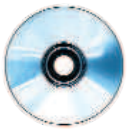
Postfix és Cyrus



Levélkézbesítő ügynök, IMAP- és POP3-kiszolgáló, valamint virtuális felhasználók. Ez volt a feladat Debian Woody alatt. Most egy sör mellett próbálom meg felidézni a nehezét.

Beteszem a kedvenc FatBoy-albumomat, hátradőlök a gurulós székben, és diadalittasan mosolygok a monitorra. A munka első megközelítésben nem volt bonyolult, viszont annyi apró részfeladatból állt, hogy úgy érzem, megérdemelten koccinthatom oda a sörösdobozt a monitor oldalához. Ha te is valami hasonló tett előtt állsz, de már szeretnél ott tartani, ahol én, ez a cikk neked szól.

SMTP és IMAP, POP3

Előttem már többen kellő mélységben tárgyalták az említett protokollokat, ezért nem szeretném őket megismételni. Ha időd engedi, feltétlenül olvasd el ***Deim Ágoston*** kapcsolódó cikkeit a ***Linuxvilág*** 2001. június–júliusi számától kezdve 2001 novemberéig. Részletesen beszél a Postfixről mint MTA-ról, és egy összehasonlító írás erejéig kitér a főbb IMAP-, illetve POP3-kiszolgálókra. Ez a cikk viszont a türelmetlen rendszergazdáknak íródott, ezért lássuk, mit is takarnak a fenti rövidítések.

Az elektronikus levelezés alapvetően két összetevőből áll. Az MTA (Mail Transfer Agent) feladata a levél kézbesítése a hálózat (legyen az helyi vagy maga az Internet) különböző részhálózatain keresztül. A MUA (Mail User Agent) pedig az a levelező-program, amit a felhasználó az elektronikus levél fogadására, elkészítésére, illetve postázására használ (pl.: Sylpheed, mutt). Az MTA-k közötti üzenetváltás protokollja az SMTP.

Ha minden egyes hálózatba kötött számítógépen lenne egy MTA, akkor itt véget is érne a történet. Ez azonban kivitelezhetetlen és felesleges. Kivitelezhetetlen, ha figyelembe vesszük, hogy az Internet használóinak nagy része még mindig nem rendelkezik állandó kapcsolattal és változó IP-címe van. Ugyanakkor felesleges azért, mert egy számítógépet többnyire nem használnak 2–3 embernél többen. Így egy MTA-nak csak 2–3 postaládát kellene kezelnie, ami erőforrás-pazarlás.

A megoldás meglehetősen egyszerű: egy központi gépet kinevezünk levélkiszolgálónak. Állandó kapcsolattal rendelkezik és DNS-rekordja is van, ami név alapján megmondja az IP-címét. Tárolja a beérkező leveleket és kézbesíti a kimenőket. A belső hálózatra kötött felhasználók ugyanúgy SMTP-t használva küldenek levelet, csak beállítják az MUA-ban, hogy melyik az SMTP-kiszolgáló. Beérkező leveleiket pedig egy külön e célra kifejlesztett protokollon át töltik le a saját gépükre. Ez utóbbi lehet IMAP vagy POP3.

Vajon IMAP-ot használjunk vagy POP3-at? Alapvető különbség a kettő között, hogy az IMAP-ot azzal az elképzeléssel fejlesztették ki, hogy a levelek mindig is a központi számítógépen maradnak, és a felhasználók távolról kezelik a leveleiket. Ezzel szemben a POP3 alapvetően arra szolgál, hogy a felhasználó az összes levelét a saját gépére tölti le, letöltés után pedig törli őket a kiszolgálóról. A válasz tehát ebben a megközelítésben a levelezőkiszolgálón lévő tárhely függvénye.

Válasszunk egy MTA-t!

Legyen mondjuk a Postfix. Miért? Mert megbízható, kellően moduláris, illetve gyorsan és egyszerűen beállítható. Nekem egy 166 MHz-es Pentiumra kellett felkönyörögnöm egy ezer felhasználót kiszolgáló rendszert. Határozottan úgy érzem, hogy a Postfix jó választásnak bizonyult, mert nem érkezett panasz a sebességet illetően.

Először telepítsd a `postfix`, `postfix-ldap`, `postfix-pcre` és a `postfix-doc` csomagokat. Akármi is volt eddig az MTA-d, mostantól a Postfix lesz az. A `postfix-ldap`-ot és `postfix-pcre`-t a Postfix igényli, ezért kötelező őket telepíteni, habár most nem fogjuk egyik előnyét sem kiaknázni. Az egyik segítségével LDAP-ból lehet venni bizonyos adattáblákat (például az alias táblát), míg a másikkal szabályos kifejezéseket (regular) lehet használni ugyanezekben a táblákban. Megjegyezném, hogy a `postfix-doc` a leírást nem a */usr/share/doc/postfix-doc*, hanem a */usr/share/doc/postfix* könyvtár alá telepíti.

A telepítés során a `General type of configuration?` kérdésre az `Internet Site` – első nekifutásra – jó választás, a finomhangolás így is, úgy is kézzel fog történni. Ezt meglepően egyszerű kérdések követik. Előbb a kiszolgáló nevét kérdezi meg, azután azt, hogyha valaki egy egyetlen tagból álló tartománynévre akar levelet küldeni, akkor hozzáragassza-e a saját tartománynevét. Így például a `pisti@mail` címet egészítse-e ki `pisti@mail.cegnev.hu`-ra. Ez akkor lehet érdekes, ha a belső hálózaton lévő felhasználóinknak nincs erejük beírni a teljes címet, amikor a szomszédnak küldenek elektronikus levelet. Ezt követően olyan tartománynéveket kell megadnod, amelyeket a Postfix a sajátjának tekint, vagyis ha valaki a @ után ezt írja be, és a levél eljut ehhez a kiszolgálóhoz, akkor már nem küldi tovább, hanem megtartja. Az alapértelmezés többnyire megfelelő. Az utolsó kérdés azokra az IP-címtartományokra vonatkozik, amelyek SMTP-kiszolgálónak használhatják a Postfixet. Ez jellemzően a 127.0.0.1/8 és a belső hálózat. Az IP-címet egy / (perjel) után követő szám határozza meg, hogy a címből hány bit az alhálózati maszk. Ez például 10.x.x.x formájú címeknél nyolc, de 192.168.x.x esetén tizenhat.

A `debconf` sokat segített, de azért maradt még feladat bőven neked is. A */etc/postfix/main.cf* állományt kell módosítanod, ez a Postfix fő beállítófájlja. Mivel a Cyrust később arra is használni szeretnénk, hogy a helyi felhasználók leveleit a megfelelő postafiókban tárolja, a következő változtatások szükségesek:

```
#mailbox_command = procmail -a "$EXTENSION"
mailbox_transport = cyrus
```

Vagyis megjegyzésbe kell tenni a `mailbox_command`-ot, és helyette fel kell venni a `mailbox_transport`-ot. Ezzel a Postfix kész is, de a java még hátravan, hiszen most jön a Cyrus.

Válasszunk IMAP- és POP3-kiszolgálót!

Legyen mondjuk Cyrus. Miért? Mert megbízható, többféle hitelesítési módszert támogat, és az ACL-ek (Access Control List) segítségével finoman szabályozhatók a jogosultságok. Az említett Pentium 166-oson is a Cyrusra esett a választásom. Telepítened kell a `cyrus-common`, `cyrus-admin`, `tcl8.0` és a `tclreadline` csomagokat, a `cyrus-imapd`-t, illetve a `cyrus-pop3d`-t – attól függően, hogy melyik protokollt szeretnéd elérhetővé tenni. A Tcl-csomagok a `cyrus-admin` végett szükségesek, ugyanis a `cyradm` nevű felügyeleti felület Tcl-t használ. Ezek mind kérdezés nélkül települnek, úgyhogy kézzel kell egy kicsit varázsolni.

A `cyrus-imapd`, illetve a `cyrus-pop3d` a Cyrus IMAP-et és POP3-at megvalósító moduljai. Mind a kettő `inetd`-ből indul. A telepítés során a */etc/inetd.conf* a következő sorokkal egészül ki:

```
pop3    stream tcp    nowait cyrus
➥/usr/sbin/tcpd      /usr/sbin/pop3d
imap2   stream tcp    nowait cyrus
➥/usr/sbin/tcpd      /usr/sbin/imapd
```

A két modulnak közös beállítóállománya a */etc/imapd.conf*. Itt a következő módosítások szükségesek:

```
admins: bela
popminpoll: 0
```

Az `admins` mező alapértelmezetten megjegyzésben van, feltétlenül ki kell tölteni, különben nincs olyan felhasználó, aki felügyelhetné a Cyrust a `cyradm`-on keresztül. A 0-s azonosítójú felhasználót (root) biztonsági szempontból itt nem ajánlott megadni. A Cyrus leírásában azt is megemlítik, hogy a rendszerfelügyelőnek ne hozzunk létre postafiókot. A legegyszerűbb egy rendszerben is létező, semmilyen egyéb joggal nem rendelkező felhasználót `admin`-nak felvenni. A `popminpoll` azt az időt határozza meg percben, amelynek egy felhasználó esetén két bejelentkezés között el kell telnie, ha a POP3-protokollt használja. Az alapértelmezés egy perc, ami éles rendszer esetén jó a nyers erő- (brute-force) támadások visszaszorítása érdekében, de amíg építed a kiszolgálót, és azt próbálgatod, hogy mely részei működnek, melyek nem, addig nem biztos, hogy ez lesz álmaid netovábbja. Én erre az időszakra ezt a korlátozást kikapcsoltam.

A terjesztésben is megtalálható, a ***SASL*** nevű függvénykönyvtárat nélkülöző Cyrus-változat kétféle hitelesítést támogat: a Kerberos-, illetve a Unix-típusút. A Unix-típusú azonosítás a `pwcheck` démonnal működik. Vagyis nem elég az `inetd`-t futtatni ahhoz, hogy a Cyrus üzemeljen. A */etc/init.d/pwcheck* néven található a vezérlő parancsfájl. A `pwcheck` démon két változatban található meg a Woodyban. A „standard” változat egy megadott ***passwd***, illetve ***shadow*** állománnyal dolgozik.

A PAM-os változattal viszont tetszőleges PAM-modult alkalmazhatunk a hitelesítéshez. Ha belenézel a */etc/init.d/pwcheck*-be, azt láthatod, hogy a futtatandó program neve */usr/sbin/pwcheck*. Ez azonban egy közvetett hivatkozás a */etc/alternatives/pwcheck*-re, ami pedig a */usr/sbin/pwcheck_standard*-re mutat. Ezért, ha a PAM-os változattal szeretnél dolgozni (én ezt ajánlom), akkor a következőt tedd:

```
# rm /etc/alternatives/pwcheck
# ln -s /usr/sbin/pwcheck_pam
➥/etc/alternatives/pwcheck
# /etc/init.d/pwcheck restart
```

A Cyrus és a PAM

Már csak egy feladatod maradt: meg kell oldanod a felhasználókezelést. Mivel a `pwcheck` most már PAM-ot használ, csupán keresni kell egy olyan PAM-modult, ami az `auth` részben használható és a */etc/passwd*-től eltérő adatbázist használ. Én először a `pam_userdb` modult próbáltam ki, ami a `libpam-modules` csomag része, vagyis alapból feltelepül. Ez egy tetszőleges Berkeley hash DB állományt képes feldolgozni, ahol a felhasználónevek a kulcsok és a jelszavak az értékek. Ezzel csak az a baj, hogy egyszerű szöveges formában tárolja a jelszavakat, vagyis ha valakinek sikerül megszereznie az állományt, az összes jelszó a birtokába jut. Tovább keresgélve ráakadtam a `pam_pwdfile.so`-ra, amellyel egy */etc/passwd*-hez hasonló felépítésű állományt használhatunk. Az utóbbi modul a `libpam-pwdfile` csomag része. Tulajdonképpen egyetlen fájlt tartalmaz, a */lib/security/libpam_pwdfile.so*-t, de ez elég is. A DES és az MD5 titkosítást is támogatja. Használatához mindössze a */etc/pam.d/cyrus* állományt kell szerkesztened, a következőképpen:

```
#auth  required      pam_unix.so nullok
auth   required      pam_pwdfile.so pwdfile
➥/etc/imapd.passwd flock
#account      required      pam_unix.so
account       required      pam_permit.so
```

Sok PAM-hívő most legszívesebben a pokolba küldene a `pam_permit.so` használatáért, de az `account` részben véleményem szerint nincs szükség semmiféle ellenőrzésre. Visszatérve a `pam_pwdfile-ra`: ennek összesen két kapcsolója van. A `pwdfile`-t szóköz után követi a használandó állomány. A `flock` nem kötelező kapcsoló. Engedélyezi a `flock()` függvény használatát, vagyis bekapcsol egy olyan szerkezetet, amivel megbizonyosodhatsz róla, hogy a megadott adatbázishoz egyidejűleg nem fordul két különböző folyamat (lásd `man 2 flock`). A kapcsolóval ellenkező hatást fejt ki a `noflock`. Ahogy a súgóoldalon is olvashatod, ez a biztosíték csak addig él, amíg az összes folyamat használja a `flock`-ot. Ezért a */etc/imapd.passwd*-vel nagyon óvatosan kell bánni.

```
#!/usr/bin/perl -w

use strict;
use Fcntl ':flock'; # A LOCK_* Ælland khoz

die "HasznÆlat: ".$0." {felhasznÆl }
➥{jelsz }\n" if $#ARGV != 1;

open (PWD,">>/etc/imapd.passwd");
flock(PWD,LOCK_EX); # csak Øn hasznÆlhatom
                    ➥az ÆllomÆnyt
seek (PWD,0,2); # ha valaki k zben elvitte
                ➥volna a mutat t
my $revord = $ARGV[0].":";
my $salt = join '',
➥('.','/',0..9,'A'..'Z','a'..'z')[rand 64,
➥rand 64]; # s  lØtrehozÆsa
$record .= crypt ($ARGV[1],$salt)."[13]\n";
print PWD $record;
flock (PWD,LOCK_UN); # lock kikapcsolÆsa
close (PWD);
```

```
#!/usr/bin/perl -w

use strict;
use IMAP::Admin;

die "HasznÆlat: ".$0." {felhasznÆl }\n"
➥if $#ARGV;

my $mb = "user.".$ARGV[0];
my $imap = IMAP::Admin->new('Server' =>
➥'localhost',  'Login' => 'bela',
                'Password' => 'almasretes');

die $imap->{'Error'} if $imap->create($mb);
die $imap->{'Error'}
➥if $imap->set_quota($mb,2000);

$imap->close;
```

Ahogy a */etc/passwd*-t csak `vipw`-vel szabad szerkeszteni, úgy ezzel is kellő körültekintéssel kell bánni. Az egyetlen bökkenő, hogy az állomány formátuma korántsem megszokott, és a *passwd*-re is csak messziről hasonlít:

```
felhasznÆl nØv : titkos tott_jelsz  [ szÆm ]
```

A szögletes zárójelek közé zárt szám jelzi, hogy milyen algoritmust használtak a jelszó titkosításakor. Akkor 13, ha a Unix `crypt()` függvényével történt, és 34, ha MD5-tel. Hogyan hozzunk létre bejegyzéseket az állományban, ügyelve a `flock()` használatára, és minél kevesebb idő alatt? A válasz a Perl (*1. lista*). Azért van szükség a `flock()` utáni `seek()`-re, mert a `flock()` függvény egészen addig nem tér vissza, amíg meg nem szerzi az „exkluzív zárat”. Ha azért nem kapja meg azonnal, mert valaki más már rátette a saját zárát az állományra, akkor megvárja, amíg leveszi róla. Ha a másik folyamat közben a fájlmutatót elvitte az állomány végéről (hozzáírásra nyitottuk meg), akkor vissza kell tenni. A `crypt()` függvényt használom titkosításra. Ez két értéket vár: a titkosítatlan szöveget és egy kis sót (salt). A só egy két karakterből álló véletlen szöveg.
A felhasználó-adatbázis most már feltölthető felhasználókkal. Mindenekelőtt vedd fel a felügyelődet, mert a `cyrus` most már nem a */etc/passwd*-t használja, így jelenleg egyetlen felhasználót sem ismer meg. Ha megvan az `admin`, létrehozhatsz mezei felhasználókat is, de ezeknek még nem lesz postafiókjuk. Ugyanis nemcsak a felhasználókat kell létrehozni, de a postafiókot is. A postafiók létrehozása a `cyradm` nevű programmal tehető meg:

```
# cyradm localhost
localhost userid: bela
localhost password: almasretes
localhost>
```

Természetesen az `almasretes` nem jelenik meg a képernyőn. Itt a legegyszerűbb meghívni a súgót, ez megadja az összes használható parancsot. A `pisti` felhasználónévhez tartozó postafiókot a `cm user.pisti` paranccsal lehet létrehozni (a `cm` a `create mailbox` rövidítése).
Most már nagyon közel vagy a sörhöz. Csakhogy a főnököd most adott a kezedbe egy hajlékonylemezt. A lemezen található állomány a felhasználóneveket és a jelszavakat tartalmazza, úgy ezret. A `cyradm` ehhez túlságosan is felhasználói beavatkozást igénylő (interactive). A válasz a Perl. Most egy olyan parancsállományt hozunk létre, ami csak egy felhasználónevet vesz át értékként, és létrehozza a megfelelő postafiókot, sőt még egy két megabájtos tárkorlátot is beállít. Jól hangzik, már csak a megfelelő Perl modul hiányzik. Telepítsd a `libimap-admin-perl` csomagot (*2 lista*).
A program magáért beszél, a `$imap objektum {'Error'}*` tulajdonsága a legutolsó meghívott elemfüggvény sikerességétől függően tartalmaz egy szöveges hibaüzenetet.

Jöhet a sör!

Az utóbbi két Perl-parancsfájlt használva lehetőségeid már korlátlanok. Bátran írj, ha valami nem úgy működik, ahogy leírtam, vagy valamit nem sikerült jól elmagyaráznom.
Jó sörözést!

A Cyrus és a Postfix programok forráskódjai a 47. CD Magazin/Cyrus könyvtárában találhatóak.

Fülöp Balázs (xut@freemail.hu)
18 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Az ELTE Radnóti Miklós Gyakorlóiskola tanulója immár ötödik éve. Kedvenc írója Slawomir Mrożek. Leginkább a számítógépes hálózatok biztonsága érdekli.