédprogram.

Gyakorlatilag ennyi elég is, mivel sok minden egész másképp zajlik a 4-es sorozatban, mint az elődben, ezt már rögtön a következő részben is láthatjuk.

Megosztások

A legfontosabb, hogy már nem kell a linuxos könyvtárszerkezetben beállítani semmilyen különös jogosultságot (csoportnak stb.), egyszerűen legyenek a könyvtáraink a root számára írhatóak.

Hozzuk először létre az alapvető működéshez szükséges megosztásokat:

```
[home]
        path = /home/samba/home
        read only = No

  [Profiles]
        path = /home/samba/Profiles/
        writable = yes
```

Majd az összes többi megosztásunkat:

```
[publikalas]
        path = /home/samba/publikalas
        comment = tanarok szamara publikalasra
        browseable = yes
        writable = yes
  [adminiszt]
        path = /home/samba/adminisztracio
        comment = adminisztrativ dolgozoknak, zart
        browseable = yes
        writable = yes
  [adminiszt2]
        path = /home/samba/adminisztrac2
        comment = adminisztrativ dolgozoknak, nyilt
        browseable = yes
        writable = yes
  [hr]
        path = /home/samba/hr
        comment = hr-es dolgozoknak
        browseable = yes
        writable = yes
  [rendszerg]
        path = /home/samba/rendszergazda
        comment = rendszergazdai publikaciok
        browseable = yes
        writable = yes
  [Deskjet_3510]
        comment = Kozpotni nyomtato adminisztrativ dolgozoknak igazgatasi
  csoport
        path = /var/spool/samba/
        printer name = Deskjet_3510
        writable = no
        printable = yes
```

Látható, jócskán csökkent egy-egy megosztásnál a tartalom, mivel a valid user és force group amikkel eddig tudtuk kordában tartani a megosztásokhoz való csatlakozást, feleslegessé vált, helyette az Active Directory-ban fogjuk megmondani, ki mihez férhet hozzá. (Működnek, elvben használhatóak, csak immár feleslegesek.)

Hozzuk létre a fájlrendszerben is a könyvtárakat, amelyekre a megosztások mutatnak, egyedül a nyomtató megosztásnál figyeljünk rá, hogy a csoportnak és mindenki másnak is teljes hozzáférési joga legyen a könyvtárhoz, azaz azt így hozzuk létre:

```
mkdir -p /var/spool/samba/
chmod 1777 /var/spool/samba/
```

A Samba alapból támogatja a CUPS-ot, tehát nem kell erre semmilyen konfig sort beírjunk a global szekcióban.

Tartomány kezelése

Eljött az idő, hogy a tartományi rendszergazda nevében bejelentkezzünk egy windowsos gépen, és a hármas Sambánál leírt módon beléptessük a tartományba a gépet (leszámítva, hogy *már a registry kulcsot nem kell módosítsunk előtte)*. A tartományi rendszergazda neve: **administrator**, a jelszava pedig az, amit megadtunk a tartomány létrehozásakor.

Rsat letöltése, telepítése

Első dolgunk a beléptetés után, hogy letöltsük a Windows Remote Server Administration Tools programot.

Windows 7-re: http://www.microsoft.com/hu-hu/download/details.aspx?id=7887

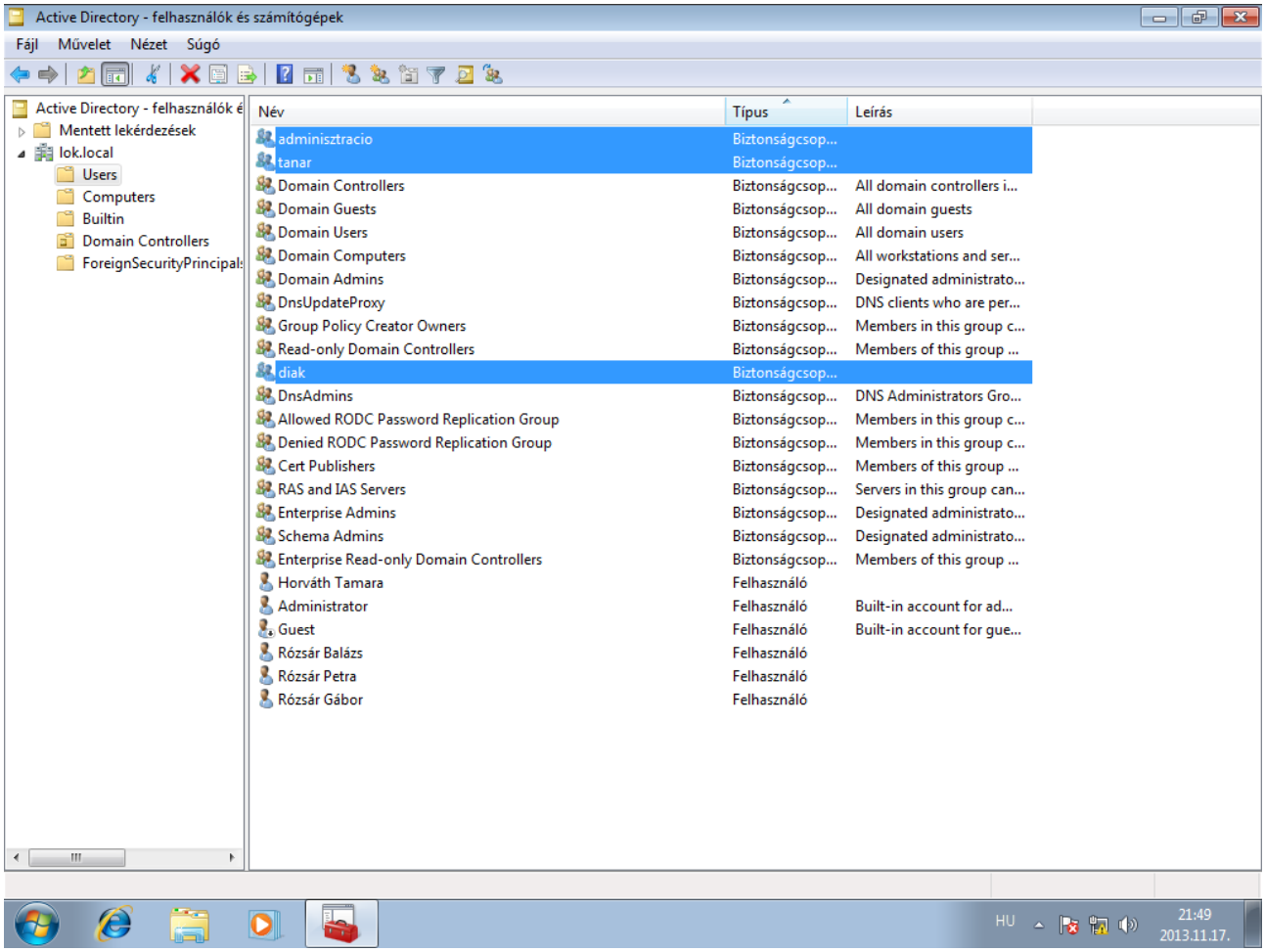
Windows 8-ra: http://www.microsoft.com/hu-hu/download/details.aspx?id=28972

Akit pedig eltérő verziók érdekelnek, itt nézhet utána: https://wiki.samba.org/index.php/Samba_AD_management_from_windows

Miután telepítettük, látszólag nincsenek új programok a gépen, ugyanis előtte még a ***Vezérlőpult > Programok > Programok és szolgáltatások*** alatt a Windows-szolgáltatások be- és kikapcsolása pontban engedélyezni kell a távoli kiszolgálófelügyeletet. Ezek után a Felügyeleti eszközök alá bekerül sok újabb lehetőség, amikkel az AD-t tudjuk elérni, vagy egyszerűen futtassuk:

```
dsa.msc
```

Ennek hatására elindul az AD felügyeletét ellátó segédprogram (Active Directory – felhasználók és számítógépek):

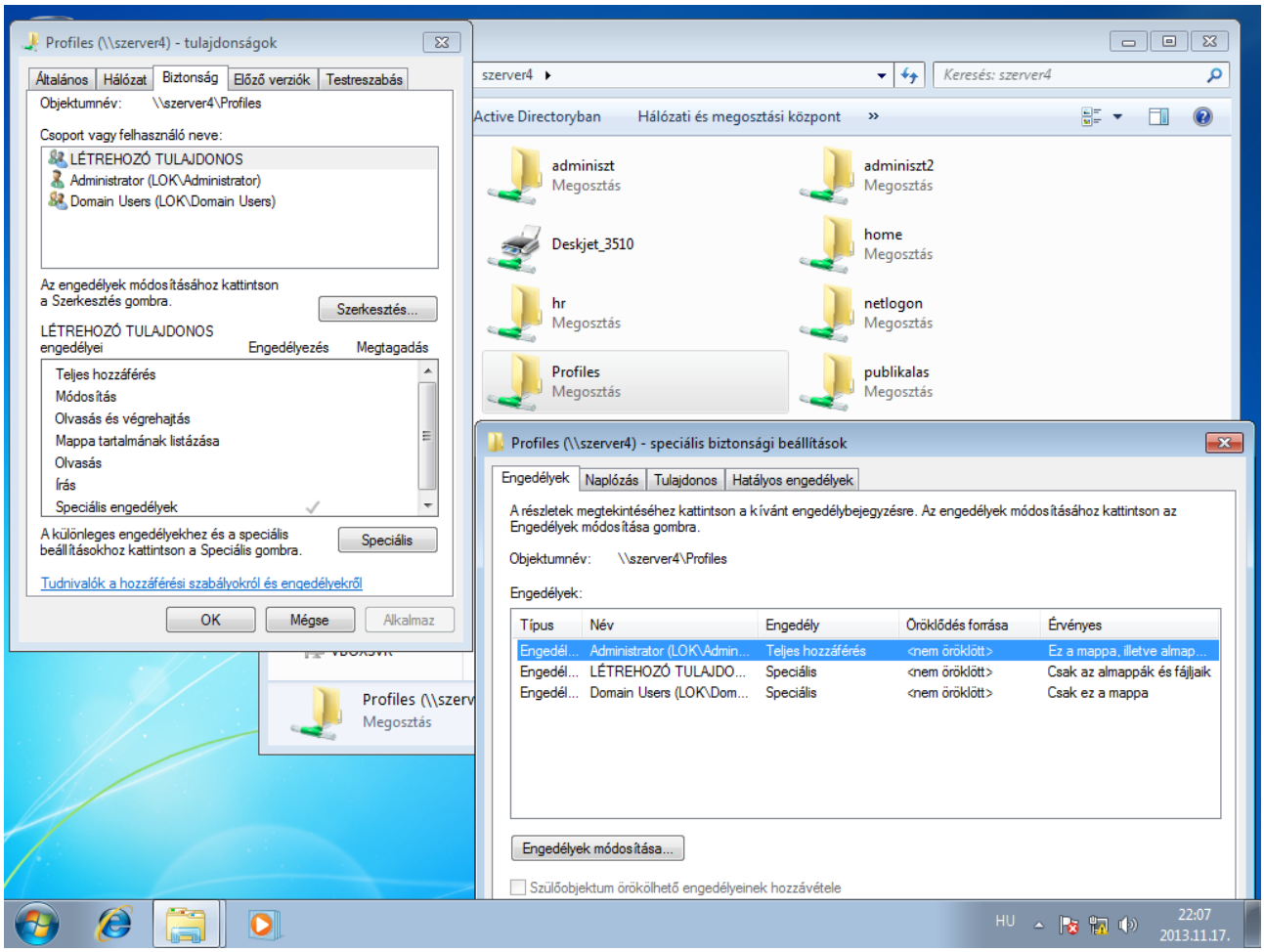


A program használata egyszerű, gyakorlatilag kattintgatással összeállítható a kívánt konfiguráció. Javasolt először létrehozni a csoportokat, azaz tanár, diak, adminisztracio, miként azt a fenti ábra is mutatja (a jobb egér gomb hatására előugró helyi menüből tudunk választani).

Megosztások jogai

Következő lépésként érdemes a megosztások megfelelő jogait beállítani. Erre legjobb elindítani az intézőt, majd az UNC szintaxisnak megfelelően a beviteli sorba beírni a szerver nevét. Jelen esetben így: \\szerver4.

Majd jobb egérgombbal kattintsunk a kívánt mappán, és a biztonság fülön kattintsunk a speciális gombra. Valahogy így:



...majd a megosztott könyvtáraink számára állítsuk be a kívánt jogokat!

Home és Profiles mappák jogai

A home és a profiles mappák számára a következő jogokat állítsuk be:

Profiles mappa:

– administaror (teljes hozzáférés, ez a mappa, almappa és fájljai)

– domain users (mappa bejárás, fájl végrehajtás, mappa listázása, adatok olvasása, mappák létrehozása, adathozzáfűzés, csak ez a könyvtár)

– létrehozó tulajdonos (teljes hozzáférés, alkönyvtár és a fájljai

Ennek a példáját látjuk a fent bemutatott képen is.

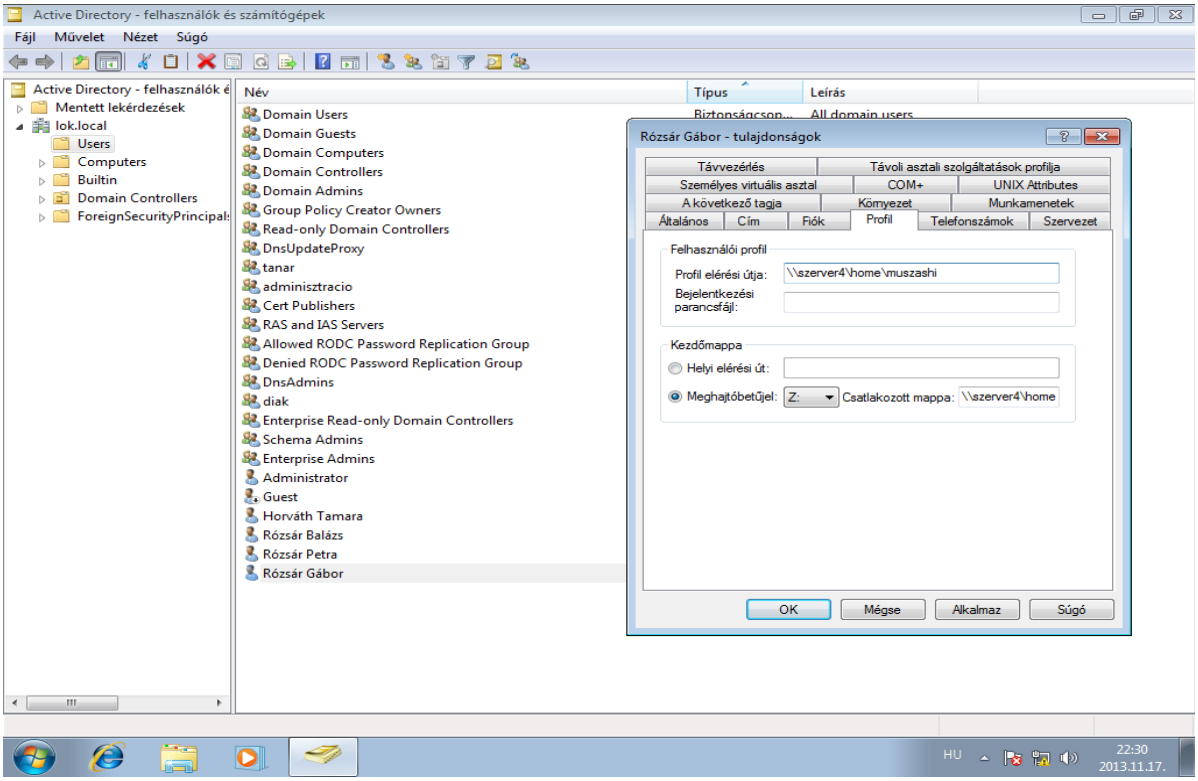
Home mappa:

– administrator (teljes hozzáférés)
– system (teljes hozzáférés)
– hitelesített felhasználók (teljes hozzáférés – itt érdemes a hatókört a **csak ez a mappára** csökkenteni, így kitallózva nem tudnak egymás könyvtárába benézegetni a felhasználók)

Home és Profil könyvtárak beállítása

Ismét indítsuk el a dsa.msc-t, és miután kiválasztottuk a felhasználót, kattintsunk a tulajdonságokra, majd a profil fülre. Válasszuk ki, mely meghajtóhoz rendelje hozzá a **home** könyvtárat, majd a csatlakoztatott mappához írjuk be az útvonalat ***\\szerverneve\home\felhasználói név*** formában. Az OK gombra kattintva elkészíti az adott felhasználó könyvtárát.

Következő lépés a **Profil** beállítása, amit szintén itt tudunk megtenni a Profil elérési útjához beírt ***\\szerverneve\Profiles\felhasználói név*** formában.



Nyomtatás beállítása

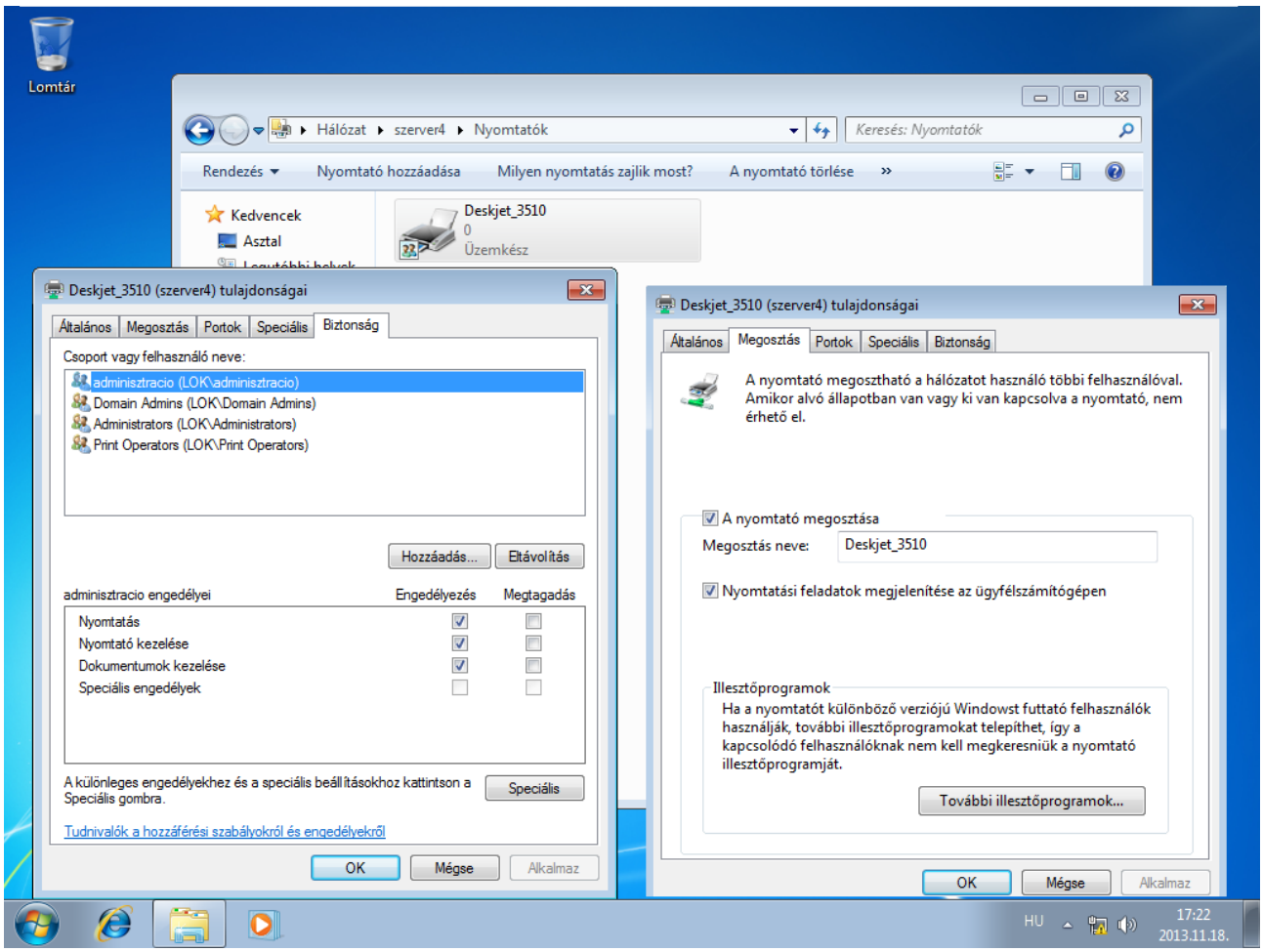
Következő lépésként a nyomtatókonfiguráció van hátra, ahol most maradunk a kliens oldalról történő meghajtótelepítésnél. Aki ennél többre vágyik, az itt talál információkat az automatikus meghajtótelepítésről: https://wiki.samba.org/index.php/Samba_as_a_print_server#General

Nekünk csak egy dolgunk van, a hálózati nyomtató használatának korlátozása az adminisztracio csoportra, hogy rajtuk kívül más ne tudjon nyomtatni rá. Feltételezzük, hogy a CUPS-ban a ***megosztás nevével megegyező nevű nyomtató*** telepítésre került, és megfelelően működik.

– Tartományi rendszergazdaként lépjünk be egy kliensen, majd az intézőben navigáljunk el a szerverünkhöz az UNC formátumú név beírásával, jelen esetben \\szerver4.
– Kattintsunk a távoli nyomtatók megtekintése pontra, és kattintsunk a nyomtató tulajdonságaira.

– Válasszuk a megosztást, adjunk nevet a megosztásnak.

– A biztonság fülön jelöljük be, mely csoportok és felhasználók érhetik el a nyomtatót.



Logon szkriptek

Az utolsó lépés, hogy miként lehet a Samba 4 alatt a logon szkriptjeinket futtatni. Ami azonos, hogy a netlogon megosztásban kell őket elhelyezni, és ügyelni kell rá, hogy legyen olvasás és futtatás jog rajta a hitelesített felhasználóknak.

Két módon tudjuk elérni, hogy lefusson egy logon szkript.

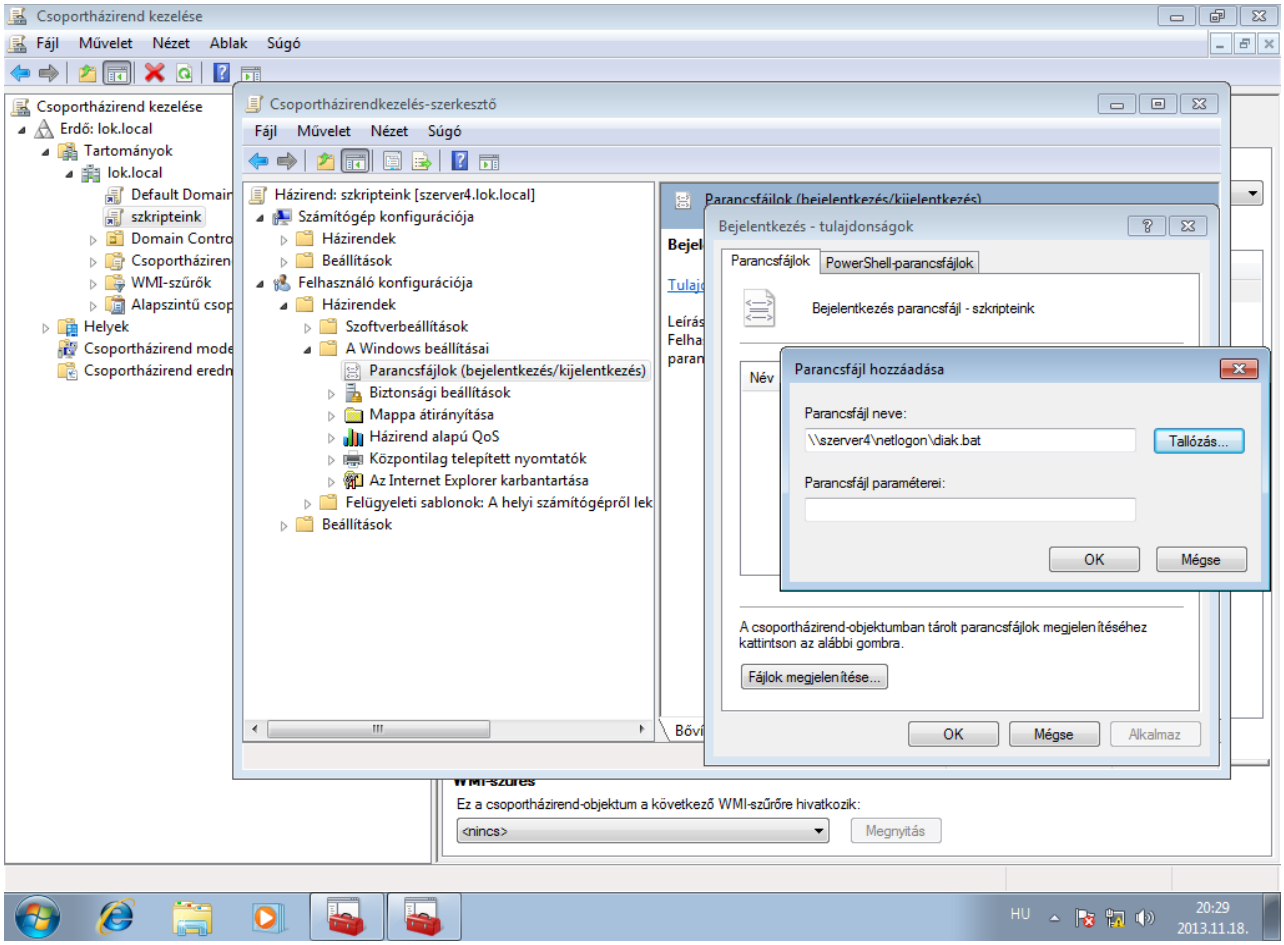
– Az egyik esetben a már ismerős Active Directory – felhasználók és számítógépek elindításával, az adott felhasználóknál a profil fülön kell beírni a logon szkript nevét. Figyeljünk rá, hogy amíg a home és profiles-nél a teljes UNC útvonalat meg kellett adni, addig itt csak a tényleges fájlnevet kell és szabad, egyéb esetben nem fog lefutni! (pl. diak.bat)

– A másik lehetőség egy globális megoldás, ehhez az RSAT részeként települő csoportházirend kezelése eszköz, melyet vagy kibogarászunk a Start menüből, vagy írjuk be:

```
gpmc.msc
```

A feljövő ablakban válaszuk a **tartományunkat,** majd a jobb egér gombbal való kattintás után válasszuk a **csoportházirend objektum létrehozását**. Adjunk neki nevet (mondjunk szkriptjeink), majd rajta jobb egérrel válasszuk a **szerkesztés** pontot. Menjünk a **felhasználó konfigurációja** ponton belül a **házirendek → Windows beállításaiba**, majd ott a **parancsfájlok (bejelentkezés/kijelentkezés)** részre, és a jobb oldalon válasszuk a **bejelentkezést**. A hozzáadás gombbal tudjuk

hozzáadni a kívánt szkriptet a házirendhez. A végén a házirendbejegyzésünkön jobb egérrel kattintva válasszuk az „**érvénybe léptetve**” pontot.



Ezzel gyakorlatilag el is készültünk, használatra kész a rendszerünk, amely nemcsak tudja azt amit a Samba 3 tudott, de jócskán van még benne lehetőség.

A teljes Samba 4 konfigurációs állományt azért a végére beszúrjuk:

```
[global]
        workgroup = LOK
        realm = lok.local
        netbios name = SZERVER4
        server role = active directory domain controller
        dns forwarder = 8.8.4.4
        interfaces = eth1 lo
        bind interfaces only = yes
[netlogon]
        path = /var/lib/samba/sysvol/lok.local/scripts
        read only = No
[sysvol]
        path = /var/lib/samba/sysvol
        read only = No
[home]
         path = /home/samba/home/
        read only = No
```

```
[Profiles]
        path = /home/samba/Profiles/
        writable = yes
[publikalas]
        path = /home/samba/publikalas
        comment = tanarok szamara publikalasra
        browseable = yes
        writable = yes
[adminiszt]
        path = /home/samba/adminisztracio
        comment = adminisztrativ dolgozoknak, zart
        browseable = yes
        writable = yes
[adminiszt2]
        path = /home/samba/adminisztrac2
        comment = adminisztrativ dolgozoknak, nyilt
        browseable = yes
        writable = yes
[hr]
        path = /home/samba/hr
        comment = hr-es dolgozoknak
        browseable = yes
        writable = yes
[rendszerg]
        path = /home/samba/rendszergazda
        comment = rendszergazdai publikaciok
        browseable = yes
        writable = yes
[Deskjet_3510]
        comment = Kozpotni nyomtato adminisztrativ dolgozoknak igazgatasi csoport
        path = /var/spool/samba/
        printer name = Deskjet_3510
        printable = yes
```

Hálózati nyomtatás, szkenner szerver

Nyomtatás Linuxon

Linuxon használt nyomtatási rendszer

Linuxon régebben az LPD[139] nyomtatási rendszereket használtuk, de az elmúlt évek alatt fokozatosan kiszorította azt a CUPS[140] (Common Unix Printing System), amely rengeteg modern nyomtató használatát teszi lehetővé. Szabványos IPP[141] protokollt használ, támogatja a PostScript nyomtatóleírást (PPD[142]), ráadásul egy könnyen használható webes felületű adminisztrációs rendszert is biztosít.

A nyomtatási rendszer mellett érdemes még megemlíteni a nyomtatómeghajtókat is, melyek közül a legismertebbek a gutenprint[143] csomag, amely jelenleg 700 nyomtatót támogat, illetve a foomatic[144]. Itt kell még megismerkednünk a Ghostscript[145] csomaggal is, mely arra képes, hogy az alkalmazások által küldött PostScript formátumú adatokat fordítsa le a PostScriptet nem ismerő nyomtató számára értelmezhető adatfolyammá, ezt használja például a foomatic is. (Ebből sejthető, az a legegyszerűbb, ha valódi PostScript-képes nyomtatót tudunk használni, sajnos ezt azonban a pénztárcánk gyakran megakadályozza...) Talán ez kissé zavaros most – ami őszintén elismerve nem csoda –, de ebből a lényeg, hogy ha nyomtatni is szeretnénk, akkor a nyomtatási rendszeren (CUPS) kívül érdemes ezeket is feltelepítenünk, amennyiben az általunk használt Linux rendszer nem tette volna ezt meg automatikusan a CUPS telepítésekor.

Amennyiben még nem kapkodtuk volna a fejünket, akkor jöjjön itt be még pár szereplő! A HPLIP[146] (Hewlett-Packard's Linux Imaging and Printing), amely kimondottan a HP eszközök támogatására készült, nem csak nyomtatókat, de szkenner- és FAX-támogatást is tartalmaz, és szerencsére GPL licenc alatt érhető el. Jelenleg 2295 eszközt támogat, és a CUPS-ot, mint háttérprogramot használja.[147]
Pár nyomtatót pedig az SpliX[148] segítségével tudunk használni, amely SPL-en alapul (Samsung Printer Language).

A fentiek fényében sejthető, hogy elég sok nyomtatót tudunk Linux alatt használni, de azért érdemes beszerzés előtt tájékozódni.

139. http://en.wikipedia.org/wiki/Line_Printer_Daemon_protocol
140. http://www.cups.org/
141. http://en.wikipedia.org/wiki/Internet_Printing_Protocol
142. http://en.wikipedia.org/wiki/PostScript_Printer_Description
143. http://gimp-print.sourceforge.net/
144. http://en.wikipedia.org/wiki/Foomatic
145. http://www.ghostscript.com/Ghostscript.html
146. http://hplipopensource.com/hplip-web/index.html
147. http://hplip.sourceforge.net/images/hplip_architecture.png
148. http://splix.ap2c.org/

Nyomtató kiválasztása

Amennyiben újonnan kerül a nyomtató beszerzésre, mindenképp győződjünk meg arról, hogy használható-e az eszköz Linuxon.

Az egyik legismertebb oldal, ahol utána tudunk nézni, hogy mely nyomtató Linux-kompatibilis, az a következő openprinting oldal: http://www.openprinting.org/printers

Itt gyártó és típus szerint tudunk keresni, azonban a legújabb nyomtatók közül meglehetősen sok hiányzik. Ez azonban nem jelenti feltétlen azt, hogy azok nem működnek Linuxon. Csak máshol is körül kell néznünk…

Szerencsére egyre több gyártó ismerte fel, hogy mennyire fontos az általa gyártott eszközök Linux-támogatása, így egyre több nyomtatót tudunk használni, azonban ennek a gyártó oldalán kell utánanéznünk.

Sok nyomtató, csak a gyártói oldalról (vagy máshonnan) leszedett PPD fájllal használható; vagy például a HPLIP legújabb verziója kellhet sok korszerű nyomtatóhoz, ami esetleg ismét csak a gyártói oldalról tölthető le, mert az általunk használt disztribúció csak egy régebbi verziót tartalmaz, amivel a nyomtató nem működik. Azt hiszem, sikerült eléggé hangsúlyozni mennyire fontos az előzetes tájékozódás.

A gyártói oldalakról letöltött nyomtatómeghajtók szerencsére könnyen telepíthetőek, általában minden függőséget letöltenek maguknak, és gyakran még grafikus telepítőfelülettel is rendelkeznek (tapasztalatból javasolt azonban a gyártói megoldások telepítése **előtt** a CUPS telepítése).

Maga a telepítés egyszerűbb, mint a fentiek alapján sejteni lehetne…

CUPS telepítése

A telepítéshez használt rendszer a jelenlegi stabil Debian Linux, a Wheezy.[149] A telepítéshez az apt-t használjuk:

```
apt-get install cups
```

Vegyük észre, hogy ezzel nem csak a CUPS szervert telepítettük, de sok más csomag mellett felkerült függőségként a printer-driver-gutenprint, a foomatic-filters, a ghostscript-cups és a hplip is. Csupa korábbról ismerős kifejezés…

Ezek után már el is indult a gépünkön a CUPS. A

```
ps ax
```

parancs kiadása után a következőhöz hasonló eredményt láthatunk:

```
/usr/sbin/cupsd -C /etc/cups/cupsd.conf
```

Rögtön észrevehetjük, hogy konfigurálni majd az */etc/cups/cuspd.conf* fájlban tudjuk, illetve ha kiadjuk a

```
netstat -lnp
```

parancsot, akkor látható, hogy jelen pillanatban a localhoston válaszol a 631-es porton:

```
127.0.0.1:631           0.0.0.0:*               LISTEN      5890/cupsd
```

149. https://wiki.debian.org/DebianWheezy

CUPS alap konfigurálása

Gyakorlatilag a legegyszerűbb megoldás az, ha azon a gépen, amire telepítettük a CUPS-ot, van grafikus felület és egy böngésző, mivel a böngésző címsorába beírva a https://127.0.0.1:631 címet betöltődik a CUPS webes adminisztrációs felülete.

Szervereknél gyakori, hogy nincs telepítve grafikus felület, ilyenkor engednünk kell, hogy más helyekről is hozzá lehessen férni a webes felülethez. Ehhez először engedélyeznünk kell, hogy a localhost-tól eltérő interfészen is lehessen kapcsolódni a CUPS-hoz, utána pedig a hozzáférési szabályoknál kell megadjuk, hogy mely helyekről lehet kapcsolódni a szerverhez.

Mindegyiket az /etc/cups/cupsd.conf szerkesztésével tudjuk megtenni.

Keressük meg a Listen sort, majd bővítsük:

```
Listen localhost:631
  Listen 10.0.0.1:631
```

Eredetileg csak a localhost sor szerepelt a konfigurációs állományban, ez után szúrtuk be a következő „Listen” sort, ahol megadtuk annak a csatolókártyának az IP címét, amelyen szeretnénk, hogy működjön. Ez jelenleg a teszt gépünk belső hálózat felé néző csatolókártyája.

Azonban ha ekkor elnavigálnánk a böngészővel a szerver címére (http://10.0.0.1:631), egy visszautasító üzenet várna minket, ugyanis engedni kell, hogy lehessen *csatlakozni a szerverhez*, az *adminisztrációs felülethez* és a *konfigurációs állományhoz*:

```
# Restrict access to the server...
  <Location />
    Order allow,deny
    Allow from 10.*.*.*
  </Location>

  # Restrict access to the admin pages...
  <Location /admin>
    Order allow,deny
    Allow from 10.*.*.*
  </Location>

  # Restrict access to configuration files...
  <Location /admin/conf>
    AuthType Default
    Require user @SYSTEM
    Order allow,deny
    Allow from 10.*.*.*
  </Location>
```

A fenti három bejegyzés vonatkozik sorban a szerver, az adminisztrációs felület és a konfigurációs állomány elérésére. Mi az „**Allow from 10.*.*.***„ sort szúrtuk be, amivel azt érjük el, hogy a **teljes 10.0.0.0 hálózatból elérhetővé vált** mindhárom szempont alapján. Megadható lenne például Allow from all, azaz mindenhonnan; vagy Allow from 10.0.0.5, azaz csak a 10.0.0.5-ös gépről; vagy *domain vagy gépnév.domain formákban is.

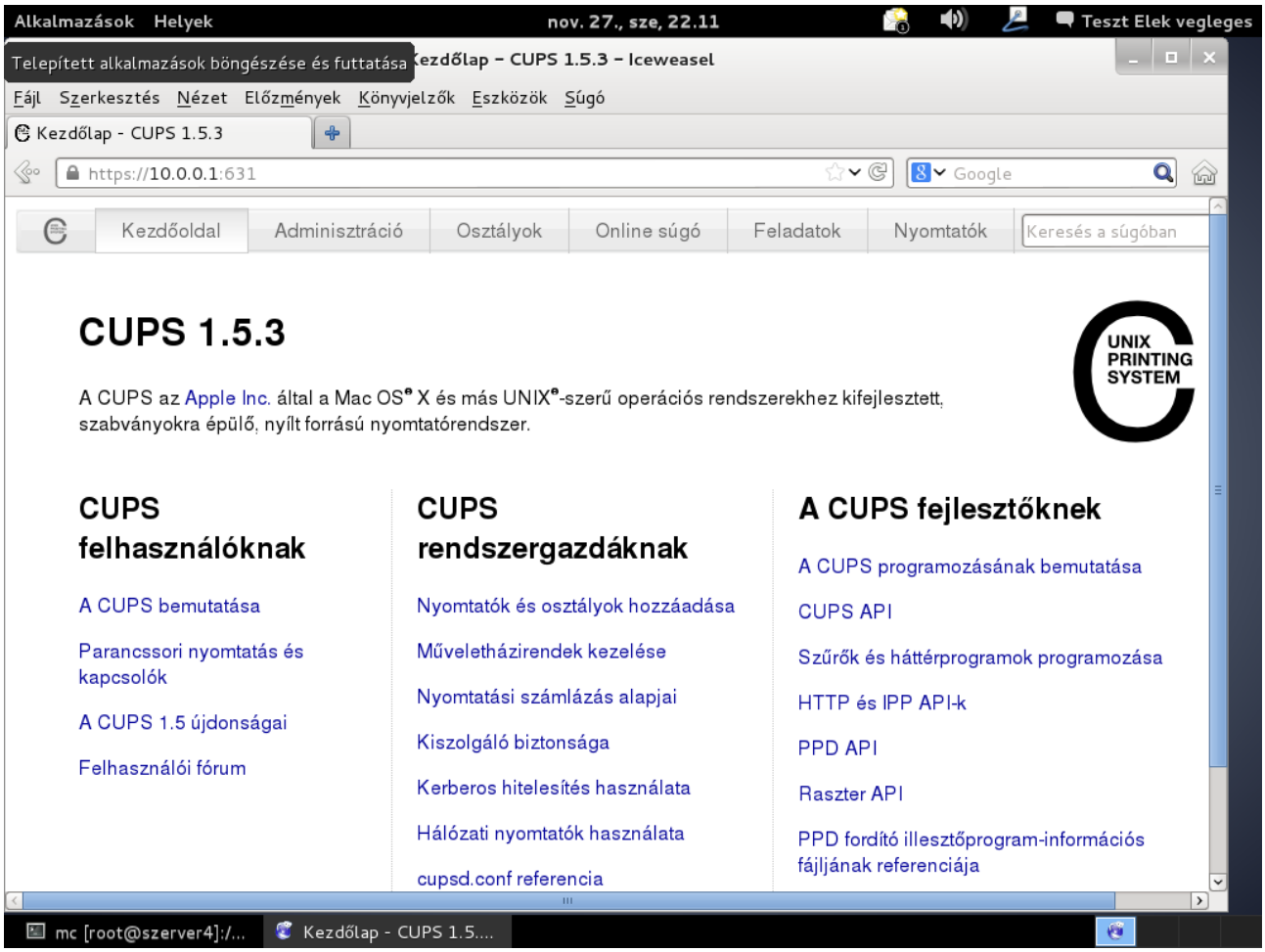
Ezek után már csak arra van szükség, hogy újraindítsuk a CUPS-ot:

```
/etc/init.d/cups restart
```

Ettől a ponttól máris tudjuk a belső hálózatunk bármely gépéről egy böngészőn keresztül adminisztrálni a CUPS szervert, és persze a nyomtatókat.

CUPS webes felülete

A böngészőnkbe írjuk be a CUPS szerver címét, amely jelen példában a https://10.0.0.1:631, majd fogadjuk el a tanúsítványt, és máris elénk tárul a webes adminisztrációs felület.



*Az adminisztráció felületen **csak hitelesítés** után tudunk bármit csinálni. Jelenleg egyedül a **root** felhasználónak van joga mindenhez, így a **root nevét és jelszavát** kell beírni a párbeszédablakba.*

Nyomtatóadminisztrátor felhasználó beállítása

Amennyiben szerencsénk van, és van kijelölt vagy esetleg önként vállalkozó, aki egyéb elfoglaltságaink miatt magára vállalja az elakadt, rossz nyomtatások törlését, nyomtatók újraindítását stb., vagy mi szeretnénk inkább a saját felhasználói nevünket használni root helyett a CUPS webes adminisztrációjakor, akkor csak annyi a teendőnk, hogy a kívánt felhasználót az lpadmin csoport tagjává kell tenni.

```
usermod -a -G lpadmin muszashi
```

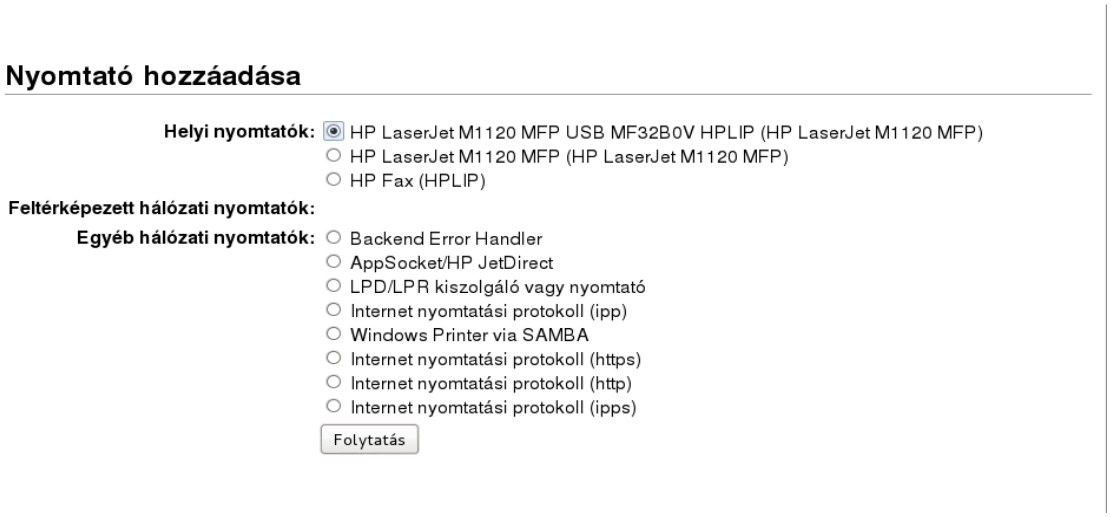
A fenti konzolon kiadott paranccsal a muszashi nevű felhasználó mostantól tudja adminisztrálni a nyomtatókat.

Nyomtatók hozzáadása

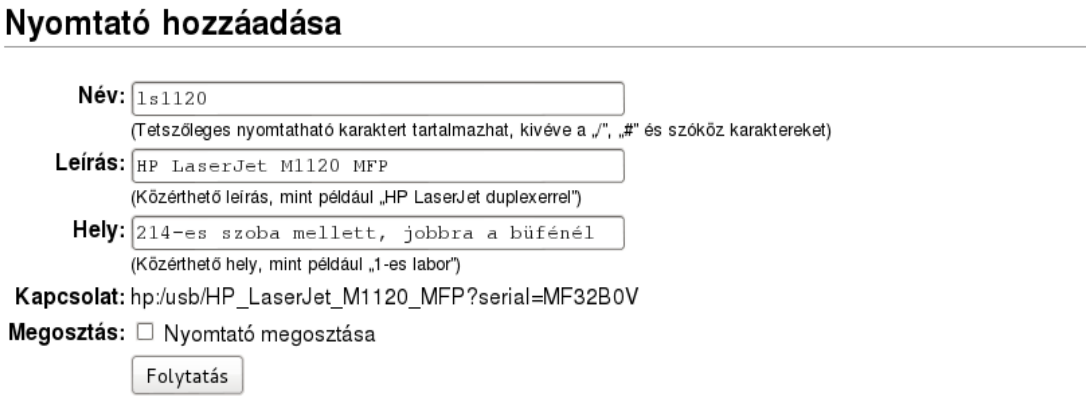
Mielőtt bármit is tennénk, ellenőrizzük, hogy a nyomtató egyáltalán elérhető-e! Főként hálózati nyomtatók telepítésekor fordul elő, hogy valami triviális hiba miatt a nyomtató nem is elérhető a hálózaton, így mielőtt időt áldoznánk a hibakeresésre, a ping paranccsal nézzük meg, hogy a nyomtató fizikailag elérhető-e a hálózaton (USB nyomtató esetén az lsusb paranccsal).

A telepítés itt kétféleképpen történhet

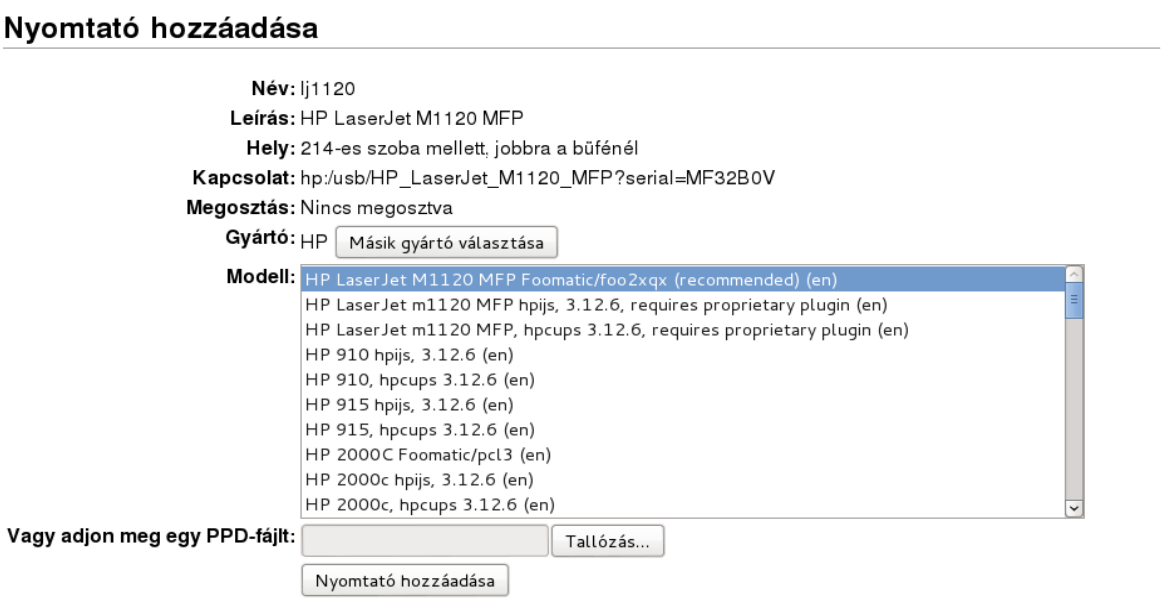
– Amennyiben gyártóoldali telepítőprogramot használunk, úgy kövessük a gyártó és a program utasításait. Ha minden jól megy, a telepítés végén látni fogjuk a nyomtatót a CUPS adminisztrációs felületén a telepített nyomtatók között, és ugyanúgy tudjuk kezelni, mintha mi telepítettük volna.
– Amennyiben a nyomtató támogatott a rendszerünkön, akkor az adminisztrációs felület Nyomtató hozzáadása alatt tudjuk felvenni a nyomtatót, 4 egyszerű lépésben:



– Válasszuk ki a nyomtató kapcsolódási pontját, majd kattintsunk a folytatásra!

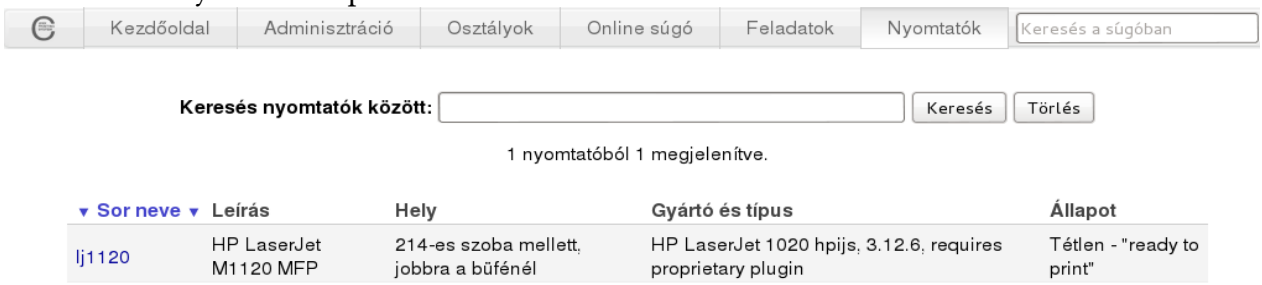


– Adjuk meg a nyomtató leírását és nevét! Amennyiben a nyomtatót szeretnénk megosztani, érdemes a nevet átírni az alapértelmezetten javasolt névnél rövidebbre, hogy könnyebb legyen arra hivatkozni, és csökkentsük a hibalehetőséget.



– Válasszuk ki a nyomtatómeghajtót!
Ezzel el is érkeztünk a kritikus kérdéshez, hogy melyik meghajtót válasszuk. Amennyiben több lehetőségünk is van, akkor először válasszuk az ajánlottat (recommended), de ha az nem működne, vagy nem hozná a kívánt eredményt, akkor a nyomtatót majd tudjuk módosítani, és másik meghajtót választani. A különböző nyomtatóknál eltérő, hogy épp melyik meghajtó hoz jobb eredményt. A nyomtatás beállításának pont az a leglényegesebb része, hogy az adott nyomtatóhoz való meghajtók mennyire felelnek meg. Szerencsére az utóbbi években a gyártói (proprietary) meghajtók sokat javultak, az esetek nagy részében megfelelően, tán jobban is működnek, mint a foomatic/gutenprint meghajtók, így talán érdemes először azokkal próbát tenni.
Van egy speciális „meghajtó” is, ez pedig a RAW. Ez gyakorlatilag semmit nem csinál, csak a beérkező adatfolyamot változtatás nélkül továbbküldi a nyomtatóra. Gyakran használjuk például akkor, ha Samban osztjuk meg a nyomtatót: ekkor ugyanis a Windows kliens oldalon feltelepítjük az adott nyomtató meghajtóprogramját, így már eleve olyan adatfolyam érkezik, amelyet a nyomtató értelmezni tud, így azt egyszerűen módosítás nélkül kell továbbküldenünk, no erre való a RAW.
Amennyiben a nyomtatóhoz PPD fájlt szereztünk, akkor azt itt tudjuk feltölteni a szerverre.

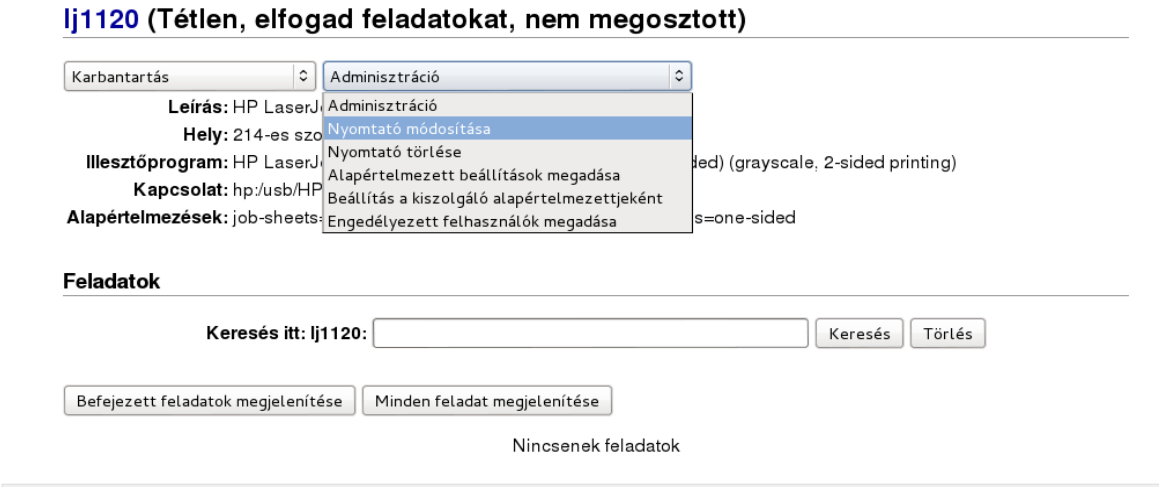
– Állítsuk be a nyomtató alapértelmezéseit!



Amennyiben a negyedik lépésnél az állapot helyén hibaüzenet fogad, akkor

– vagy nem érhető el a nyomtató
– vagy telepítés után szüneteltetésre került (előfordul), engedélyezzük a folytatást a karbantartás pontban, és „megjavul”
– indítsuk újra a CUPS szervert
– próbálkozzunk másik meghajtóval
– Most már látjuk a telepített nyomtatót a nyomtatók között. Készen is vagyunk.

Nyomtatót módosítani, adminisztrálni bármikor tudunk, amennyiben a nyomtatóra kattintunk, vagy az adminisztráció pontban kiválasztjuk.



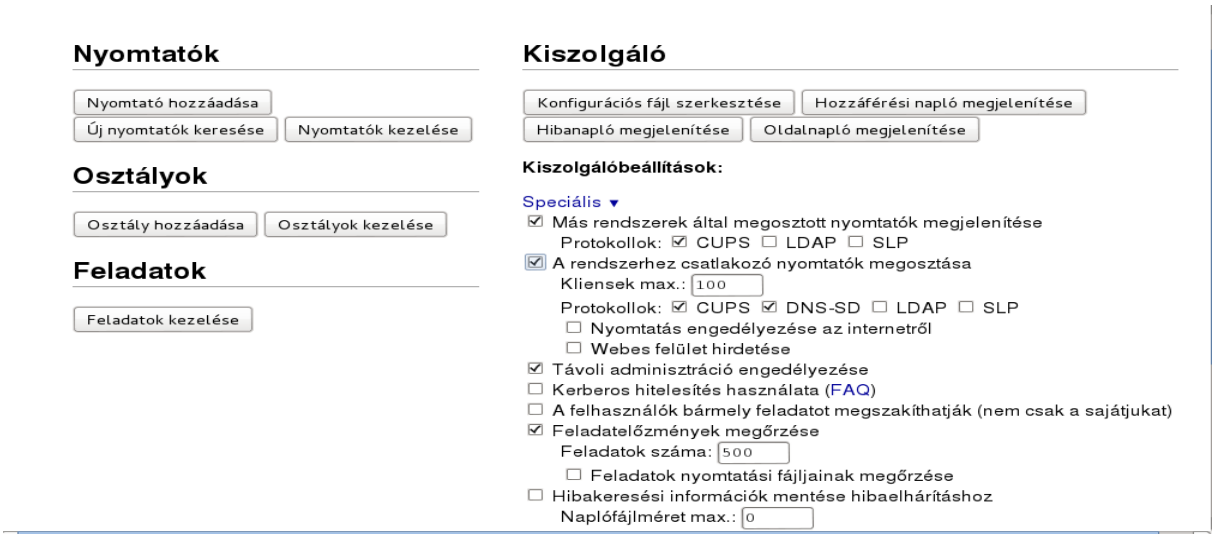
Amennyiben a nyomtató nem helyben, hanem hálózaton keresztül kapcsolódik, ugyanígy járunk el, csak az első oldalon a nyomtatónak megfelelő kapcsolódási formát kell választani, és megfelelő adatokkal kitölteni. Erre látható példa az alábbi nyomtató esetén:

Nyomtatók megosztása a hálózaton

A nyomtató megosztásánál érdemes eldönteni, mit is szeretnénk pontosan. Ha cél, hogy csak bizonyos felhasználók, csoportok tudjanak nyomtatni bizonyos nyomtatókra, és van mondjuk egy Samba fájlszerverünk, akkor érdemes a nyomtatót azon keresztül elérhetővé tenni a kliensek számára. Ez a legegyszerűbb út, és a leginkább finomhangolható. Erről ezen kiadvány Sambával kapcsolatos részében lehet olvasni.

Amennyiben csak egyszerűen azt szeretnénk, hogy a hálózatunk bármely tagja nyomtathasson a nyomtatókra, akkor legegyszerűbb rögtön CUPS-on keresztül megosztani a nyomtatókat. (Természetesen az elején említett módon szabályozhatjuk, a hálózat mely gépeiről, alhálózatokról érhető el a CUPS, illetve felhasználók alapján is lehet szabályozni az elérést, de akkor inkább Sambán keresztül érdemes megosztani)

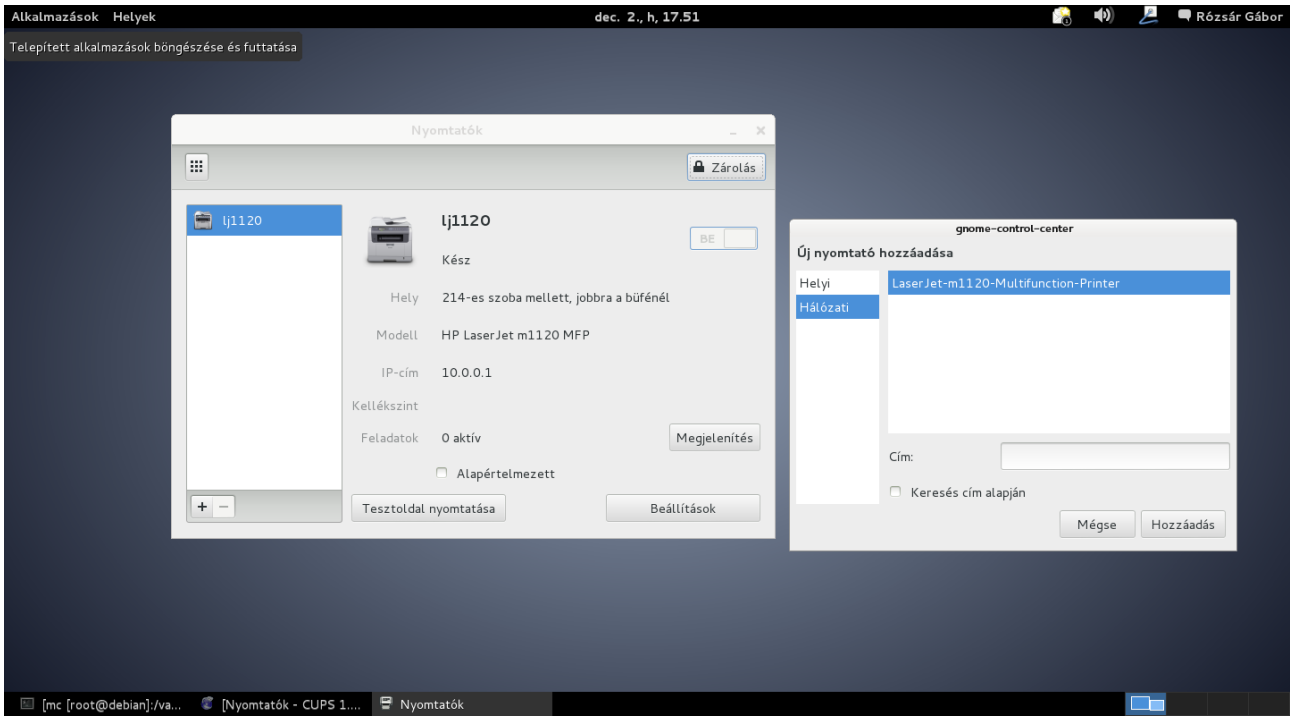
A megosztáshoz a webes felületen, jelöljük be a *„rendszerhez csatlakozó nyomtatók megosztása”* négyzetet, illetve az adott nyomtatóknál engedélyezzük a megosztást. Innentől kezdve a kliensek elérhetik a nyomtatót, nyomtatókat.



Linux kliensek

Linux kliens esetén gyakorlatilag semmi teendő nincs, szinte valamennyi grafikus nyomtatókezelő programban automatikusan megjelenik a megosztott nyomtató, vagy oda 1-2 kattintással könnyedén felvehető. *(Amennyiben nem jelenne meg, ellenőrizzük, hogy a kliensen telepítve van-e a CUPS, illetve a http://127.0.0.1:631 címen elérhető a kliensen lévő CUPS, ahol az Adminisztráció > Kiszolgáló > Speciális pont alatt engedélyezhetjük, hogy más szerveren megosztott nyomtatók is megjelenjenek a mi CUPS szerverünkön – lásd előző ábra.)*

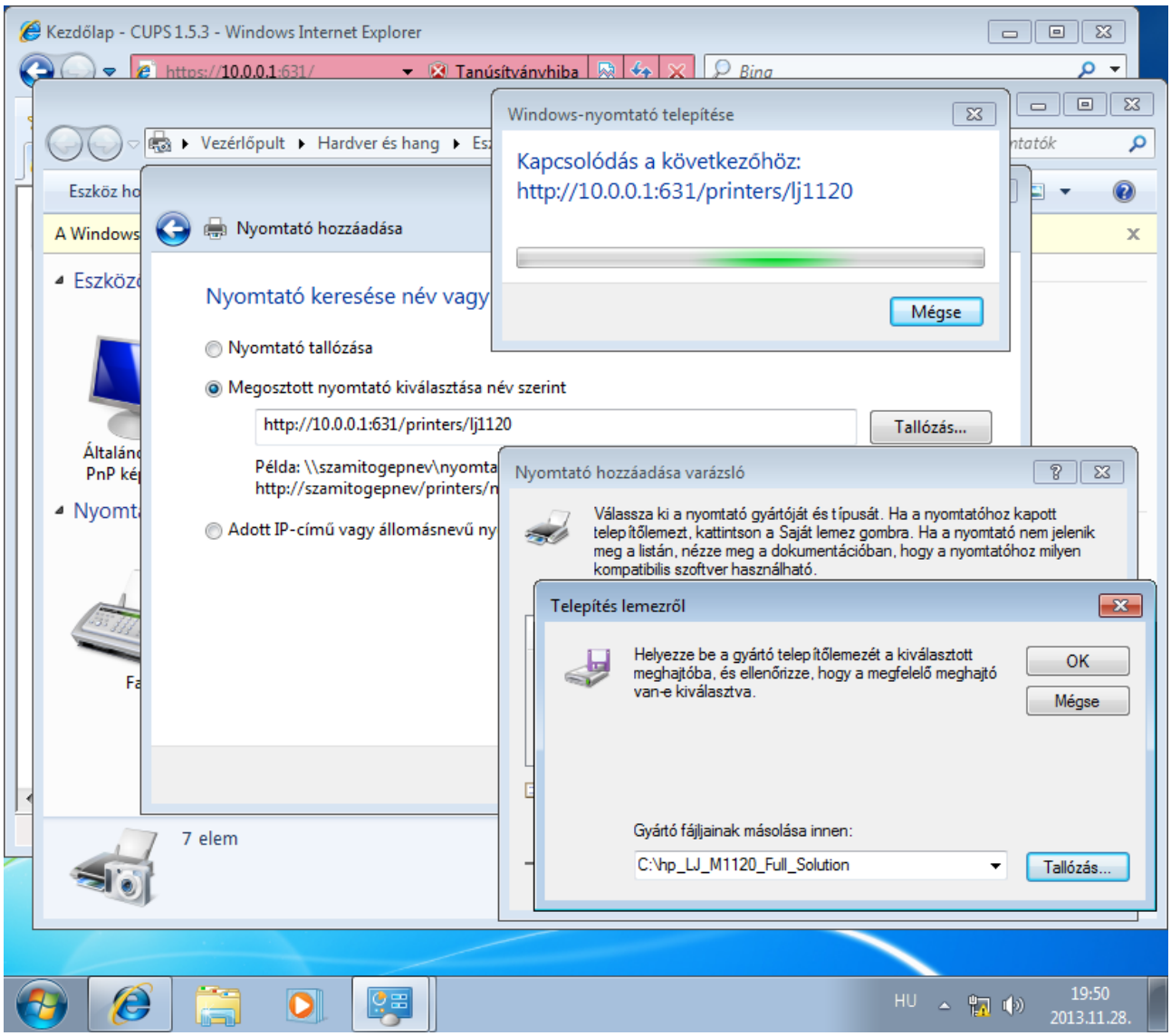
Legtöbbször semmi teendőnk nincs, automatikusan megjelenik:



Windows kliensek

Windows kliensen is könnyen elérhetővé tehetjük a megosztott nyomtatóinkat a nyomtató hozzáadása ponton keresztül. Válasszuk a hálózati nyomtató hozzáadását és ott pedig a „Megosztott nyomtató kiválasztása név szerint pontot". Elérkeztünk oda, ahol nem szabad követni a nyomtató eléréséhez az ablakban látható megadott mintát, ugyanis a *http://szamitogepnev/Printers/ nyomtatonev/.printer* minta nem működik!

Helyette adjuk meg a ***http://cups szerver neve vagy IP címe:631/printers/nyomtatónév*** formában, ahogy az alábbi képen is látható:



Ezek után a telepítés lezajlik és a nyomtató használható.

Egyéb CUPS ötletek

Amennyiben szükséges, hogy a nyomtatásokat nyilvántartsuk, akkor a webes felületen kényelmesen megnézhetjük minden korábbi nyomtatási munkának a bejegyzését. Néha kívánatos, hogy kimutatást vagy egyéb jelentést készítsünk a nyomtatásokról, ilyenkor viszont hasznosabb, ha ezek az adatok fájlban is megtalálhatóak. Ez a fájl létezik: a /var/log/cups/page_log nevezetű naplófájlban láthatjuk a nyomtatások naplózását. (Szintén itt található a szervert és a hozzáférést naplózó állomány is. {error_log és access_log}, probléma esetén ezekben keresgéljünk.)

Lehetőség van arra is, hogy ne webes felületen, hanem parancssorban konfiguráljunk nyomtatókat, illetve kezeljük a nyomtatásokat.

Lekérdezni a jelen példában szereplő nyomtató állapotát így lehet:

```
lpstat -p lj1120
```

A CUPS parancssori eszközeiről itt lehet részletesebben olvasni:
http://www.cups.org/documentation.php/options.html

Szkennelés Linuxon

Linux rendszereken a szkenner használatát a SANE[150] (Scanner Access Now Easy) segítségével tudjuk megvalósítani, amelynek célja az egységes hozzáférés biztosítása minden képolvasó hardverhez. Igen, a SANE nem csak szkennerrel, hanem például kamerákkal is együttműködhet, itt most azonban csak a szkennerről lesz szó.

Szkenner támogatottsága

Mielőtt eszközbeszerzésbe kezdenénk, ugyan azt kell tegyük, mint a korábbi oldalakon a nyomtatókkal kapcsolatban olvasható volt már. Meg kell nézni, támogatja-e az eszközt a SANE jelenlegi operációs rendszerünkön található verziója, esetleg a SANE oldaláról letölthető újabb verzió, vagy pedig az eszköz gyártójának oldaláról letölthető-e egy Linuxos telepítőprogram. Beszerzésünket ennek figyelemvételével tegyük meg.

A SANE adatbázisa a támogatott eszközökről: http://www.sane-project.org/sane-mfgs.html

Jelen pillanatban 2036 eszközt ismer, melyből 1087 kapott teljes vagy jó minősítést. Amelyik eszköz ezeknél gyengébb minősítésű, annak a használata igencsak kétséges. A fenti adatbázison túl újabb eszközök használatakor ismét érdemes a gyártói weboldalakat is felkeresni, mivel például a nyomtatásnál már említett HPLIP is rengeteg képolvasó eszköz (szkenner, multifunkciós eszköz) használatát teszi lehetővé.

Telepítés

A SANE tulajdonképpen háttér- (backend) és előtér- (frontend) programokból áll, amelyek korábban a jól elkülönült sane-backend (tulajdonképp a szkenner elérését teszi lehetővé) és sane-frontend (a szkenner tényleges használatához szükséges) neveken voltak telepíthetőek, mára azonban már az általunk használd Debian Wheezy rendszeren is csak egyszerűen sane néven hivatkozhatunk rá.

```
apt-get install sane
```

A fenti paranccsal telepíthetjük a sane csomagot. A szükséges függőségeket hozza magával. (Amennyiben szkennerünk igényli, ellenőrizzük, hogy a libsane-extras és a hplip telepítésre került-e. Ha nem, pótoljuk!)

Konfiguráció

Amennyiben a szkennerünk valóban támogatott, akkor az most már le is kérdezhető. Ehhez csak a következő utasításra van szükség:

```
scanimage -L
```

Kilistázza az észlelt képolvasó eszközöket. Amennyiben nincs képolvasó eszköz kilistázva, akkor vagy nem támogatott az eszköz, vagy egyéb probléma van vele. Jelen példánkban ezt kaptuk a parancs kimeneteként:

150. http://www.sane-project.org/

```
device `hpljm1005:libusb:001:005' is a Hewlett-Packard LaserJet M1120 multi-
  function peripheral
```

Szkenner használata

Parancssorból

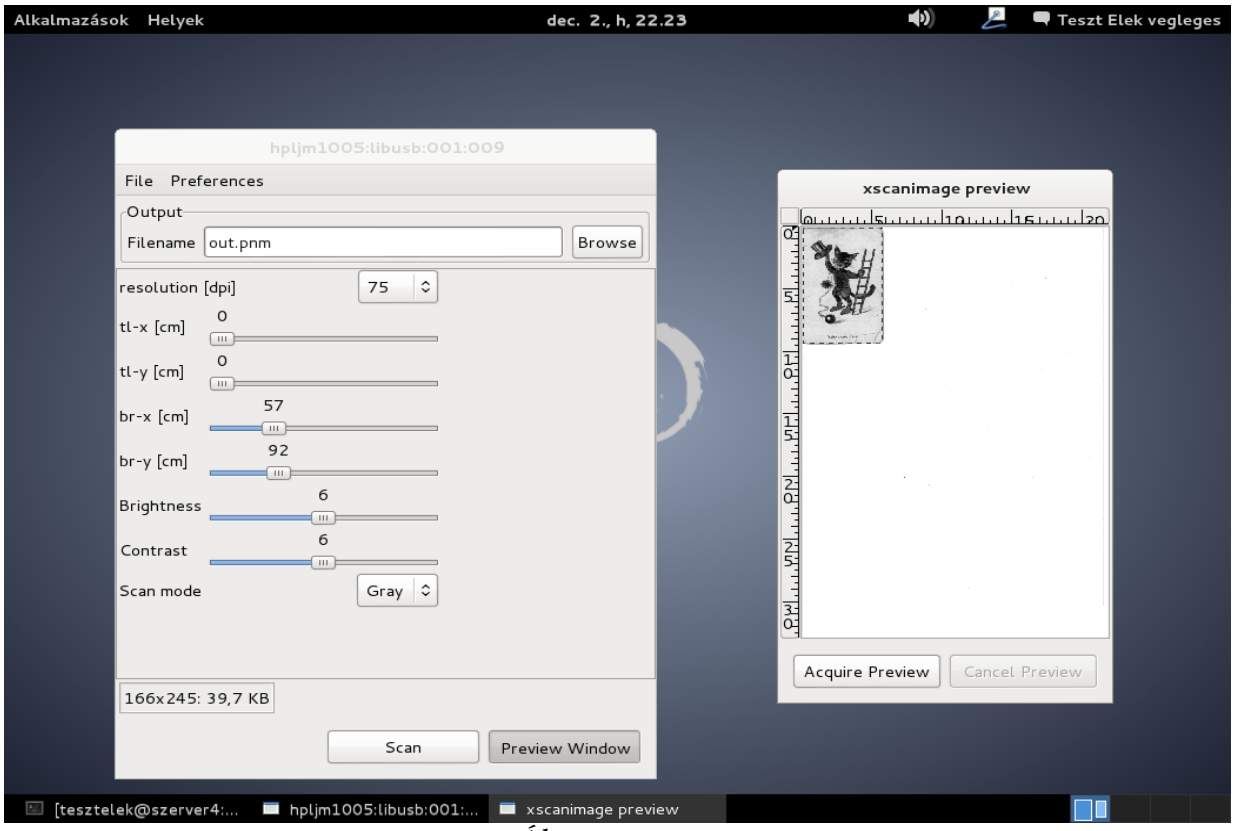
Tulajdonképpen a szkenner már használható, a működését le is tudjuk ellenőrizni a már előzőleg megismert scanimage parancssoros eszközzel, amely lehetővé teszi a szkennelést terminálból.

```
scanimage --mode Color --format tiff -p -d hpljm1005:libusb:001:005 >teszt.tiff
```

Azaz készít egy színes képet tiff formátumban a megadott eszközzel, és a beolvasott képet teszt.-tiff néven tárolja. A -p hatására pedig lesz egy állapotjelzőnk is, amin nyomon tudjuk követni a szkennelés előrehaladását. Amennyiben a szkennelés rendben lezajlott az eszköz működőnek tekinthető, lehet próbálkozni a grafikus eszközökkel.

Grafikus felületen

A SANE telepítésével rögtön kaptunk egy grafikus előtétprogramot is az **xscanimage** nevezetűt. Ezt elindíthatjuk, illetve a szkennert kezelő programok (pl. GIMP) már meg is hívják ezt az eszközt, ha a szkennert választjuk adatforrásnak. Nagyon egyszerű kis program, csak az alapvető funkciókat tudja.

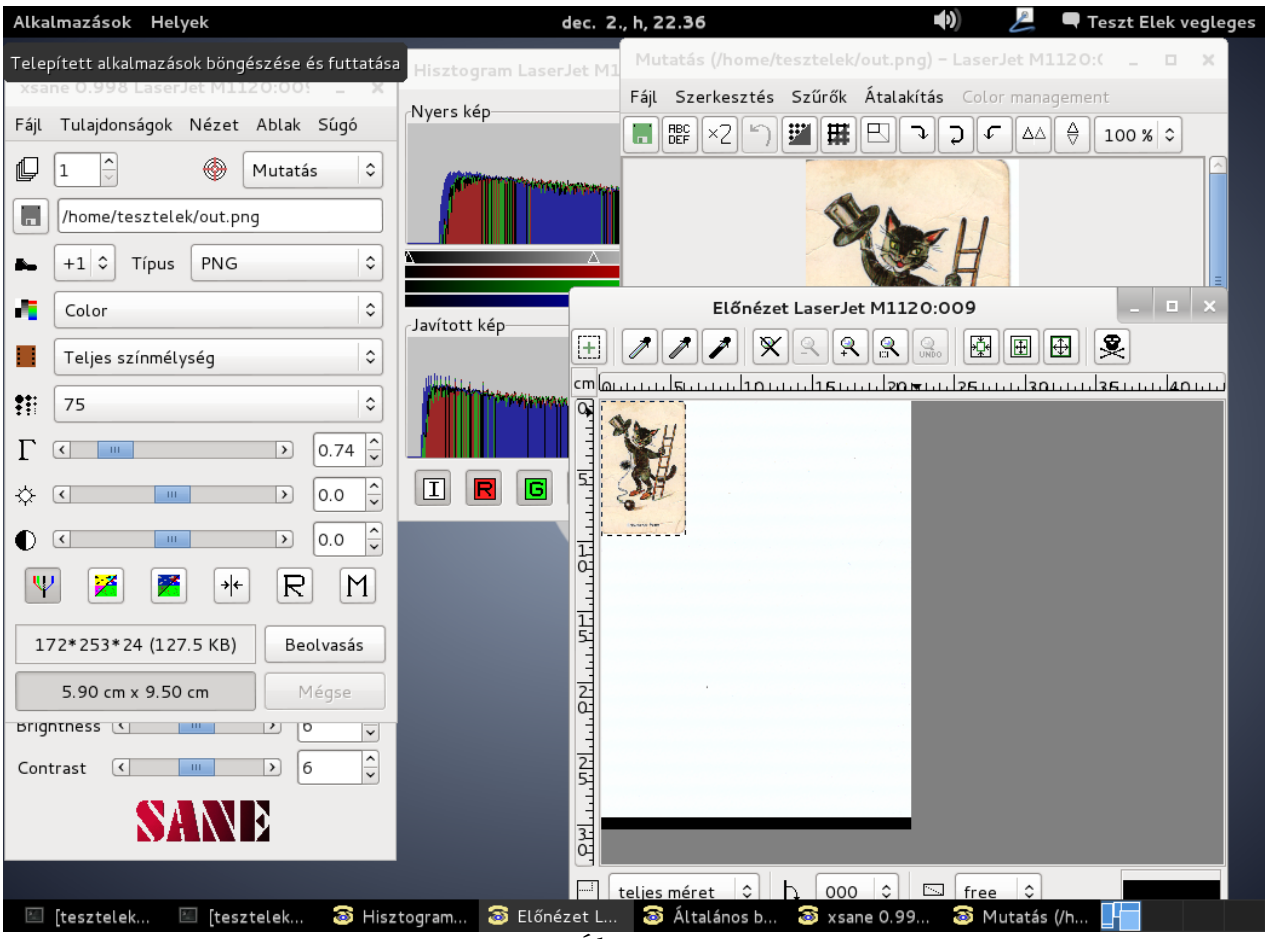


1. Ábra: xscanimage

Az xsanimage programnál vannak lényegesen nagyobb tudású eszközök is, ilyen például az **xsane**, amely amellett hogy sokkal több lehetőséggel rendelkezik, okozott már pozitív meglepetést, amikor a nem épp tökéletes eszközmeghajtó az xscanimage programmal hibásan működött, míg a **xsane**-vel kompromisszumok árán, de lehetett használni...

```
apt-get install xsane
```

A fenti paranccsal telepíthető az xsane.



2. Ábra: xsane

Látható, hogy az xsane sokkal több funkcióval rendelkezik.

Hálózati lapolvasás

Telepítés

A sane-utils csomag lehetővé teszi, hogy a gépünkhöz csatlakoztatott szkenner a hálózat többi gépéről is elérhető legyen. Amennyiben eddig nem telepítettük fel, azt tegyük meg most az alábbi parancs begépelésével:

```
apt-get install sane-utils
```

A sane utils által szállított **saned**[151] fogja elérhetővé tenni a szkennerünket, azonban ehhez még telepítenünk kell a **xinetd**[152] csomagot is (Internet superserver).

```
apt-get install xinetd
```

Most, hogy már telepítettük a sane démont és xinet démont is, csak konfigurálnunk szükséges a rendszert.

151. http://penguin-breeder.org/sane/saned/
152. http://en.wikipedia.org/wiki/Xinetd

Konfiguráció

– Kezdjük a /etc/default/saned fájllal. Itt módosítsuk a RUN=yes sorra a korábbi RUN=none sort, ezzel elérjük, hogy elindulhasson a saned.

```
# Set to yes to start saned
  RUN=yes
```

– Folytatásképp keressük meg az /etc/sane.d/saned.conf fájlt, és egyszerűen írjuk bele a hálózatunk vagy azon gépek címét vagy nevét, amely irányba elérhetővé szeretnénk tenni a szkennert.

```
# The hostname matching is not case-sensitive.
  10.0.0.0/8
```

A fenti beállításokkal tulajdonképp a saned konfigurációja véget ért, már csak a xinetd van hátra.

– Létre kell hoznunk egy állományt az /etc/xinetd.d/ könyvtárban, amelynek tetszőleges nevet adhatunk, de célszerű mondjuk saned-nek elnevezni (pl. touch /etc/xinetd.d/saned), majd ebbe az alább látható tartalmat elkészíteni (pl pico /etc/xinetd.d/saned):

```
service saned
  {
  socket_type = stream
  server = /usr/sbin/saned
  protocol = tcp
  user = saned
  group = scanner
  wait = no
  disable = no
  }
```

Gyakorlatilag a fentiek megadásával adtuk a xinetd tudtára, hogy mit és hogyan kell elindítania, ha kérés érkezik hozzá.

A konfigurálás elkészült, indítsuk újra a xinet démont:

```
/etc/init.d/xinetd restart
```

Kliens oldali lehetőségek

Linux kliens

Linuxos kliensen rögtön van lehetőségünk tesztelésre (természetesen ehhez a sane csomagnak telepítve kell lennie a kliens gépen is), adjuk ki ismét:

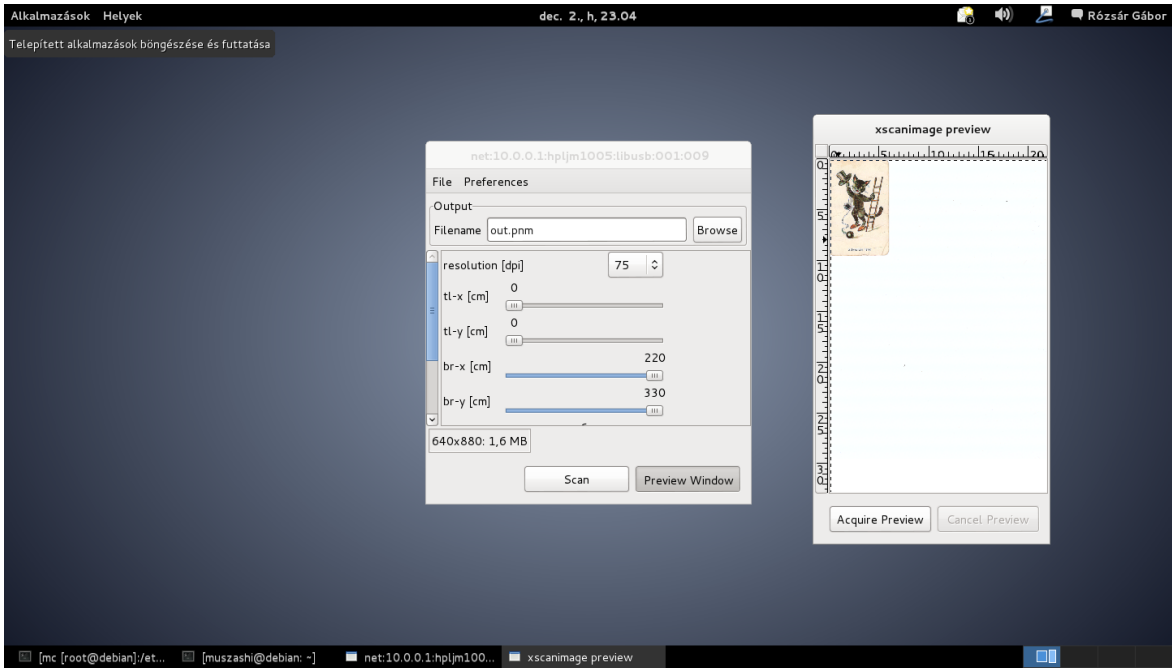
```
scanimage -L
```

A kiszolgálótól eltérően immár hálózati címmel látjuk a korábban telepített szkennert:

```
device `net:10.0.0.1:hpljm1005:libusb:001:009' is a Hewlett-Packard LaserJet
  M1120 multi-function peripheral
```

A fenti sort látva biztosak lehetünk, hogy elérjük a hálózati lapolvasót.

Ezek után a szokásos grafikus felületű eszközök közül válogathatunk, ugyan úgy fognak működni mintha helyi eszközzel dolgoznának.



A korábbi képhez képest csak annyiban különbözik ez, ami a hálózati kliensen készült, hogy látható az xscanimage címsorában, hogy valójában egy hálózati eszközzel dolgozik (net 10.0.0.1).

Windows kliens

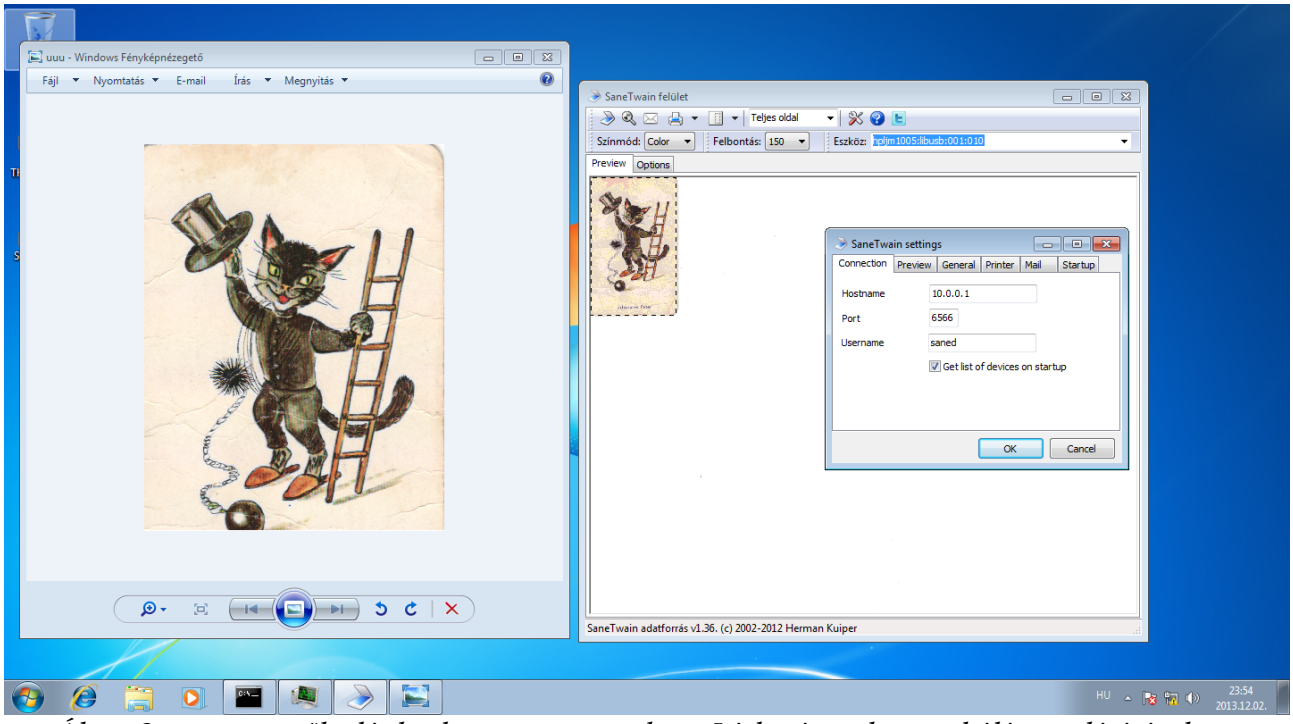
Windowsos kliens esetén első lehetőségként arra gondolunk, hogy az xsane rendelkezik win32-es binárisokkal is, így akár használhatnánk azt is, de a weboldalának tanúsága szerint ezt évek óta nem fejlesztik, nem tanácsos a használata.

Helyette a **sanetwain** programot érdemes használni, mely elérhető az alábbi címről:

http://sanetwain.ozuzo.net/

A program letöltése és telepítése rendkívül egyszerű: amennyiben minden beállítás rendben van, azonnal működik. (Amennyiben elrontottunk valamit a kiszolgálói oldalon, vagy a kliensen hibáztunk a beállításokkal és a szkennert nem éri el, akkor a legváltozatosabb hibaüzeneteket produkálja, ilyenkor csak a feladatkezelőből történő kilövése segít.)

A telepítő innen érhető el: `http://sanetwain.ozuzo.net/#download`



3. Ábra: Sanetwain működés közben, magyar nyelven. Látható a szkenner hálózati elérésének megadása.

Adatbázis-kezelés

Az általános definíció szerint az adatbázis-kezelő szoftverek olyan speciális alkalmazások, amelyek adatok tárolására és elemzésére használhatók. Az adatbázis-kezelő rendszerek gyakran hálózatos környezetben többfelhasználós, párhuzamos hozzáférést biztosítanak adatbázisokhoz. Angol megnevezésük: DBMS – Database Management System.

Történeti áttekintés

Már a 20. század legelején mutatkozott igény a népesség nyilvántartási adatok kezelése miatt adatbázisba szervezésre. Valójában azonban csak a 1960-as években alakult ki az a struktúra, amelynek nyomait a ma működő adatbázis-kezelők működési modelljeiben is megtaláljuk. Elsősorban bérszámfejtésre és adatsorok, statisztikák elkészítésére használták a lyukkártyás technológiát. A ma használt legnépszerűbb adatbázis-modell működési logikáját 1970-ben Edgar F. Codd[153] publikálta az IBM munkatársaként. Az első modern adatbázis-kezelőt is az IBM hozta létre System-R néven (ma DB/2 néven ismert). Az első nagyszabású adatbázis-kezelő mégis az Oracle[154] nevéhez fűződik, amelyet ma is széles körben használnak. Fejlesztésének motivációi között többek között a CIA igényei is szerepeltek. A mai SQL alapjait azonban a System-R fejlesztői fektették le, amelyet SEQUEL nyelvnek neveztek el. Ma már sok kereskedelmi termék, köztük számos kiváló nyílt forráskódú rendszer is a felhasználók rendelkezésére áll. A legelterjedtebb szabad szoftverek:

– MariaDB[155]
– MySQL[156]
– PostgreSQL[157]
– SQLite[158]

Rendelkezésre áll még több olyan adatbázis-kezelő is, amely akár bizonyos felhasználószámig, vagy bizonyos nem kereskedelmi esetekben ingyenesen használhatóak, de mivel ezek nem szabad szoftverek, és vagy funkció-, vagy felhasználószám-korlátozással rendelkeznek, nem szerepelnek ebben a felsorolásban. Egy napjainkban üzemeltetett átlagos LAMP (mely a Linux, Apache, MariaDB és PHP stack közös rövidítése) szerveren biztosan telepítve van egy adatbázis-szerver, hiszen nem csak a modern CMS, CRM és egyéb webes technológiák követelik meg a DB meglétét, hanem nagy felhasználó szám esetén az FTP, MAIL és egyéb felhasználói profilokat, jelszavakat is célszerű adatbázisban tartani.

Speciális igények, például nagy számú hálózati hozzáférés ideiglenes adatainak tárolása, vagy rendkívül nagy mennyiségű adat (big data) gyors, párhuzamos feldolgozása speciális adattároló- és kezelő alkalmazásokat hozott létre, amelyek a hagyományos adatbázis-kezelők kiegészítésére, illetve akár kiváltására is használhatók. Az első esetben a meglévő adatbázis-kezelők gyorsítótára-

153. http://hu.wikipedia.org/wiki/Edgar_F._Codd
154. http://hu.wikipedia.org/wiki/Oracle_Corporation
155. https://mariadb.org/
156. http://www.mysql.com/about/legal/licensing/index.html
157. http://www.postgresql.org/
158. http://www.sqlite.org/copyright.html

zó képességeinek teljesítményét is meghaladva tömörebb, csak memóriában történő tárolást, és ezzel több felhasználó egyidejű kiszolgálását teszik lehetővé olyan gyors kulcs-érték kezelők, mint a népszerű nyílt forráskódú memcached vagy Redis. A második speciális esetben a noSQL adatbázis-kezelők és a kapcsolódó map-reduce feldolgozó-rendszerek, mint például a szintén nyílt forráskódú MongoDB és a hozzá kapcsolható Hadoop jöhetnek számításba.

MySQL

A MySQL[159] jelenleg az egyik leginkább ismert és elterjedt adatbázis-motor, amelynek közösségi változata (MySQL Community Edition) szabad szoftver. A jogtulajdonos Oracle nagyvállalati részre elérhetővé teszi zárt forráskóddal, klasszikus licenc vásárlási konstrukcióban a közösségi változat kibővített funkcionalitású variánsát. Feltehetőleg ez a fő oka, hogy a közösség a nyílt licenc alatt hozzáférhető változat alapjain elkezdte fejleszteni a MariaDB nevű adatbázis kezelőt, mely a jelenlegi trendek szerint tempósan elkezdte átvenni a MySQL helyét a szabad operációs rendszer terjesztésekben és számos felhasználónál. De erről majd a következő fejezetben.

A MySQL fejlesztését a svéd MySQL AB kezdte meg kettős licencelés alatt. Lehetősége volt a felhasználóknak GPL szabad szoftver licenc[160] alatti üzleti felhasználásra vagy zárt (tulajdonosi) licenccel egyéb üzleti modell megvalósítására is. A MySQL-t 2008-ban felvásárolta a Sun, amelyet két évvel később az Oracle vásárolt fel. Így mára a MySQL fejlesztését az Oracle végzi, de az egyelőre nyílt forráskódú maradt. Ez egyrészt kedvező, mivel az Oracle egyike az adatbázis-fejlesztő cégek legnagyobbikának, másrészt pedig sok emberben kételyeket ébreszt, hogy az Oracle-nek szüksége van-e egy nyílt forráskódú, konkurens termékre. Mivel azonban a MySQL továbbra is GPL alatt elérhető, így a lehetőség adott egy esetleges kódbezárás utáni a rendszer fejlesztésének szabad folytatására. Jelenleg a legtöbb platform alatt futtatható (Linux és Unix klónok, illetve Windows és Mac OS X alatt is). Üzemeltetési szempontból a kezelése egyszerűen elsajátítható és az elérhető dokumentációk tekintetében is igen versenyképes. Főbb verzióit tekintve hatalmas fejlődésen ment keresztül az elmúlt években és egyre inkább kezdenek nem kis lekérdező motorként, hanem igen komoly feladatok ellátására alkalmas DB-ként gondolni rá. Adminisztrálását számos eszköz könnyíti, többek között van általános célra használható parancssoros felülete, ahol gyakorlatilag majdnem mindent könnyen el lehet érni (az adminisztrátorok inkább ezt szeretik és használják, mivel SSH-n keresztül távolról is könnyen és biztonságosan kezelhető). A szabad szoftver MySQL GUI Tools a funkciók grafikus elérését teszi lehetővé, de ennek fejlesztését az Oracle leállította és a részben szabad MySQL Workbench használatát ajánlják helyette, mindkét eszköz megtalálható a fontosabb Linux terjesztésekben, de elérhetők a MySQL honlapjáról is. Sajnos a legtöbb esetben a phpMyAdmin[161] felületen keresztül adminisztrálják, amely PHP-alapú, nyílt forrású eszköz. Jellemzően a rendszer-üzemeltetők (az adatbázis üzemeltetőkkel szemben) nem kedvelik, mivel a biztonsági hiányosságai és azok befoltozásának gyakorisága és gyorsasága jellemzően problémás, folyamatosan veszélyezteti az őt futtató szerver és az adatbázis biztonságát egyaránt. Ennek ellenére, megfelelő környezetben használva (SSL mögött, Htaccess és egyéb szabályok a hozzáférés korlátozására, nem kitalálható URL stb.) ajánlható és remek eszköz a MySQL DB-adminok számára. Az adatbázisok mentése viszonylag egyszerű és a gyári segédprogramokkal jól megoldható, a legtöbb mentő program képes kezelni natívan, illetve bizonyos esetben lehetőség van MySQL Hotcopy-t csinálni az éppen futó adatbázisról. Az alaptelepítés minden esetben egy

159. http://www.mysql.com/about/legal/licensing/index.html
160. GPL v2, szerkesztési (linking) kivétellel, amely még nagyobb teret enged az üzleti célú felhasználásnak.
161. http://www.phpmyadmin.net/

általános my.conf állománnyal érkezik, amely finomhangolása feltétlen szükséges az adott rendszer várható terhelése és a hardverparaméterek függvényében. A MySQL számos olyan képességgel rendelkezik, ami kiemeli a mezőnyből.[162] A MySQL-hez nagyvállalatoknak szánt megoldások is elérhetők, amely segítségével magas rendelkezésre állású rendszereket lehet építeni. Ilyen megoldás például a MySQL Cluster. Működése pont olyan, mint egy teljes klaszteré, azonban csak az adatbázis-motor van elosztva. Kisvállalati környezetben, illetőleg ahol nincs szükség a teljes funkcionalitás (pl. LAMP) klaszterbe szervezésére, jó terheléselosztó módszer lehet, illetve az adatok redundáns tárolását lehet vele megoldani. A funkció beállítása viszonylag egyszerű, és igazi prémium funkcionalitást biztosít. Nagyvállalati környezetben a MySQL mellett szóló érv, hogy mivel hazánkban is igen elterjedt, sok informatikai vállalkozás képes támogatást nyújtani hozzá.

Telepítése és beállítása

Mivel a MySQL minden ismertebb platformra fordítható és természetesen a legnépszerűbb terjesztések csomagolt részét képezi, ezért igen egyszerű:

```
apt-get install mysql-server mysql-client
```

a függőségek kezelése miatt, az apt-get innentől felrakja a szükséges csomagokat. Természetesen, ha PHP alól is el akarjuk majd érni, akkor szükséges a php5-mysql csomag telepítése is, de erről bővebben olvashatunk a WEB fejezetben.

A telepítő tehát letölti a függőségeket és beállítja egy alap működésre a MySQL-t, kérni fogja az installáció végén a MySQL root jelszavát. Ezt érdemes legalább 10 karakteres véletlen jelszónak beállítani, amelyet majd a MySQL backup során is fel kell hogy használjunk, így kénytelenek leszünk a MySQL gépen tárolni, ezért olyan állományba és könyvtárba rakjuk, ahol csak a root felhasználó olvashatja (igaz nem ez lesz az Achilles-sarka a MySQL rendszerünknek :)

Ezt követően a MySQL készen áll a működésre, azaz csak teljesen alap funkcionalitással. Ha szeretnénk alapesetben egy kicsit biztonságosabbá tenni, akkor erre egy remek lehetőséget kínál a mysql_secure_installation script. Szintén rootként futtatva, interaktívan tudunk pár igen hasznos opciót tiltani vagy engedni: tilthatjuk az anonymous felhasználó hozzáférését, a távoli root hozzáférést, törölhetjük vagy korlátozhatjuk a teszt adatbázishoz való hozzáférést. Ezek azért fontos dolgok, mivel többnyire egy LAMP környezet gyenge pontja (a PHP-n kívül) általánosságban az SQL szokott lenni.

A további konfigurálást a /etc/mysql/my.cnf állományban tudjuk megtenni. A fontosabb opciók:

```
[mysqld]
  user            = mysql
  pid-file        = /var/run/mysqld/mysqld.pid
  socket          = /var/run/mysqld/mysqld.sock
  port            = 3306
  basedir         = /usr
  datadir         = /var/lib/mysql
  tmpdir          = /tmp
  lc_messages_dir = /usr/share/mysql
  bind-address            = 127.0.0.1
```

A fenti alap beállítások megváltoztatására csak indokolt esetben van szükség. Ilyen lehet, ha a nem a standard 3306-os porton akarjuk, hogy válaszoljon, vagy ha DATADIR-t át akarjuk rakni, pl. lemez I/O vagy telítettség miatt. Fontos megjegyezni, hogy a MySQL nagyon rosszul tolerálja,

162. http://hu.wikipedia.org/wiki/MySQL

ha normál üzem közben elfogy alóla a lemezterület. Ilyenkor a táblák és a DB is sérülhetnek, ezért nagyon fontos jól méretezi a DATADIR területet, valamint érdemes folyamatosan egyedi scriptek, vagy nagios, vagy munin segítségével figyelemmel kísérni, nehogy kifusson a MySQL a neki fenntartott lemezterületből. A bind-address segítségével pedig beállíthatjuk, hogy a ne csak a localhoston hallgasson DB motor, hanem pl. a külső lábon is. Ehhez ugyan aztán a DB-n belül is engedélyezni kell. Mivel ez egy jellemző dolog ezért a többi konfiguráció előtt érdemes tárgyalni. Tehát ha egy távoli host részére szeretnénk engedni, hogy egy adott db vagy akár egy adott db táblájához hozzáférjen, akkor a következőket kell tennünk:

a bind-address-t átírjuk a megfelelő IP-címre, pl.

```
bind-address            = 192.168.1.1
```

majd újraindítjuk a MySQL-t: service mysql restart

Ezek után belépünk a mysql -u root -p parancs segítségével. Ha még nem hoztunk létre adatbázist, akkor a legegyszerűbb, ha rögtön úgy hozzuk létre, hogy ahhoz a megfelelő IP-ről hozzá is tudjanak férni:

```
mysql> CREATE DATABASE webmotor;
mysql> GRANT ALL ON webmotor.* TO webuser@'192.168.1.2' IDENTIFIED BY 'jelszo';
mysql>flush privileges;
mysql>quit;
```

jelen esetben tehát a webmotor db-hez a webuser felhasználónak engedtük a 192.168.1.2-ről a hozzáférést. Ezt később bármikor módosíthatjuk a GRANT parancs segítségével. A távoli SQL hozzáférések alapesetben titkosítatlan csatornán utaznak, azaz tökéletesen lehallgathatóak. Éppen ezért, ilyen távoli hozzáférést, csak VPN-en belül, vagy belső hálózaton belül, de mégis leginkább akár ssh port forward, vagy stunnel[163] segítségével csináljunk csak! A távoli management fejezetben mindkettő opció elolvasható.

A továbbiakban a MySQL finomhangolását végezhetjük el:

```
max_connections         = 100
```

alapesetben 100 db párhuzamos csatlakozást enged a szerver felé, természetesen ez akár lehet párhuzamosan 100 db PHP kód, ami ír vagy olvas az SQL-ben. Éppen ezért, nagyon fontos, hogy ezt a paramétert szükség esetén emeljük fel a megfelelő számra. Nagyon gyakori hiba, hogy egy jól méretezett vason a MySQL hatalmas terhelést ad, egyszerűen azért, mert elfogynak a nyitható kapcsolatok és berakja őket egy várakozó sorba, ami az Apache+PHP oldalon is fennakadást okoz. Így előbb-utóbb megeszi az erőforrásokat, a memóriát és a CPU-t is. Egy átlagos és jól méretezett LAMP környezetben, átlagos vasteljesítmény mellett az ideális szám 900.

```
connect_timeout         = 5
wait_timeout            = 600
max_allowed_packet      = 16M
thread_cache_size       = 128
sort_buffer_size        = 4M
bulk_insert_buffer_size = 16M
tmp_table_size          = 32M
max_heap_table_size     = 32M
```

Jellemzően a timeout és cache értékek beállításai megfelelőek szoktak lenni egy teljesen alap rendszernek. Megint csak egy megfelelően megválasztott timeout vagy cache mérettel nagy gyorsulást lehet elérni, de ugyan akkor egy túl rövidre vagy hosszúra választott timeout, szépen le tud-

163. http://shellboy.com/secure-remote-mysql-connection-using-stunnel.html

ja ültetni a rendszert, akár csak a túl kicsi vagy nagy cache. Ezeket szépen fokozatosan érdemes igazítani az aktuális rendszerhez, a programozókkal együttműködve. Érdemes folyamatosan monitorozni egy-egy átállított paraméter hatását pl. a munin segítségével.

```
slow_query_log_file     = /var/log/mysql/mysql-slow.log
long_query_time = 10
```

a slow query log beállítása egy befektetés a jövőbe, hiszen utólag itt lehet majd visszakeresni azokat a lekérdezéseket, amelyek valamiért akár percekig, vagy órákig futottak. Itt megint nagy segítséget jelenthet a Munin Slow-Query mrtg diagramja, ahol látható hogy volt vagy nem volt ilyen. Utána pedig oda lehet adni a programozónak a logot elemzésre. Ezen kívül természetesen a MySQL teljes és részleges log rögzítésre is képes, amely szintén megtalálható az alap konfigurációban, viszont ezzel nagyon csínján bánjunk, ugyan is nagyon hamar a telítődhet a teljes log partíció, ha nem csak adott periódusra kapcsoljuk be a teljes logot.

A MySQL mentése

Értelemszerűen az installálás utolsó lépcsőjében érdemes egy alap mentést azonnal beállítani, amit aztán lehet bővíteni, vagy mentési rendszerbe kötni. A MySQL beépített mentési mechanizmusát felhasználva a következőképpen lehet a legegyszerűbben lementeni a DB-t:

```
/usr/bin/mysqldump  -prootjelszo webdb | /bin/gzip > /backup/mysql-webdb-$1.tgz
/usr/bin/mysqldump -prootjelszo information_schema | /bin/gzip >
/backup/mysql-information_schema-$1.tgz
/usr/bin/mysqldump --all-databases -prootjelszo | /bin/gzip > /backup/mysql-
  ALLDATABASE-$1.tgz
```

Mint az látható, az első két sor adott adatbázisokat ment, amelyeket a jelszó után nevesítettünk, a harmadik sor pedig az összes adatbázist menti egyben. A mentő sorok gzip tömörítést is alkalmaznak, mivel a DB állományok akár 80-90%-os méret csökkenéssel is tárolhatóak, hiszen azok jellemzően sima text állományok. A fenti példában azért van külön és egyben is mentve az összes használt DB, mivel egy esetleges részleges adat vesztés vagy korrupció esetén, nagyon nagy segítség, ha nem a teljes mentésből kell visszaállítani 1 táblát vagy 1 DB-t. Viszont fontos, hogy egy teljes dump is rendelkezésre álljon. Fontos továbbá az information_schema DB mentése is, mint az látható, hiszen itt tárolódnak a DB közötti hozzáférések és egyéb fontos információk. A teljes mentés természetesen ezt tartalmazza. A fenti sorokat érdemes tehát egy héjprogramba belemásolni, és a megfelelő időben, amely jellemzően hajnalban üresjárati időben van, futtatni. Akár napi, heti, havi rotációs tárolással.

Egy alternatív mentési lehetőség, amikor a MysqlHotCopy[164] Perl script gyűjteményt használjuk. Jellemzően ezt a módszert nem az előzőekben leírt mysqldump helyett, hanem annak kiegészítésére, akár napközbeni vagy tesztelés előtti gyors DB-mentésre használhatjuk bizonyos megkötésekkel. Pl. csak MyISAM DB és Archive db típusok esetében használható.

Természetesen a komplexebb mentő eszközök kezelik a MySQL-t, de erről a Backup fejezetben olvashatunk.

164. http://dev.mysql.com/doc/refman/5.0/en/mysqlhotcopy.html

Referenciák

Néhány fontosabb referencia, ahol a MySQL-t megelégedéssel használják[165]:

– WhiteHouse.Gov
– NASA
– Department of Homeland Security
– Swedish National Police
– Ministère de la Justice (France)
– Bank of Finland
– Swedish National Police
– US Navy

MariaDB

A MariaDB[166] a MySQL forráskódjából kiindult fejlesztés, mely jelenleg a MySQL-lel teljes mértékben kompatibilis, közösség által fejlesztett és támogatott adatbázis szerver. A rendszer egyben kiválóan szemlélteti a szabad szoftver licencelésben rejlő hatalmas lehetőséget is. A MariaDB-t az Oracle üzleti célokat szem előtt tartó lépései miatt aggódó közösség hozta létre a GPL licenc előírásait betartva, a MySQL forráskódjának továbbfejlesztésével. A legtöbb platformra elérhető binárissal vagy forráskóddal rendelkezik. Több klienssel használható, és rendelkezik a saját fejlesztésű MariaDB Client Library for Java Applicationsszel is. 2012 decemberében egy alapítványt is létrehoztak, amelynek célja a fejlesztés felügyelete, valamint annak hátterének megteremtése. Jelenleg azonban nem minden Linux terjesztésnek része (bár ez 2013 elején gyorsan változik, szinte minden fontosabb terjesztés átállt rá MySQL-ről), viszont minden lényeges terjesztéshez elérhető, binárisan formában lévő programcsomagok szintjén (pl. Debian és Ubuntu rendszerekhez apt.source szinten is). Minden jel arra mutat, hogy **a MariaDB a jövő szabad szoftveres adatbázis-kezelője**, hiszen az Oracle MySQL fejlesztése nagyon könnyen vehet nem várt fordulatot, így a felhasználók jelentős része tart a MySQL-től.

A MariaDB installálása majdnem pontosan megegyezik a MySQL beállításaival, csupán jelenleg nem alap része a terjesztések szoftver kínálatának. Ezért első közelítésben szükséges felvenni a MariaDB csomag tárolóit[167] a rendszerbe:

```
sudo apt-get install python-software-properties
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0xcbcb082a1bb943db
sudo add-apt-repository 'deb
  http://ftp.heanet.ie/mirrors/mariadb/repo/5.5/ubuntu precise main'
```

Innentől pedig már csak telepíteni kell:

```
sudo apt-get update
sudo apt-get install mariadb-server
```

165. http://www.mysql.com/customers/
166. https://mariadb.org/
167. https://downloads.mariadb.org/mariadb/repositories/

A továbbiakban pontosan ugyanolyan parancsok és lehetőségek segítségével állíthatjuk be, mint a MySQL esetében.

Referenciák

Számos ismert sikertörténetről olvashatunk a MariaDB honlapján[168], ezek közül néhány nagyobb, érdekesebb:

– A Wikipédia angol és német kiadásai, valamint a WikiData 2013. április 22-én befejezte a MariaDB-re való átállást[169], melyet azóta is elégedetten használnak. A rendszer terhelésére jellemző, hogy az intenzív gyorstárazás ellenére (melyet memcached és Redis segítségével valósítottak meg) az angol Wikipédia csúcsterhelése 50k lekérdezés másodpercenként.
– A 150 millió felhasználós Nimbuzz háttérrendszerén átállt MariaDB-re[170].
– Az OLX, mely napi 40 millió lapot szolgál ki, szintén MariaDB-t használ[171].

PostgreSQL

A PostgreSQL[172] fejlesztését az 1970-as években kezdték meg a kaliforniai Berkeley egyetemen Ingres néven. A 80-as évek közepétől több különböző kereskedelmi termék (mint a Sybase vagy a Microsoft SQL Server), és akkor még Postgres (Post Ingres) néven ingyenes verzióként élt tovább a projekt. 1996-ban kapta a manapság használt PostgreSQL nevet (a 6.0-s verziótól). Alapvetően egy olyan adatbázis-kezelő rendszer, amely szabványos SQL-felületen vezérelhető, azonban a tudása jóval komplexebb a függvényrendszernek köszönhetően. A függvények megvalósítása több nyelven is lehetséges (pl. Java, Perl, Python, Ruby, Tcl, C/C++, PL/pgSQL). További előnyei a triggerek[173] definiálása, a tranzakciókezelés, a komplex adattípusok használhatósága, a sémák és a replikáció. Természetesen rendelkezik a magyar nyelv teljes körű támogatásával DB szinten. Komplexitásának és tudásának köszönhetően gyakran választják nagy teherbírású és magas rendelkezésre állást igénylő feladatokhoz. A PostgreSQL is szabad szoftver, amelyet a MIT-stílusú (a módosított változatok zárt licencelését is megengedő) PostgreSQL[174] licenc alatt tesznek közzé.

Jelenleg a következő, igencsak meggyőző korlátokkal rendelkezik:

– maximális adatbázisméret korlátlan
– maximális táblaméret 32 TB
– maximális sorméret 1,6 TB
– maximális mezőméret 1 GB
– sorok maximális száma táblánként korlátlan
– oszlopok maximális száma táblánként 250–1600, oszloptípustól függően

168. https://kb.askmonty.org/en/mariadb-case-studies/
169. http://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb/
170. https://kb.askmonty.org/en/nimbuzz-uses-mariadb/
171. https://kb.askmonty.org/en/olx_serves_40_million_page_views_daily_MariaDB/
172. http://www.postgresql.org/
173. http://hu.wikipedia.org/wiki/Trigger_%28adatb%C3%A1zisok%29
174. http://opensource.org/licenses/postgresql

– indexek maximális száma táblánként korlátlan

Rendelkezik a természetesnek számító parancssoros kezelést lehetővé tevő eszközzel, és számtalan támogatott és igen magas szinten elkészített GUI segédprogrammal[175] is, amelyek megkönnyítik az adatbázis-adminisztrátorok munkáját. Az egyik ilyen igen népszerű segédprogram a phpPgAdmin[176], amely akárcsak a MySQL esetében a phpMyAdmin, egy PHP alapú GUI. Érdemes megnézni a tudásmátrixot[177], amely jól szemlélteti, hogy az elmúlt években hatalmas fejlődésen ment keresztül, amelyet folyamatosan honorálnak is a különféle díjakkal[178]. A kételkedőknek pedig érdemes átfutni a nagyobb felhasználók[179] listáját tartalmazó weboldalt, ahol meggyőződhetünk arról, hogy mekkora cégek döntöttek a PostgreSQL mellett. Természetesen rendelkezik az összes ilyenkor elvárható support és információs felülettel, Wiki-oldallal, célzott fórumrendszerrel. Inkább érdekesség, mint fontos információ, de a PostgreSQL team kizárólag Debian GNU/Linuxot használ a saját[180] projektjén belül. Számos európai cég vállal 7×24-es támogatást a PostgreSQL-re. Érdemes megemlíteni a számos gyári kiterjesztés közül a PostGIS[181] kiterjesztést, amely kifejezetten földrajzi adatok kezeléséhez nyújt hathatós segítséget. Gyakorlati tapasztalat, hogy a PostreSQL a komplexebb Oracle-szintű DB kiváltására kiválóan alkalmas.

Referenciák

A rendszer honlapján számos komoly referencia van feltüntetve[182], ezek közül néhány fontosabb:

– U.S. Department of Labor
– U.S. General Services Administration
– U.S. State Department
– National Physical Laboratory of India
– United Nations Industrial Development Organisation
– City of Garden Grove, California
– BASF, Agricultural Product Division
– IMDB.com
– Creative Commons
– Greenpeace
– Fujitsu
– Cisco

175. http://wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools
176. http://sourceforge.net/projects/phppgadmin/
177. http://www.postgresql.org/about/featurematrix/
178. http://www.postgresql.org/about/awards/
179. http://www.postgresql.org/about/users/
180. http://www.postgresql.org/about/servers/
181. http://www.postgresql.org/about/news/1387/
182. http://www.postgresql.org/about/users/

SQLite

Az SQLite[183] egy igen széles körben használt (a többi felsorolt DB-hez képest apró) adatbázis-kezelő függvénykönyvtár, mely egyszerű kezelését egy parancssori adminisztrációs eszköz és egy Firefox kiegészítő (SQLite Manager) segíti. Egyéni felhasználásra szánt (bár helyben párhuzamosan több folyamat és szál is használhatja), nem biztosít hálózati hozzáférést az egyetlen állományként tárolt adatbázisaihoz. Jellemzően kliens oldali szoftverekbe építik a rendszer adatainak és a gyorsító tárnak egyszerű kezelésére. Jó példa erre a Firefox[184] böngésző és Thunderbird[185] levelezőkliens, ahol az elsődleges adatok tárolására használják. Számos más cég építi bele[186] zárt vagy szabad szoftveres termékébe. Felhasználási feltételei nincsenek[187], hivatalosan közkincs (public domain)[188], amely szabad kezet ad a felhasználás tekintetében. Tudása folyamatosan fejlődik,[189] és méreteihez képest igen figyelemreméltó. Az adatbázis fájlok kényelmes elérésére használható még az Sqliteman[190], az SQLite Studio[191] és a (PHP-ban íródott) SQLiteManager[192].

Számos további adatbázis-kezelő megoldás létezik szabad szoftverként, azonban egy tipikus vállalati környezetben (ide nem értve az olyan felhasználási területet, ahol párhuzamosan több ezer, százezer, vagy millió felhasználó férhet hozzá egy időben ugyanahhoz az adatbázishoz, mint pl. a Google vagy a Facebook esetében) jellemzően a több főkiszolgálós replikációt[193] tudó adatbázis-motorokat használjuk.

Referenciák

Meglepő, hogy az SQLite teljesen észrevétlenül mennyire részévé vált a mindennapjainknak. A projekt honlapján található fontosabb referenciák közül[194] kiemelünk néhány fontosabbat:

– Mozilla Firefox, Thunderbird, FirefoxOS - ezek az óriási felhasználói bázissal rendelkező programok és rendszerek a belső adataik tárolására használják az SQLite-ot.
– A Google számos helyen használja az SQLite-ot. A leginkább ismert felhasználási terület az Android rendszer, de ez végzi Google egyre népszerűbb böngészőjének, a Chrome-nak az adattárolását is.
– Az Apple is használja a rendszert a MacOS X, Apple Mail, Safari, iTunes szoftverekben, valamint az iPhone rendszerben.
– A széles körben használt Dropbox adatmentő és szinkronizáló szolgáltatás kliens oldalán ebben tárolják az adatokat.

183. http://www.sqlite.org/
184. http://www.mozilla.org
185. https://www.mozilla.org/hu/thunderbird/
186. http://www.sqlite.org/famous.html
187. http://www.sqlite.org/copyright.html
188. http://en.wikipedia.org/wiki/Public_Domain
189. http://www.sqlite.org/features.html
190. http://sqliteman.com
191. http://sqlitestudio.pl/index.rvt
192. http://www.sqlitemanager.org/
193. http://hu.wikipedia.org/wiki/Multi-master_replication
194. http://www.sqlite.org/famous.html

Hogyan válasszunk?

Felmerül a kérdés, hogy a felsoroltak közül melyik az, amelyik egy adott feladatra megfelelő. Ha hálózaton is elérhető, többfelhasználós adatbázis kezelőre van szükség, akkor a kérdés a PostgreSQL és a MariaDB között fog eldőlni. Általánosságban elmondható, hogy mindkét rendszer kiválóan használható általános feladatokra. De a rendszer használatát jól ismerők nagyobb száma (korábbi MySQL DBA-k), elterjedtsége (a MySQL-t is beleértve), illetve népszerűségének gyors növekedése miatt a MariaDB[195] a legjobb választás. Ha mindenképpen szeretnénk nagyvállalati rutinnal licencvásárlással egybekötni a támogatási igényünket, akkor javasolt a MySQL nagyvállalatoknak szánt változatai közül választani. A legjobb tanács, ami adható, hogy a feladathoz válasszunk eszközt, és ne fordítva. Azaz, nézzük meg, hogy közép távon mire van szükségünk, mekkora adatmennyiség, milyen adatbázis-felépítés, milyen típusú adatbázis-kiterhelés várható, és ezt az ajánlásokkal és a két projekt weboldalán feltüntetett tudásmátrixszal összevetve döntsünk. Fontos további szempont, hogy a MariaDB azt az utat járja, amit előtte a MySQL sok sok éven át, mielőtt felvásárolták, így aki biztos szeretne lenni abban, hogy még sokáig a közösség támogatását fogja élvezni, annak érdemes megfontolni a MariaDB-re való váltást.

195. http://db-engines.com/en/system/MariaDB%3BPostgreSQL

Webproxy

Proxynak[196] nevezzük azokat a szoftvereket, amelyek a kliensek kéréseit köztes elemként más szerverekhez továbbítják. A kliens alkalmazás csatlakozik a proxy szerverhez, akár transzparenciával (anélkül, hogy tudná, hogy egy proxy szolgáltatásait veszi igénybe – átlátszó), akár a kliens szoftverében rögzített módon, majd onnan egy erőforrást, vagy szolgáltatást igényel, amely egy vagy több másik szerveren található. Lényegében a proxy egy erőforrást vagy szolgáltatást oszt el. A proxy természetesen a rajta keresztüláramló adatokat és kapcsolatokat szabadon, a beállításai szerint módosíthatja. Azaz elemzés után a lokális tárolójából szolgálhatja ki a klienst anélkül, hogy a szerverhez fordult volna, vagy a kérésben és/vagy a válaszban megváltoztatott volna bármit. Sokszor a proxy célja a gyorsabb és erőforrás-kímélő felhasználás, de lehet korlátozás, megfigyelés alapeszköze is. A proxy legszélesebb körben a HTTP kiszolgálásra alkalmas eszközként terjedt el az 1990-es és a 2000-es évek elején, főként a modemes internetes időkben. Manapság azonban a legtöbb magasabb szintű tűzfal alapvető része több fajta proxy (elemző megoldás), illetve használjuk tunnelezésre, vagy erőforrások elosztására is. *Másik elterjedt felhasználása, hogy az illegális szervereket rejtik el proxy szerverek mögé, főként a torrent világban. Így biztosítva azt, hogy ha az egyik vagy másik proxy kiszolgálót a hatóság le is kapcsolja, akkor nem a központi (core) modulhoz (adatbázis stb.) jut el.* Ugyanilyen felhasználási módja a TOR[197] project megoldása, ahol szintén proxy szervereken keresztül juthatunk el az internetre anonim módon (igaz, a működési elvéből adódóan a mi internet hozzáférésünkön keresztül mások is forgalmazhatnak tartalmat szintén anonim módon). A proxyk tehát széles körben használatosak a szerveroldali felhasználás számtalan területén, és sok esetben egy tűzfal mögött, vagy egyéni alkalmazásokat használva nincs is közvetlen tudomásunk róluk.

A HTTP proxy

Többgépes hálózat esetén szinte kikerülhetetlen tényező volt az 1990–2000-es évek derekán, amikor is a legtöbb cég vagy betárcsázós analóg vagy ISDN vonalon osztozott, de később a bérelt vonalak feltűnésével is maradt a legtöbb kisvállalkozás életében a web proxy. Tipikusan jellemző viselkedés minta, hogy az egy területen dolgozó kollégák szabadidejükben ugyanazon oldalakat látogatják. A megfelelően beállított proxy csak a változásokat figyelve, mindig csak aktualizálva a saját lokális gyorsítótárában tárolt tartalmat szolgálja ki a belső hálózatot. Így előfordulhat az, hogy bár 20 kolléga nézi meg ugyanazon híroldal nyitólapját, amelynek mérete felhasználói letöltésenként kb. 5 MB lenne, azonban mégsem 20×5 MB a bejövő forgalom, mivel a proxy lokálisan szolgál ki 19 felhasználót, és csak az első felhasználó tölti le teljesen a tartalmat, ezzel is kímélve a sávszélességet. Az internetszolgáltatók régebben hatalmas proxy szervereket tartottak fenn, ugyanezen okokból, csak ott a belső hálózatot a betárcsázós modemes és ISDN/bérelt vonali felhasználók alkották, amíg a külső, spórolni kívánt vonal a nagyon drága nemzetközi sávszélesség volt, amely a régebbi időkben műholdas és mikrohullámú rendszerek segítségével jutott el többnyire hazánkba. Egy ilyen proxy rengeteg pénzt tudott megspórolni, és a felhasználói élményt is nagyban tudta javítani, hiszen a gyakran látogatott tartalmak szinte gyorsabban jöttek be mint a közelebbi, kevésbé látogatott itthoniak. Kezdetben számos zárt webproxy termék uralta a piacot,

196. http://en.wikipedia.org/wiki/Proxy_server#Forward_proxies
197. https://www.torproject.org/

olyan cégek termékei mint a Netscape vagy a Cisco. A szabad szoftverek között a legelterjedtebb mind a mai napig a Squid proxy[198].

Squid

Alapjai az 1996-ban lezárult Harvest projekt keretein belül készültek el. Természetesen az elmúlt több mint 1,5 évtized alatt a kódja teljesen átalakult. A szoftver fejlesztésért felelős Duane Wessels az NLANR[199] munkatársa fogja össze a többi fejlesztőt. Alapja egyetlen blokkolásmentes I/O alapon vezérelt folyamat, ez végzi el a feladatok nagy részét. Sokszor ez is jelenti a legnagyobb problémát, pl. a memóriafoglalások tekintetében. Elterjedésének alapja, hogy funkcionalitása a kezdetek óta olyan bő, hogy a legtöbb zárt termék színvonalával nem csak felveszi a versenyt, hanem sok esetben meg is haladja azokat. Ennek köszönhető az is, hogy számtalan kereskedelmi dobozos termék, vagy eszköz felhasználja a Squidet.

A teljes körű szolgáltatási palettán olyan dolgok szerepelnek, mint: HTTP és FTP protokollok teljes körű kiszolgálása, az SSL kapcsolatok kezelése és gyorsítása, DNS cache, transzparens működési lehetőség. Részletesebb szakmai ismeret nélkül konfigurálható a tartalomszabályozás, és olyan népszerű gyorsítótárazási protokollok[200] széles körét támogatja, mint az ICP, HTCP, CARP és WCCP. Egyszóval egy igazi szabad szoftveres sikertörténet, főként a tudása és a rendkívül széles körű skálázhatósága miatt: ugyanúgy használható egy 10-20 fős iroda kiszolgálójaként, mint egy 100 ezer ügyféllel rendelkező ISP[201] szervereként – természetesen más beállításokkal és merőben más hardverelemekkel. A Squidre ugyan úgy jellemző, mint a Postfixre és az SSH-ra is (hogy csak két példát ragadjunk ki), hogy a tényleges tudásának csak a felszínét kapargatjuk átlagos kis- és középvállalati üzemeltetés során.

A Squid telepítése

```
sudo apt-get install squid3
```

Ahogy az már megszokott, ez a parancs a függőségekkel együtt letölti a tárolókból, telepíti azt, majd alapbeállítás szerint létrehozza a cache dir környezetet és el is indítja a Squidet. Mivel az alapbeállítások nem feltétlenül felelnek meg nekünk, ezért a legjobb, ha telepítés után rögtön le is állítjuk:

```
sudo service squid3 stop
```

A konfigurációs állomány az /etc/squid3/squid.conf útvonalon található.

Nézzük tehát a squid.conf legfontosabb lehetőségeit:

```
http_port 3128
```

Definiálhatjuk, hogy a Squid melyik TCP porton fog hallgatni. A felhasználók ide fognak csatlakozni a klienseikkel (böngésző stb.). Jellemzően érdemes 3128-as alapbeállításon hagyni.

```
hierarchy_stoplist cgi-bin \?
```

198. http://www.squid-cache.org/
199. http://www.nlanr.net/
200. http://en.wikipedia.org/wiki/Web_Cache_Communication_Protocol
201. http://en.wikipedia.org/wiki/Internet_service_provider

```
  acl QUERY urlpath_regex cgi-bin \?
  cache deny QUERY
```

Jelentése: A dinamikus tartalmat nem gyorsítótárazzuk.

```
access_log /var/log/squid/access.log squid
```

A naplót itt fogjuk tárolni, ezért nagy terheltségű proxy esetében figyeljünk arra, hogy bár a logrotate fogja ezeket a naplókat is kezelni, de azzal együtt se nőhessenek túl nagyra.

```
cache_mem 256 MB
```

A Squid által használt memóriaterület nagyságát határozhatjuk meg. Érdemes itt[202] tájékozódni arról, hogy mennyit érdemes beállítani. Minden rendszer egyedi, és főképpen a felhasználók számától és viselkedésüktől függ. Az úgynevezett „hot object" mérettől is és a maximálisan tárolható objektum méretétől is függ, amely a következő opcióval állítható. Azaz képzeljünk el egy olyan hálózatot, ahol vegyesen Ubuntu kliensek és Windows kliensek használják a proxy szervert. Jellemzően az Ubuntu frissítések 0,5-5 MB méretűek, esetenként egy kernel 30-50 MB méretű is lehet. Azaz legalább ennek érdemes belefférnie a memóriába, hiszen ahogy a kliensek felkapcsolódnak a hálózatra, majdnem egyszerre fognak frissíteni. Windows frissítéseknél ugyan így kell ügyelni a méretezése, azaz:

```
maximum_object_size 128 MB
```

Ide azt a legnagyobb objektumméretet írjuk be, amely jellemző a környezetünkre.

Fontos az is, hogy a maximum_object_size beleférjen a cache_mem területbe is. A Squid a cache_mem terület sokszorosát is lefoglalhatja terhelés alatt. Jellemzően egy gyorsítótár-szerverbe legalább 4 GB memóriát érdemes beépíteni már 10-15 felhasználó esetében is, vagy ha van rá lehetőség, akkor ennél akár többet is, valamint ezzel párhuzamosan lehet növelni a cache_mem méretét is.

```
cache_dir ufs[203] /var/spool/squid3 1024 16 256
```

Ezzel a beállítással megadjuk, hogy hol helyezze el az átmeneti tárolókat, valamint sorban a gyorsítótár mérete MB-ban, a könyvtárak és az alkönyvtárak száma. Már egy alap proxy esetében is fontos, hogy a napló és a lemezgyorsítótár jól elkülönüljön, lehetőleg más partíción, még inkább más lemezen legyenek tárolva, hiszen ha ugyanabban az időben fogjuk a naplót írni és a tárolt adatot írni/olvasni, akkor a diszk olvasófejének mozgatása két távoli rész között fölöslegesen lassítana. Nagyobb rendszerek esetében érdemes egy külön vezérlőn RAID0-ba összefűzni egy nagyobb partíciót, hiszen itt az adatvesztés maximum átmeneti fennakadást okozhat csak, amíg újra megtelnek a tárolók. Érdemes továbbá a „noatime" mount-opciót használni a külön lemezgyorsítótárat tartalmazó fájlrendszer csatolásához, ezzel is nagyban gyorsítva a hozzáférést. További jó megoldás hagyományos merevlemezek esetében, ha minél gyorsabb (ma: 15k-val pörgő) lemezeket választunk és a fájlrendszer foglaltságát 50% közelébe kalibráljuk be. Ugyanakkor a folyamatosan egyre megfizethetőbb SSD-tárolók térnyerésével érdemes elgondolkodni egy sima SSD cache_dir megoldáson, hiszen ezt közel memória sebességgel fogja a rendszer elérni. Természetesen egy SSD Squid cache-ként való használata erősen csökkenti az SSD élettartamát, hiszen folyamatos írásnak/olvasásnak van kitéve, így az egész rendszert nem érdemes SSD-re telepíteni, hanem kifejezetten csak a lemezgyorsítótár részt. Valamint az SSD állapotát folyamatosan monitorozni kell, például bekötve Nagios/Icinga alá, vagy smartmoontols-szal[204] figyelni. A fájlrendszer elkészítése során (mkfs) egy normál nem túl nagy cég kiszolgálásra alkalmas proxy beállítása ese-

202. http://wiki.squid-cache.org/SquidFaq/SquidMemory
203. http://www.squid-cache.org/Doc/config/cache_dir/
204. http://sourceforge.net/apps/trac/smartmontools/wiki

tén használhatjuk a default paramétereket, azonban ha nagy terheltségű gyorsítót akarunk üzemeltetni akkor érdemes utánaolvasni a blokkméretnek és úgy hozzuk létre az FS-t, hiszen jellemzően a Squid állományai pár KB-os méretűek.

```
acl[205] manager proto cache_object
acl localhost src 127.0.0.1/32 ::1
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32 ::1
acl SSL_ports port 443          # https
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access deny all
```

A fenti beállításokkal egyben határozzuk meg egyrészt a belső hálózatunkat (jelen példában csak a localhost van engedve), illetve határozzuk meg azokat a portokat, amelyeket biztonságosnak ítélünk. Majd az első szabályok segítségével mindent engedünk a localhost-ról, minden Safe portot engedünk és végül minden mást kizárunk. Itt sorolhatjuk fel az alhálózatainkat, úgy mint:

```
acl localnet src 192.168.2.0/255.255.255.0
```

De ebben az esetben ne feledjük a http_access allow localnet segítségével azt engedni:

```
http_access allow localnet
```

Roppant fontos, hogy a belső proxy, ha van külső lába semmiképpen se látszódjon, vagy engedjen forgalmat kívülről, ugyanis előbb vagy utóbb a szemfüles, direkt ilyen hiányosságokat, azaz kilátszó anonymous proxykat kereső emberek vagy robotok ezt meg fogják találni és onnantól gyakorlatilag a nevünkben indíthatnak kéréseket a nagyvilág felé.

A Squid3 összekapcsolható LDAP-ból való azonosítással vagy Active Directory-val[206] is, így a már meglévő felhasználó-azonosítási rendszerbe is integrálható.

Transzparens üzemmódban

Ha nem szeretnénk, hogy a felhasználó maga dönthessen, hogy használja-e a proxyt, vagy sem, akkor kikényszeríthetjük annak használatát a helyi tűzfal és egy proxy szerver segítségével. A már beállított paramétereken kívül a Squid3-ban igazán csak a következő opcióra van szükség ahhoz, hogy átlátszó elemként beépüljön a hálózatunkba:

```
http_port 3128 transparent
```

Természetesen azért a tűzfalat módosítani kell, hogy a HTTP és az FTP forgalom átmenjen a proxy gépünkön. Egy aktuális LTS Ubuntuhoz készített leírást találhatunk itt[207], de álljon itt egy

205. http://wiki.squid-cache.org/SquidFaq/SquidAcl
206. http://wiki.squid-cache.org/ConfigExamples/Authenticate/WindowsActiveDirectory
207. http://ubuntuserverguide.com/2012/06/how-to-setup-squid3-as-transparent-proxy-on-ubuntu-server-12-04.html

egyszerű példa amelyet a tűzfal gépünkön kell beállítanunk, hogy a forgalom átmenjen a proxy gépen:

```
iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j DNAT --to-
  destination 192.168.1.1:3128
iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 80 -j REDIRECT --to-
  ports 3128
```

Jelen esetben a Proxy gép IP címe a 192.168.1.1 és az eth0 a WAN felöli oldal.

A transzparens proxy beállításával lehetőség nyílik arra, hogy a teljes, titkosítatlan közegben történő felhasználói HTTP forgalmat monitorozzuk, majd elemezzük a Sarg[208] segítségével. Ez sok esetben célravezetőbb, mint tiltani és arra sarkalni a felhasználót, hogy keressen kiutat. Természetesen figyelembe kell venni a hazai törvényi környezetet, azaz a felhasználóval megfelelő módon közölni kell, hogy megfigyelés alatt áll.

Proxy beállítások központosítása:

A böngészők támogatnak egy úgynevezett automatikus proxy beállítást (Proxy AutoConfiguration – PAC). Ezt abban az esetben érdemes használni, ha nem tudunk vagy nem akarunk transzparens proxyt beállítani, de nem szeretnénk a klienseken az esetleges proxy IP-cím, port stb. változásokat kézzel állítgatni. Az ilyen esetekben egyszer kell csak a kliens böngészőben az automatikus proxy beállítást kiválasztani, majd megadni a proxy beállító szkript URL-jét. A hálózaton elhelyezett (HTTP- vagy HTTPS-protokollon elérhető) PAC[209] állományból olvassa majd fel a böngésző, hogy hol található a proxy szerver. Valamint a PAC[210] szkriptben azt is ki lehet kötni, hogy amennyiben a proxy szerver nem elérhető, akkor direktben menjenek ki az internetre a gépek.

Weboldalak blokkolása

Gyakori igény, hogy az ismertebb „sávszélességrabló” oldalakat szükség esetén blokkoljuk, erre a Squid többféle megoldást is kínál. A legegyszerűbb egy acl létrehozása és kitiltása:

```
acl block_site dstdomain "/etc/squid3/blocked-sites"
http_access deny block_site
```

Értelemszerűen a /etc/squid3/blocked-sites állomány tartalmazza a blokkolni kívánt oldalak listáját.

Ennél kifinomultabb megoldás kínál a **squidguard**[211], amely beépülő modulként hívható a Squid-ből, természetesen telepítés után:

```
apt-get install squidguard
```

208. http://sarg.sourceforge.net/
209. http://wiki.squid-cache.org/SquidFaq/ConfiguringBrowsers
210. http://en.wikipedia.org/wiki/Proxy_auto-config
211. http://www.squidguard.org/

Tudása messze túlmutat az egyszerű blokkolási lehetőségeken[212], hiszen adatbázisból dolgozik, így eleve mások által készített feketelistákat is felhasználhatunk[213]. A Squid-del való integrációja igen egyszerű, a Squid konfigurációját a következő sorral kell kiegészíteni:

```
redirect_program /usr/sbin/squidGuard -c /etc/squid3/squidGuard.conf
```

Majd a Squid újraindítása után használatba vehetjük:

```
sudo service squid restart
```

Az /etc/squid3/squidGuard.conf fájlban beállíthatunk mások által definiált adatbázisokat, kategóriákat is létrehozhatunk:

```
  dbhome /usr/local/squidGuard/db
  logdir /usr/local/squidGuard/log

  dest gambling {
          log             gambling
          domainlist      gambling/domains
          urllist         gambling/urls
  }

  dest warez {
          log             warez
          domainlist      warez/domains
          urllist         warez/urls
  }

  acl {
        default {
              pass !gambling !warez all
              redirect 302:http://gnu.hu
        }
  }
```

Így ha a példában szereplő gambling/domains, gambling/urls, és warez/domains és warez/urls fájlok megfellelően ki vannak töltve tartomány- vagy számítógépnevekkel, illetve URL-ekkel, akkor pl. a szerencsejátékos és a warezoldalak egy jelentős részét tudjuk blokkolni, vagy átirányítani akár egy meglévő külső akár egy saját hálózatban lévő belső weboldalra, amely természetesen lehet egy figyelmeztetés is.

Sarg[214]

Ez az igen hasznos alkalmazás kiegészíti a Squid naplózási képességeit. Természetesen a proxy szerver naplóállománya igen részletes, így nagy felhasználói szám mellett még a gyakorlott napló-

212. http://www.squidguard.org/about.html
213. http://www.squidguard.org/about.html
214. http://sarg.sourceforge.net/

elemző számára is tökéletesen átláthatatlan. A Sarg ebben nyújt segítséget, mivel grafikusan elemzi számunkra a felhasználók szokásait. Használata és telepítése egyszerű, felépítése és konfigurációs állománya nagyon hasonlít a Webalizerhez[215]. Praktikusan éjszaka lefut, és az aznapi tevékenységet grafikonok és táblázatok segítségével elhelyezi egy kiértékelés könyvtárban, amelyet aztán akár weblapszerűen (webszerver szükséges) vagy lokálisan (a tartalmat saját gépre átmásolva, pl időzített scp, stb) egy böngészőből kiértékelhetünk különböző rendezési elvek alapján. Nagyon könnyű összegezni vele a legnagyobb letöltőt, illetve mérni vele az esetleges nem tiltott, de nem ajánlott tartalmakon eltöltött időt. Éppen ezért komoly visszaélésekre is lehet használni, hiszen a felhasználók teljes napi/heti/havi weben töltött naplóját könnyen értelmezhetően mutatja. Tehát kezeljünk szenzitív adatként, és mindenképpen védjük/tároljuk megfelelően a kiértékelt adatokat.

215. http://en.wikipedia.org/wiki/Webalizer

VPN beállítás: IPSec és OpenVPN

Ha már eljutottunk oda, hogy az SSH alagút és az egyéb megoldások nem kínálnak kellő komfortot, vagy nem jól központosíthatóak, vagy egész egyszerűen csak nem mindenki számára jelentenek könnyen kezelhető megoldást, akkor érdemes megfontolni egy olyan megoldást, amikor a teljes hálózati kommunikáció egy titkosított magánhálózat kiépítésével jár együtt. A VPN nem csak akkor jelent megoldást, amikor két céges alhálózatot akarunk összekapcsolni, bár valójában erre tervezték. Akkor is kiválóan használható, ha az otthoni gépünkkel akarunk rákapcsolódni egy céges hálózatra, illetve bizonyos esetben szerver és szerver közötti kommunikációra is használható, de arra talán nem a legideálisabb megoldás. A fő előnye, hogy az alhálózatokat típusától függően erős titkosítással biztosítja, valamint minden, az adott kapcsolaton átmenő forgalom az átjáró beállításainak függvényében a titkosított közegben utazik. A manapság használatos VPN kliensek olyan egyszerűen kezelhetőek, hogy egy átlagos tudású felhasználónak sem okoz gondot a kezelése, természetesen ha megfelelően automatizálta a rendszergazda a felépülési folyamatot. Segítségével nem csak a szervereinket érhetjük el, hanem az irodában lévő bekapcsolva hagyott klienseket is (például egy távoli asztal szoftver segítségével). VPN megoldásokból számtalan létezik, mint ahogyan a VPN-protokoll használata alapján is sokrétűek (PPP, PPTP, L2TP, IPsec)[216]. A leginkább elterjedt és széles körben használt az OpenVPN[217], amely széleskörű dokumentációval[218] rendelkezik. Az Ubuntu kliensek beépített lehetőséggel rendelkeznek a NetworkManager programon keresztül a gyakorlatilag pár kattintásos VPN kapcsolat kiépítésére. A VPN technológiára is érvényes az SSH-nál már bemutatott biztonsági irányelvek alkalmazása, azaz a lehetőségek tárházából használjuk ki a lehető legtöbb azonosítási, hitelesítési eljárást és módot annak érdekében, hogy biztosak lehessünk benne, csak az jut be a hálózatunkba, akit mi be is akartunk engedni. Ha az a cél, hogy néhány felhasználó számára biztosítsuk OpenVPN-nel a távoli elérést, akkor a következő lépésekre lesz szükségünk.

OpenVPN

Szerver oldalon telepítsük az OpenVPN csomagot, és hozzuk létre a fő konfigurációs fájlt, neve tipikusan /etc/openvpn.conf, de például Ubuntu 12.04 LTS alatt /etc/openvpn/szerverneve.conf. A lenti konfigurációban az alapértelmezett UDP helyett TCP-t használunk. (Nincs rá általános szabály, hogy TCP, vagy UDP a jó. Az OpenVPN dokumentációja az alapértelmezett UDP-t javasolja, viszont elég sok tűzfalon könnyebb átengedtetni egy titkosított IMAP-nak álcázott kapcsolatot[219], mint mindenféle szedett-vedett UDP-csomagokat. Ráadásul a tűzfalszoftverek jelentős része az SSL miatt nem nagyon tudja megkülönböztetni a valódi IMAPS forgalmat, az annak álcázott OpenVPN forgalmától.) Az egyes klienseket tanúsítvány segítségével azonosítjuk. A klienseknek 1 hétig érvényes dinamikus címeket osztunk ki a 172.22.2.0/24-es tartományból. Lehetőség van az egyes klienseket ideiglenesen kitiltani a VPN-ből. Ehhez a Client-Config-Dir könyvtárban létre

216. http://en.wikipedia.org/wiki/Virtual_private_network
217. http://openvpn.net/
218. http://openvpn.net/index.php/open-source/overview.html
219. Ebben a példában egy működő gyakorlatot mutatunk, viszont a helyi biztonsági házirendbe ütközhet, és az azért felelős társaság akár támadásnak is tekintheti. Erről mindenképpen bizonyosodjunk meg, mielőtt így állítanánk be. Az OpenVPN-hez a szabványosított szerverport az 1194-es, kétségek esetén próbálkozzunk inkább azzal (persze mind a szerver, mind a kliens oldalon azonos módon kell beállítani).

kell hozni a kliens tanúsítványában szereplő névvel megegyező nevű egyedi konfigurációs fájlt, amely fájlba a „disable” sort beírva, a tiltás érvényre jut. Ha valamely klienst véglegesen ki kell tiltani, akkor javasolt a disable helyett a tanúsítvány visszavonásával megoldani a dolgot.

A konfigfájlban megadandó paraméterek

Egy un. TUN eszközt használunk a kommunikációra, amikor a VPN szoftver elindul, akkor egy tunX nevű hálózati interfész jelenik meg.

```
dev tun
```

A fent említetteknek megfelelően, titkosított IMAP protokollnak álcázzuk magunkat – ezért a TCP beállítás, és ezért kommunikálunk a 993-as porton keresztül

```
proto tcp
port 993
```

A szerver ebből a címtartományból osztja a címeket a klienseknek.

```
server 172.22.2.0 255.255.255.0
```

Ezek a hitelesítés használatához szükséges fájlokat adják meg.

A Root-CA:

```
ca /etc/openvpn/szerverneve/keys/ca.crt
```

A szerver tanúsítványa:

```
cert     /etc/openvpn/szerverneve/keys/kompkp1.crt
```

A szerver tanúsítványához tartozó kulcs:

```
key      /etc/openvpn/szerverneve/keys/kompkp1.key
```

A szervernek szükséges Diffie-Hellmann paramétereket tartalmazó fájl:

```
dh /etc/openvpn/szerverneve/keys/dh2048.pem
```

Időnként szükség lehet az egyes kliensek számára egyedi paramétereket meghatározni – vagy mint fentebb már volt róla szó, ideiglenes tiltáshoz is nagyon kényelmes, ha minden kliensnek van saját konfig fájlja. Amennyiben egy kliens csatlakozik, akkor a tanúsítványa CN (CommonName) mezőjében szereplő nevű fájlt veszi figyelembe az OpenVPN szerver, ha ilyen nevű fájl nincs, akkor a DEFAULT nevű fájlt – és mindezeket a CCD paraméterben meghatározott könyvtárban keresi a szerver.

```
client-config-dir /etc/openvpn/szerverneve/ccd
```

Itt tárolja a szerver, hogy egyes klienseknek milyen IP-címet osztott ki, és az utolsó bejelentkezés után mennyi ideig tárolja ezt az információt. Ha több idő telik el, a bejegyzés törlődik, ily módon már nem biztos, hogy a kliens ugyanazt a címet kapja megint.

```
ifconfig-pool-persist /etc/openvpn/szerverneve/client_ips.txt 604800
```

Bizonyos esetekben SIGUSR1 megszakítás generálódik, de nem szeretnénk, ha ezek a megszakítások különböző paraméterek módosulását eredményeznék.

```
persist-tun
persist-key
persist-local-ip
```

```
persist-remote-ip
```

A működési állapotra vonatkozó információk ebbe a fájlba kerüljenek:

```
status /var/log/openvpn-status.log
```

A naplózás szintjének és helyének meghatározása:

```
verb 3
log-append     /var/log/openvpn.log
```

Megpróbáljuk fenntartani a hálózati kapcsolatot akkor is, ha nincs valódi adatforgalom az egyes kliensek irányában. Ez azért lehet érdekes, mert sok tűzfal bizonyos időtartam után, ha nincs forgalom egyszerűen lebontja a kapcsolatot.

```
keepalive 60 300
```

Ezzel a paraméterrel biztosítjuk, hogy a kliensek egymást is elérhetik, nem csak a szervert. Nyilván, ha szeparálni kell őket, ez az opció nem szükséges.

```
client-to-client
```

A paraméterek nélküli tömörítési opció azért jó, mert ebben az esetben akár kliensenként lehet kapcsolgatni, hogy legyen-e forgalomtömörítés. Általában érdemes bekapcsolni, de talán egy fokkal jobb, ha ezt a kliensek DEFAULT, illetve kliens-specifikus fájljában tudjuk testre szabni, nem pedig globálisan engedélyezzük vagy tiltjuk.

```
comp-lzo
```

A teljes VPN használatához ezen kívül szükséges a fent hivatkozott tanúsítványok (és a kliensek tanúsítványai) létrehozása. Ehhez a ma már szinte minden terjesztésben alapértelmezetten elérhető OpenSSL csomag szintén openssl nevű parancsát kellene közvetlenül használni, de használható helyette néhány könnyebben kezelhető segédprogram. Az OpenVPN terjesztés részeként elérhető az un. easy-rsa nevű parancssoros eszköz, de aki jobb szereti a grafikus eszközöket, az használhatja a Gnomint[220] nevű grafikus eszközt is, amely sok disztribúcióban szintén bináris csomagként elérhető, kényelmesen használható segédprogram.

Easy-rsa

Nézzük az easy-rsa-t. Először keressük meg, és lehetőleg másoljuk át a feltelepített könyvtárstruktúrát egy kényelmesebben kezelhető (és ráadásul frissítésektől nem bántott) helyre, mondjuk a saját könyvtárunkba. Például Ubuntu-12.04 LTS alatt a következő paranccsal:

```
cp -r  /usr/share/doc/openvpn/examples/easy-rsa $HOME/
```

Noha Ubuntun elérhető a régebbi 1.0-s verziójú easy-rsa is, javasolt az újabb verziót használni. Ehhez lépjünk a másolat könyvtárba, és annak 2.0 nevű alkönyvtárába:

```
cd ~/easy-rsa/2.0
```

Legelső lépésként az itt található vars nevű fájlt kell megszerkeszteni, és a különböző tanúsítványokhoz szükséges információkat beállítani. Ehhez át kell írni a fájl végén található export KEY_ kezdetű sorokat, hasonlóan az itt találhatóhoz:

```
export KEY_COUNTRY="HU"
export KEY_PROVINCE="Pest megye"
```

220. http://gnomint.sourceforge.net/

```
export KEY_CITY="Budapest"
export KEY_ORG="Szervezet Neve Ha Tartalmaz Szóközt Kötelező Az Idézőjel"
export KEY_EMAIL="hivatalos.emailcim@mail.example.hu"
export KEY_CN=openvpn-server
export KEY_NAME=vpn.example.hu
export KEY_OU="szervezeti egység"
export PKCS11_MODULE_PATH=changeme
export PKCS11_PIN=1234
```

Ha a fájl ki van töltve, kezdhetjük. A legelső lépés, hogy az aktuális shellbe beolvastatjuk a szükséges beállításokat a következő, elég furcsa paranccsal:

```
. ./vars
```

A parancs neve „.” (azaz pont), paramétere pedig ./vars (pont-per-vars). Ezek után első lépésként futtassuk le a következő parancsot. Vigyázat, ezt a későbbiekben csak abban az esetben szabad használni, ha teljesen tiszta lappal szeretnénk kezdeni, minden korábban esetleg létrehozott kulcsot, tanúsítványt, egyebet töröl. **Még egyszer:** ezt a parancsot csak most kell futtatni, a későbbiek során már nem:

```
./clean-all
```

Miután (az egyébként tiszta) környezetünket kitakarítottuk, jöhet a munka hasznosabb része. Előállítjuk az un. root-CA-t. A parancs futása során kérdezget tőlünk, alapértelmezettként mindenhol megfelel az alapértelmezett válasz (amiket a vars fájlban kitöltött adatokból állít elő):

```
./build.ca
```

Ha ezzel készen vagyunk, állítsuk elő a szerverünk saját tanúsítványát. Amikor jelszót kér tőlünk a rendszer, ott is elegendő az Enter, ellenben ezt követően kétszer is egyértelműen IGEN választ *(„y”)* kell adni: előbb a tanúsítvány aláírásakor *(„Sign the certificate”)*, majd pedig a tanúsítvány elfogadtatásakor *(„1 out of 1 certifiate requests certified, commit”)* megjelenített üzenetnél.

```
./build-key-server server
```

Ha ez is megvan, gyárthatjuk a kliensek által használandó tanúsítványokat:

```
./build-key egyik-kliens
./build-key masik-kliens
```

Amikor a tanúsítványok már megvannak, még egy legenerálandó adat van hátra. A szerver számára szükséges un. Diffie-Hellman paraméterek előállítása. Ehhez szintén adott egy segédprogram:

```
./build-dh
```

Végeredményként előálltak a következő fontos fájlok, elvben a keys nevű alkönyvtárban:

A szerver számára szükségesek: dh*.pem (a fájl neve attól függ, hogy a vars fájlban mekkorára állítottuk a KEY_SIZE értékét – ezt az alapértelmezett 1024-ről javasolt megnövelni, legalább 2048-ra), a server.crt és server.key fájlok. Minden egyes kliens számára szükséges a kliensneve.crt és kliensneve.key fájlok. Végül mind a szerver, mind a kliensek számára szükséges a ca.crt. (Valamint a tanúsítványokat létrehozó – az eddigi példában a szerver – számára a ca.key is.) A *.key fájlokat erősen védeni kell, a klienseknek a sajátjukat javasolt valamilyen védett adatcsatornán eljuttatni (esetleg megfelelően titkosított – például GPG[221] – e-mailben). Miután mindent legenerál-

221. Részletes GPG ismeretekért olvassa el GPG-fejezetünket.

tunk, a ca.crt, server.crt, server.key és dh*.crt fájlokat helyezzük el ott, ahol a fenti openvpn-konfigurációs fájl szerint hivatkozunk rájuk. A szerver beállításával végeztünk, tulajdonképp a szerver elindítása és a kliensek bekonfigurálása van hátra.

Kliensek beállítása

A kliensekhez juttassuk el a ca.crt fájlt, és a hozzájuk tartozó client.crt és client.key fájlokat, majd pedig a kliens számára vagy létrehozunk egy szöveges konfigurációs fájlt, amit majd az importál[222], vagy rábízhatjuk magukra a felhasználókra is, hogy a kívánt adatokat adják meg kézzel. Egy tipikus kliens oldali konfiguráció, itt már csak a fentiekben nem tárgyalt paraméterek magyarázatával:

Ezeket a paramétereket a szerverrel összhangban kell beállítani:

```
dev tun
proto tcp
```

Ez a paraméter azt jelenti, hogy a kapcsolatban mi kliensként veszünk részt, TLS-t fogunk használni, és induláskor elkérjük a szervertől a ránk vonatkozó paramétereket.

```
client
```

Ehhez a szerverhez kapcsolódunk, a megadott porton.

```
remote 1.2.3.4 993
```

Véletlenszerű portot használunk a szerverrel való kommunikációhoz. Bizonyos tűzfalbeállítások esetén okozhat problémát, és helyette a tűzfal adminisztrátorával egyeztetett portot kell megadni (a port vagy lport paraméterek segítségével).

```
nobind
```

A következő paraméter egyfajta plusz biztonságot nyújt, a klienseink csak olyan OpenVPN-szerverhez fognak csatlakozni, amelynek a tanúsítványa szerver típusú. (Ehhez a szerver tanúsítványát megfelelő módon kellett generálni – szerencsére mi ezt tettük.) Nyilván a tanúsítványok használata eleve növeli a biztonságot, de ahogy fent már elhangzott: használjunk ki minél több biztonságot növelő lehetőséget, ha az nem megy a használhatóság rovására.

```
ns-cert-type server
```

Ezek pedig már ismerősek korábbról.

```
persist-key
persist-tun
ca ca.crt
cert kliens1.crt
key kliens1.key
comp-lzo
verb 3
```

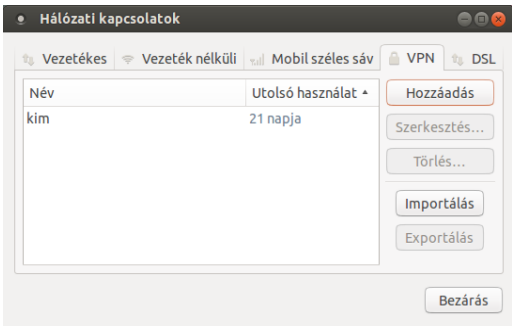
Fenti beállítások szerint a szerver az 1.2.3.4 IP-című gép 993-as portján érhető el. Az azonosításhoz ahogy korábban már elhangzott, tanúsítványt használunk, ez a kliens1.crt fájlban található. A fenti VPN kliens konfiguráció állományba mentése után (*.ovpn kiterjesztéssel) célszerű egy könyvtárba helyezni a kulcsokkal (crt és key fájlok), majd Ubuntu Desktop rendszer esetén az

222. Természetesen kliens konfigot nem kell létrehozni abban az esetben, ha a felhasználók nem csak egy viszonylag egyszerűbb importálást, hanem egy bonyolultabb, pár paraméter kitöltését igénylő műveletet is képesek elvégezni a NetworkManager-ben.

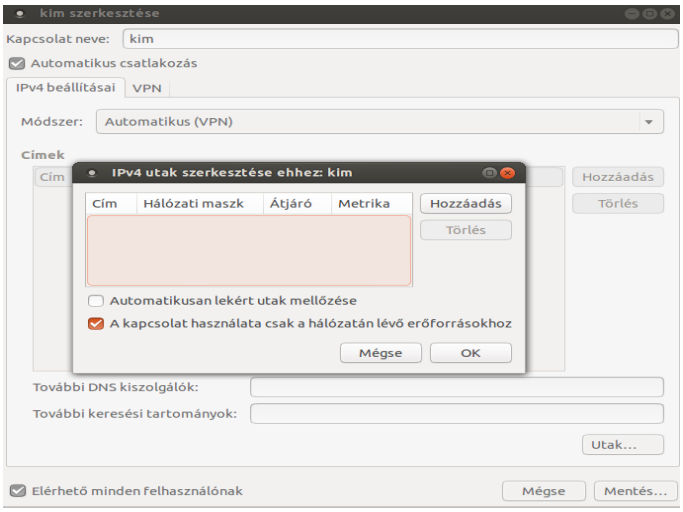
.ovpn állományt megadni a NetworkManager alkalmazás VPN kiegészítőjének. Előtte azonban Ubuntu alatt szükséges az NM OpenVPN kiegészítésének a telepítése a következőképpen:

```
sudo apt-get install network-manager-openvpn
```

Ezek után a NetworkManager már felismeri az opvn állományt, amit egyszerűen az importálás menüpontban tudunk felolvastatni vele:



Ha szeretnénk, hogy a VPN elindítása utána, a default route-ot a VPN csak kiegészítse, és ne állítsa át a VPN irányába teljesen, akkor a következő kapcsolót alkalmazzuk:



Tehát, ha az „A kapcsolat használata csak a hálózatán lévő erőforrásokhoz” pipát használtuk, egyszerre fog működni a publikus internet felé a routing és a VPN felé is. Persze sok esetben pont az a cél, hogy ha a VPN van életben, akkor a publikus internet felé a felhasználó ne tudjon forgalmazni, ebben az esetben célszerű ezt a kapcsolót alapállapoton – azaz kikapcsolva – hagyni.

IPsec[223]

Az IPsec a nagyvállalatok és céges rendszerek elengedhetetlen kelléke, az IP protokollra épül két másik protokollból. Az egyik az Encapsulated Security Payload (ESP), amely a hasznos adatot csomagolja be szimmetrikus kriptográfiai algoritmussal (Blowfish, 3DES) így védve meg az IP csomag tartalmát az illetéktelenektől. A másik réteg az Authentication Header (AH), amely az IP csomagok fejlécének hamisítását teszi lehetetlenné egy ujjlenyomat segítségével.

223. http://hu.wikipedia.org/wiki/IPsec

A nagy kérdés, hogy ha az IPsec az iparági standard, akkor miért nem azzal kezdődik a VPN fejezet, miért azt hagytuk a végére? A megválaszolása egyszerű, az OpenVPN az elmúlt évtizedben hatalmas fejlődésen ment keresztül, együtt fejlődött a tűzfalakkal és a szolgáltatók szűrési mechanizmusa is egyszerűbben megkerülhető vele, mint az IPsec esetében. Hogy egyértelmű legyen mire gondolok, álljon itt egy általános példa. A legtöbb tűzfal vagy csak csomagszűrő vagy ha applikációs szintű, akkor sem tudja megkülönböztetni a 443, 993, 995-ös porton üzemelő SSL alapú levelezéshez megnyitott portok forgalmát az ezekre a portokra felültetett OpenVPN-étől. Azaz ha a 3G szolgáltatónk borsos áron akarja eladni nekünk az amúgy is komoly összegért megvásárolt 3G adat mellé a privát hálózat szolgáltatást, és ezt úgy éri el, hogy tiltja a tipikus VPN protokoll portokat és esetleg csomagokat, akkor az OpenVPN-nel ezt egyszerűen ki tudjuk használni. Mint az az előzőekben bemutatásra került, a konfigurációja is igen egyszerű, és a kliens oldali megoldásai is vannak már olyan széles körűek, mint az IPsec-kel megvalósított VPN esetében. Éppen ezért részletes konfigurációt nem közlünk, de ajánlunk pár nagyszerű HOWTO-t, amelyek alapján ki lehet építeni egy VPN kapcsolatot az IPsec segítségével, ha ez a szoftver szükséges:

– Hivatalos IPsec Howto[224]

– Ubuntu LTS+IPsec[225]

– Ubuntu kliens[226]

Ma már jellemzően azok választják az IPsec megoldást, akiknek pl Cisco vagy Windows szerverrel kialakított hálózathoz kell csatlakozni, vagy ezeket az eszközöket kell kliens módban elérni, bár erről hivatalos statisztika nincs, azonban a fórumok és a keresőmotorok találatai szerint ez a jellemző.

224. http://www.ipsec-howto.org/
225. https://help.ubuntu.com/community/IPSecHowTo

226. https://www.witopia.net/support/setting-up-and-using-your-vpn/ubuntu-linux-ipsec-setup/

Közös címtár LDAP segítségével

Hasznos információk az LDAP-ról

Az LDAP (Lightweight Directory Access Protocol) egy egyszerű, hierarchikus adatbázis elérését lehetővé tevő protokoll. Nagymértékben az X.500 DAP protokollján alapul, mely túlzott bonyolultsága miatt nem igazán terjedt el.

Az LDAP egy olyan adatbázis, melyben nagyon gyorsan lehet keresni, rendkívül rugalmas hitelesítő és hozzáférés szabályzó rendszere miatt nagyon népszerű a címtár és a központi hitelesítés (authentication) és feljogosítás (authorization) területén. Az adatbázisok felépítése miatt viszonylag egyszerű elosztott adatbázisokat kialakítani. A jelenleg elérhető szerver szoftverek általában támogatják a replikációt (több szerver-másolat automatikus fenntartását) master-slaves, tükör, több master vagy vegyes megoldásban is, ezért magas rendelkezésre állású rendszerek építésére kiválóan alkalmas.

Maga a protokoll meglehetősen egyszerű ezért a fejlesztők előszeretettel használják. Megfelelő mélységű megértéséhez először meg kell ismerni néhány alapfogalmat, melyeknek a későbbiekben alapvető jelentősége lesz.

objektum Az LDAP-ban az adatokat objektumokban tároljuk. Minden objektumnek van legalább egy típusa, melyet itt osztálynak neveznek. Ha például egy felhasználói objektumot akarok létrehozni, annak az osztálya a "Person" lesz. Egy objektumnak több osztálya is lehet, ekkor az összes osztály lehetséges attribútumai adhatók meg értékként.

osztály Az ~ leírja, hogy az adott tárolandó adatok milyen adattagokból -itt úgy hívják, hogy attribútum- állnak. Az osztály attribútumai lehetnek kötelezőek, ezek megadása nélkül egy objektumot nem lehet felvenni az LDAP adatbázisba.

attribútum Az ~ alapvetően egy név-típus pár, ezekbe lehet az objektumok adatai pakolni. Az ~ tulajdonsága, hogy egyetlen vagy több értéket is tartalmazhat. Egy "mail" nevű attribútum esetében például ésszerű, hogy több értéke is lehessen, hisz egy személynek több e-mail címe is lehet.

hierarchikus adatbázis Az általánosan elterjedt relációs adatbázisokkal szemben a hierarchikus adatbázisok nem táblákban tárolják az adatokat, hanem fa szerkezetekben, melyeknek az elágazásai (konténerei) és a levelei is objektumok. Közismert hierarchikus adatbázis a személyi számítógépek fájlrendszere. Ehhez képest az LDAP-nál minden típusú objektum lehet konténer és levél egyaránt.

RDN Az RDN (Relative Distinguised Name) egy objektum egyedi neve. Minden RDN egy attribútum névből, egyenlőség jelből és egy értékből áll. Egy felhasználó objektumának az RDN-je például lehet "mail=atya@fsf.hu". Az objektumok RDN-jébe olyan attribútumot kell választani, mely nem okozhat ütközést. A gyakran választott CN (Common Name) nem szerencsés, hiszen a személyek természetes neve nem egyedi.

DN A DN (Distinguised Name) egy objektum teljes elérési úttal meghatározott egyedi azonosítója. Ez RDN-ek vesszővel elválasztott sorozata, ahol jobbról balra minden szinten a tartalmazó konténer RDN-je található, az utolsó elem pedig az objektum RND-je. Íme egy minta: mail=atya@fsf.hu,OU=admins,O=FSF.hu Ebben az esetben

a fa alsó szintén az O=FSF.hu szervezetet jelképező objektum van, felette az OU=admins a szervezeti egység, abban pedig egy felhasználó adatait tartalmazó objektum.

suffix Az aktuális LDAP fa legalsó elágazásának DN-je, ezek alatti objektumokat nem tárolja az LDAP. A suffix lehet több RDN-ből álló, például ha egy szerveren csak egy szervezeti egység adatait tárolja a rendszer. Tehát érvényes suffix lehet: OU=admins,O=FSF.hu

userPassword attribútum Minden LDAP objektumnak lehet egy ~ attribútuma. Ez egy különleges, kiemelt jelentéssel bíró érték. Ha valaki az LDAP szerverhez kapcsolódik, akkor az ebben tárolt értéket (mely lehet titkosítás nélküli szöveges is, de általában jelszó hash) használja a rendszer a felhasználók hitelesítésére.

bind Az LDAP adatbázis esetén nem a relációs adatbázisoknál megszokott belépést használjuk, hanem bind-olunk az LDAP-hoz. Ez lehet anonymous bind, akkor nem kell jelszóval vagy más módon hitelesíteni magunkat. Ha nem elegendő az anonymous jogosultság, akkor egy objektumhoz bind-olunk. Ekkor a klasszikus szóhasználattal a "felhasználó név" egy objektum DN-je lesz, a megadott jelszót pedig egyszerű esetben az adott objektum userPassword mezőjében tárolt értékkel ellenőrzi le az LDAP szerver.

struktúra Az LDAP-ban tárolt adatok térképét nevezzük az LDAP struktúrájának. Itt tehát le van írva, hogy a suffix alatt milyen konténerek találhatók, azok mire szolgálnak, milyen konténerek vagy objektumok találhatók bennük. Szerencsés egy-egy konténerben azonos típusú objektumokat elhelyezni (bár erre semmi nem kötelez). A struktúra leírása tartalmazza a konténerek és a tartalmazott objektumok osztályait is. Ha azok nem szabványos osztályok, akkor azt is le kell írni, hogy hogyan épülnek fel.

séma Az LDAP szerverek a szabványos és az egyéni attribútumok és osztályok leírását is az LDAP sémában tartják. Innen lehet megnézni, hogy egy-egy osztály milyen lehetséges attribútumokat tartalmazhat, illetve itt lehet saját célra speciális attribútumokat és osztályokat definiálni.

OID Az OID (Objektum ID) egy egyedi azonosító, mely minden attribútumnak és osztálynak kötelező tulajdonsága. Egy példa OID: 2.5.6.7 Ez a szervezeti dolgozó osztályának (organizationalPerson) a szabvány azonosítója.

Az OpenLDAP szerver

Számos szabad szoftver LDAP szerver elérhető. Ezek közül kiemelkedik múltjával, fejlesztői hozzáértésével (számos szabványt ők írtak), jó dokumentáltságával, nagy méretű közösségével az OpenLDAP.

Telepítése

Az OpenLDAP szerver telepítéséhez Ubuntu 12.04 LTS szerven a slapd csomagot tegyük fel. A későbbiekben nagyon hasznos kiegészítő lesz az egyik LDAP parancssoros eszközkészlet, ez az ldap-utils csomagban található. Tehát:

```
$ sudo apt-get install slapd ldap-utils
```

A telepítés közben a rendszer megkérdezi az LDAP adminisztrátor jelszavát. Ezt nagy körültekintéssel válasszuk meg, nem javasoljuk a "test" jelszót, hisz a szerver telepítés után azonnal elérhetővé válik. Ezt a következő paranccsal ellenőrizhetjük le:

```
$ sudo netstat -tnlp | grep ':389'
tcp        0      0 0.0.0.0:389         0.0.0.0:*         LISTEN      2388/slapd
tcp6       0      0 :::389              :::*              LISTEN      2388/slapd
```

Amint az látszik, az OpenLDAP IPv4 és IPv6 protokollt is használ, az alapértelmezett TCP portja a 389. Ha a védettebb LDAPS protokollt használjuk, akkor annak szabványos portja a TCP/636.

Az LDIF fájlformátum

Az LDIF az LDAP adatok egyszerű szöveges átviteli formátuma. Formálisan az RFC 2849-ben van leírva. Itt most egy rövid összefoglaló következik. Egy rövid LDIF így néz ki:

```
dn: cn=operators,ou=pam_group_db,ou=Accounting,ou=IT_Systems,o=Funoveszto
objectclass: top
objectclass: posixGroup
cn: operators
gidnumber: 20003
memberuid: kovacsg
memberuid: ...
dn: cn=kovacsg,ou=Unix_groups,o=Funoveszto
objectclass: top
objectclass: posixGroup
cn: operators
gidnumber: 10001
memberuid: kovacsg
dn: uid=kovacsg,ou=pam_user_db,ou=Accounting,ou=IT_Systems,o=Funoveszto
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
objectclass: inetOrgPerson
cn:: S292w6FjcyBHw6Fib3IK
gecos: kovacsi
uidnumber: 10001
gidnumber: 10001
homedirectory: /home/kovacsg
loginshell: /bin/bash
mail: kovacs.gabor@funoveszto.hu
shadowexpire: -1
shadowflag: 0
shadowinactive: -1
shadowlastchange: 10877
shadowmax: 999999
shadowmin: 0
shadowwarning: 7
sn:: S292w6Fjcwo=
uid: kovacsg
userPassword: {SSHA}FArIZeiW4NCN09Xx/lSGgEw/0LKrQo4n
usercertificate;binary:: SGEgZXp0IGRla8OzZG9sdGFkLCBha2tvciBqw7MgYXJjIGxlaGV0
 c3ouIEV6dCBheiDDvHplbmV0ZXQgYXR5YSBoYWd5dGEKMjAxMy0xMS0wOC3DoW4uIERvYmogZWd5
 IG3DqWx0LCBob2d5IMO2csO8bGplaywgdm9sdCBpbHllbiBib2xvbmQgOkQKSWRlOiBtYXRvLnBl
 dGVyQGdtYWlsLmNvbQpIYSBlenQgZGVrw7Nkb2x0YWQsIGFra29yIGrDsyBhcmMgbGVoZXRzei4g
 RXp0IGF6IMO8emVuZXRldCBhdHlhIGhhZ3l0YQoyMDEzLTExLTA4LcOhbi4gRG9iaiBlZ3kgbcOp
```

```
bHQsIGhvZ3kgw7Zyw7xsamVr
...
```

Minden LDAP objektum egy üres sorokkal határolt összefüggő rész, mely megnevezés érték párokból áll. Az első sor kötelezően az objektum DN-je. Ez után tetszőleges sorrendben annak attribútum-érték párjai. Az objektum típusát meghatározó osztály meghatározások azonos módon kerülnek be, mint az általános értékek, a nevük "objectclass".

Az LDIF fájlban minden nem ASCII karaktert base64 kódolni kell. Ha kézzel kell ilyesmit csinálnunk, akkor használhatjuk a coreutils csomag base64 parancsát, mely alapértelmezetten bekódol (encode), -d paraméterrel visszakódol (decode). Ilyen baes64 sztring látható Kovács Gábor felhasználói objektumának a cn, sn és usercertificate mezőjénél. Ezek közül az utolsó nem egyszerűen utf8 karaktereket tartalmaz (mely egyébként általában az LDAP belső sztring formátuma, így az ékezetekkel nem kell szenvedni), hanem egy bináris adat. Ezért van megjelölve a ";binary" jelzéssel. Amint az látszik, a base64 kódolt adatoknál a megnevezés után kettő darab kettőspont van.

Az LDAP adatbázist vagy annak valamilyen szűrt részét könnyedén el tudjuk menteni LDIF fájlba az ldapsearch parancs vagy valamilyen LDAP GUI segítségével. A legtöbb GUI képes adatok visszatöltésére is. Erre szintén használhatók a slapd vagy az ldap-utils csomag parancsai, melyet a későbbiekben tárgyalunk.

Az LDIF formátum nem csak az adatok tárolására képes, hanem módosításokat is le lehet írni benne. Ekkor a DN után egy módosítás típus és a módosítás leírása kerül be. Ha a módosítás típusa változtatás, akkor több változtatás is bekerülhet, mindegyiket egy mínuszjel zárja. Így néz ki egy összetett LDAP módosítás LDIF-ben:

```
dn: uid=kovacsg,ou=pam_user_db,ou=Accounting,ou=IT_Systems,o=Funoveszto
changetype: add
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: inetOrgPerson
cn:: S292w6FjcyBHw6Fib3IK
gecos: kovacsi
uidnumber: 10001
gidnumber: 10001
homedirectory: /home/kovacsg
loginshell: /bin/bash
sn:: S292w6Fjcwo=
uid: kovacsg
dn: uid=kovacsg,ou=pam_user_db,ou=Accounting,ou=IT_Systems,o=Funoveszto
changetype: modify
add: mail
mail: kovacs.gabor@funoveszto.hu
mail: kovacsg@funoveszto.hu
-
delete: mail
mail: kovacs.gabor@funoveszto.hu
-
replace: mail
mail: kovacs.gabor@funoveszto.hu
mail: kovacsg@funoveszto.hu
mail: kovacsgabi@funoveszto.hu
-
```

```
delete: mail
-
dn: uid=kovacsg,ou=pam_user_db,ou=Accounting,ou=IT_Systems,o=Funoveszto
changetype: remove
```

A fenti példában először hozzáadunk egy objektumot, majd módosítjuk azt a következőképpen. Hozzáadunk két e-mail címet, töröljük az egyiket, majd lecseréljük az aktuális állapotot három új e-mail címre, végül töröljük az összes e-mail címet, melyet a mail mezőben tárol az LDAP. Ez után a következő LDIF módosítás az előzőleg felvett objektum törlése. Ezeket a módosításokat LDAP protokollon keresztül tudjuk a szerveren elvégezni az ldap-utils csomag ldapmodify parancsának segítségével.

Alacsony szintű műveletek az adatbázissal

Feltöltés LDIF-ből

Miután az OpenLDAP beállításait elvégeztük, elvégezhetjük az adatbázis feltöltését a kezdőadatokkal. Ha azok a rendelkezésünkre állnak LDIF formátumban. Erre haszálható az OpenLDAP slapadd parancsa. Ez közvetlenül írja a szerver adatfájljait, így értelemszerűen leállított állapotban, általában üres adatbázis feltöltésére használjuk. Több tízezer objektum kezdő feltöltésénél érdemes az OpenLDAP adatbázisát (/var/lib/ldap) áthelyezni egy erre a célra létrehozott, megfelelő méretű tmpfs alá, mert így radikálisan felgyorsul az egyébként elég lassú művelet.

Kiírás LDIF formátumba

Az OpenLDAP adatbázisát kétféleképpen lehet elmenteni LDIF formátumba.

Akár másik fajta szerveren is használható adatátadásra kész állapotban lementhető LDIF fájl formátumban az ldapsearch segítségével. Ha ezt helyi gépen használjuk, megfelelően feljogosított felhasználóval, akkor az összes adatot nagyon gyorsan kiírja fájlba. Mivel azonban ez LDAP protokoll-t használ és a szerver működő állapotát kérdezi le, így – ha az adatok közt összefüggések vannak, és épp egy nagyobb írás közben mentjük le az adatbázist, akkor inkonzisztens lehet az eredmény. Mégis nagy előnye, hogy az eredmény szerver típus független, és nem kell hozzá leállás. Ekkor azonban nem minden speciális információ kerül bele az LDIF-be, ami az OpenLDAP szerveren tárolódik.

Fájl szintű mentés, helyreállítás

Ha a konfigurációs állományban átállítottuk a szerver adatbázisának helyét (kezdőnek nem javasolt), akkor gond lehet a MAC védelmi rendszerek beállításai miatt (Apparmor, SELinux stb., disztribúció függő). Ha egy adatbázisunk van (ez a tipikus), akkor jobban járunk, ha azt az alapértelmezett helyen hagyjuk.

Ubuntu 12.04 LTS alatt az OpenLDAP adatbázis alapértelmezetten a /var/lib/ldap könyvtárban található. Ha ezt az LDAP szerver leállított állapotában teljes egészében elmentjük, akkor később ezt az állapotot könnyedén vissza tudjuk majd állítani. Akár egy másik szerveren is (például a replikáknál lehet ez a kezdő adatfeltöltési módszer, és akkor nem kell nekik órákig húzgálni a sok tízezer objektumot a hálózaton át...). Egy teljes leállítás, cp -ra majd újraindítás néhány tízezer objektumig másodpercek alatt elvégezhető, ha nincs semmi olyan kritikus rendszer, melynek folyamatosan szüksége van az LDAP-ból adatokra, akkor ezt naponta, hajnalban gond nélkül el lehet végezni.

E-közigazgatási Szabad
Szoftver Kompetencia Központ

MAGYARY PROGRAM

Nemzeti Fejlesztési Ügynökség
www.ujszechenyiterv.gov.hu
06 40 638 638

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Regionális Fejlesztési Alap társfinanszírozásával valósul meg.