



Vírusos levelek? Kizárva!

A levélalapú vírusok megállításának legjobb módja az, ha be sem eresztjük őket a hálózatunkra.

Múlt hónapban láthattunk egy módszert, hogy miként alkalmazhatjuk az Amavis csomagot a levélforgalom vírusellenőrzésére. Nagyobb hálózatoknál komoly gondot jelenthetnek a vírusok, főleg azért, mert ha egyszer bejutottak, akkor kegyetlen gyorsasággal végigfertőzhetik a munkagépeket. Manapság rendkívül fontos kérdés ez, és bár szerencsére a linuxos gépek esetében nem hallunk sokat vírusokról, a munkagépek védelmét is jellemzően kiszolgálóinkkal kell elősegítenünk, hiszen ezáltal rengeteg felesleges munkától mentjük meg magunkat. A belső hálózatot érő támadások leggyakoribb formái a levélvírusok. Az első lépés, amit egy rendszergazda általában tenni szokott ellenük: vírusvédelmi rendszert telepít a munkaállomásokra. Ez bölcs dolog, de számomra járhatóbb útnak tűnik, ha a vírusok rendszerbe jutását mindjárt a bejáratnál meggátoljuk. A vírusok, különösen a makróvírusok, messze leggyakoribb belépési pontja a szervezet levelezőrendszere. Mégis többnyire ez a vírusvédelmi rendszerek legelhanyagoltabb része. A piacon jelenleg kapható levélvírus-védelmi rendszerek gyakran alkalmazáshoz kötöttek, drágák, vagy mindkét tulajdonságot felmutatják (nem beszélve a megbízhatatlanságukról). Közepes méretű vállalkozás lévén a miénk, az idei költségvetésbe vírusirtó csomag beszerzését nem tervezték, így aztán csupán a Linuxhoz és a nyílt forráshoz fordulhattunk. Találtam is az Interneten néhány igen érdekes projektet, amely esetleg megfelelő lehetett volna az igényeinknek, de végül mégis a saját változat megírása mellett döntöttem. Azt szerettem volna elérni, hogy bármely felhasználó könnyen nyomom követhesse a rendszerünket, és egyszerűen bővíthesse, anélkül, hogy C- vagy Perl-guru lenne. A másik célom az volt, hogy a rendszer ki tudja azokat a hatékony eszközöket használni, amelyek általában minden linuxos alapterjesztésben megtalálhatók. E két tényező biztosítja a program hordozhatóságát, illetve, hogy más is képes legyen kezelni a rendszert a segítségem nélkül.

A rendszer alapjait Bash-héjprogramok, a `metamail`, a `grep`, az `Obtuse Systems SMTPd` termékei, a `Samba` és egy parancssoros víruskereső alkotják. Az 1. ábrán egy folyamatábra stílusú vázlatot láthatunk. Az `Obtuse Systems SMTP-tároló` és -továbbító csomagja ingyenesen hozzáférhető a <http://www.obtuse.com/smtpd.html> címen. E sorok írásának idejében a legfrissebb változat a 2.0. Az általam választott víruskereső a `McAfee Virus Scan for UNIX/Linux` volt, de számos másikat is választhattam volna. Ezek közül némelyek ingyenesek, mások nem. Mindenféleképpen olyat válasszunk, amelyek a keresés eredményének megfelelően állítja be a kilépési értékét, és amelyhez rendszeresen letölthetünk az újjlenyomat-frissítéseket. A rendszert felépíthetjük egy már meglévő linuxos tűzfalon, de akár egy külön gépen is, amennyiben linuxos tűzfal esetleg nincs kéznél. Ha erre a célra külön gépet használunk, nem kell túlságosan nagy teljesítményűnek lennie, egy 200 MHz-es 586-os 32 MB memóriával tökéletesen megfelel. A hálózatunk `SDSL` segítségével kapcsolódik az Internethez, a védelmet pedig egy IP-álcázást (`masquerading`) futtató `Mandrake` linuxos gép biztosítja. Ez a felépítés megkönnyíti a tűzfal telepítését.

A belső levelezőrendszer nem annyira fontos, elegendő, ha `SMTP` vagy `ESMPT` alatt működik. Mi például a `Novell Groupwise` termékét használjuk. Minden `SMTP`-forgalmat (25-ös kapu), ami a tűzfal `SMTP`-kapujára érkezik, át kell ahhoz a belső géphez irányítanunk, amelyen az `SMTP`-tűzfalat kiépítettük (vagy magához a tűzfalgéphez, mint a mi esetünkben is). Most lépünk tovább a tulajdonképpeni beállításokhoz! Az első lépés a könyvtárszerkezet felállítása. Az ide vonatkozó vázlatot a 2. ábrán találhatjuk. Rendszerünket a `/var/spool/smtpd` könyvtár alatt fogjuk felépíteni. Amennyiben átlagos levélforgalmunk meghaladja a napi 25 000 levelet, javasolom, külön lemezszt fűzzünk be a `/var/spool/smtpd` könyvtár alá. Az alapkönyvtár tehát a `/var/spool/smtpd` lesz. Ebben a könyvtárban öt alkönyvtárat hozunk létre: `incoming` (bejövő), `outgoing` (kimenő), `etc`, `bin` és `quarantine` (karantén). Először is váltsunk át rendszergazdai jogosultsággal rendelkező felhasználóra, majd gépeljük be a következő parancsot:

```
mkdir -p /var/spool/
smtpd/{etc,bin,incoming,outgoing,quarantine}
```

Következő lépésként állítsuk be a jogosultságokat, hogy a teljes könyvtárrendszert csak a `uucp`-felhasználó érhesse el, mivel az összes program e felhasználó jogosultságával fog futni. A következő parancsok megoldják számunkra a gondot:

```
chown -R uucp.uucp /var/spool/smtpd
chmod 700 /var/spool/smtpd
```

Most már beállíthatjuk a rendszer első összetevőjét. A korábban említett `Obtuse Systems` honlapjáról le kell töltenünk a `smtpd` csomagot. A letöltött fájl könyvtárában adjuk ki a következő parancsot:

```
tar -xzf smtpd-2.0.tar.gz
```

Ezután váltsunk az `smtp-2.0` könyvtárba és a `Makefile`-t szerkesszük át a következők szerint:

```
SPOOLDIR = /var/spool/smtpd
```

```
SPOOLSUBDIR = incoming
```

```
POLL_TIME = 300
```

```
PARANOID_SMTP = 1
```

```
JUNIPER_SUPPORT = 0
```

```
CHECK_IDENT = 0
```

Azt szeretnénk elérni, hogy az `smtpd` a leveleket az `incoming` alkönyvtárban tárolja, a `smtpd` pedig az `outgoing` alkönyv-

tárból olvassa őket. Hogy ezt lehetővé tegyük, az `smtpfwdd.c` fájlba a 75. sornál szúrjuk be a következő két sort:

```
// levelek let lt0se az outgoing alk nyvtÆrb l
#define SPOOLSUBDIR "outgoing"
```

Befejezésül fordítsuk le és telepítsük a csomagot a következő parancsokkal:

```
make
make install
```

A következő lépés az `/var/spool/smtpd/etc` könyvtár benépesítése néhány, az `smtpd` helyes működéséhez szükséges állománnyal. Másoljuk a `resolv.conf` fájlt a `/etc` könyvtárból a `/var/spool/smtpd/etc` könyvtárba, majd a `/etc` könyvtárból a `localtime` fájlt is másoljuk ide. Az `smtpd-2.0` terjesztés könyvtárból az `antirelay_check_rules_example` fájlt átmásolhatjuk a `/var/spool/smtpd/etc` könyvtárba. Amennyiben szükségünk van ilyesmire, az Obtuse System honlapján nézhetünk körbe további ellenőrző szabályokkal kapcsolatos utasításokért. Az `smtpd` program önműködő indításához a következő sort kell a `/etc/inetd.conf` fájlba helyezni:

```
smtpl stream tcp nowait root
  ↪ /usr/local/sbin/smtpl smtpl
```

Ezt a sort az esetleg már meglévõ `smtpl`-sorok helyére kell írni. Az `smtpfwdd` programot a `/etc/rc.d/rc.local` fájlból (vagy ahogy az `rc` fájlunkat éppen nevezik) kézzel kell elindítanunk. Fogjunk hozzá, és a következő sort írjuk be:

```
### Az smtplfwdd tovÆbb t d0mon ind tÆsa
  ↪ /usr/local/sbin/smtplfwdd
```

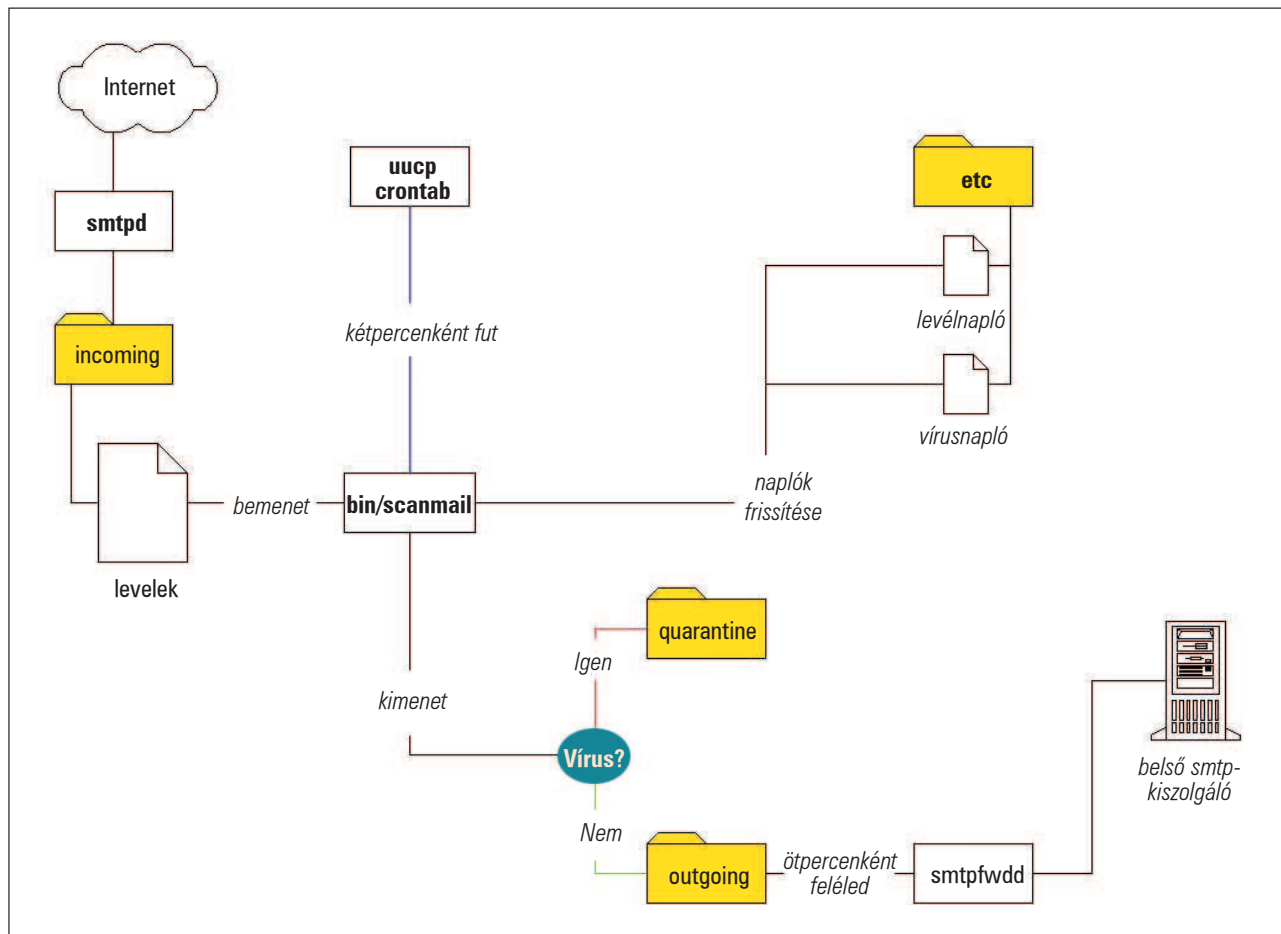
Végül le kell állítanunk minden esetleg még futó levél-továbbító ügynökprogramot (MTA-t). Ide értendők a *Postfix*, *Sendmail*, *Qmail* és társai. RedHat-rendszereken ezt egyszerűen a beállítóprogram segítségével is megtehetjük, amennyiben az összes MTA-t leállítjuk. Figyeljünk arra, hogy némely MTA például a Postfix vagy más folyamatok gyermekeiként futnak, így közvetlenül a `kill` paranccsal nem lehet őket „meggyilkolni”. A következő két parancs kiadásával indítsuk be az `smtpl` és az `smtplfwdd` démonokat:

```
kill -HUP `cat /var/run/inetd.pid`
  ↪ /usr/local/sbin/smtplfwdd
```

Mikor már futnak a démonok, ki is próbálhatjuk őket, ha levél-szűrő tűzfalunk 25-ös kapuján (ez az `smtpl`-kapu) elindítunk egy `telnet`-kapcsolatot:

```
telnet email.firewall.com 25
```

© Kiskapu Kft. Minden jog fenntartva



1. ábra A hálózati forgalom és a tűzfal felépítése

ahol az `email.firewall.com` a levélszűrő tűzfalunk neve. A következő visszajelzést kell kapnunk:

```
220 email.firewall.com SMTP ready,
Who are you gonna pretend to be today?
```

Ha bármilyen más üzenetet kapunk, valószínűleg elfelejtettük a kiszolgálón futó MTA-t kikapcsolni. A `ps -e` segítségével ezt könnyen kideríthetjük.

Nézzük csak, hol is tartunk? Ha minden jól ment, most van egy gépünk, amelyen az `smtpd` démon fut és leveleket fogad. Minden beérkezett levél egyszerű szöveges fájlként a `/var/spool/smtpd/incoming` könyvtárban tárolódik. Hogy valóban így is van-e, a következő parancsokkal nézzük meg:

```
$ telnet email.firewall.com 25
helo firewall.com
mail from: joe@firewall.com
rcpt to: fred@firewall.hu
data
Ez egy pr ba.
.
quit
```

Ne felejtjük el a levelünk törzsét egy egyetlen pontot tartalmazó sorral zárni! Ez mondja meg ugyanis a kiszolgálónak, hogy a levelet el akarjuk küldeni. Ha minden jól megy, a `/var/spool/smtpd/incoming` könyvtárban most egy szöveges fájl fogunk találni. Néhány percen belül a fájlunk el kell tűnnie, mi pedig egy levelet találunk a levelesládánkban. Az `smtpd` a leveleket `smtpd` formátumban menti, ahol `smtpd` egy véletlenszerűen készített üzenetazonosító.

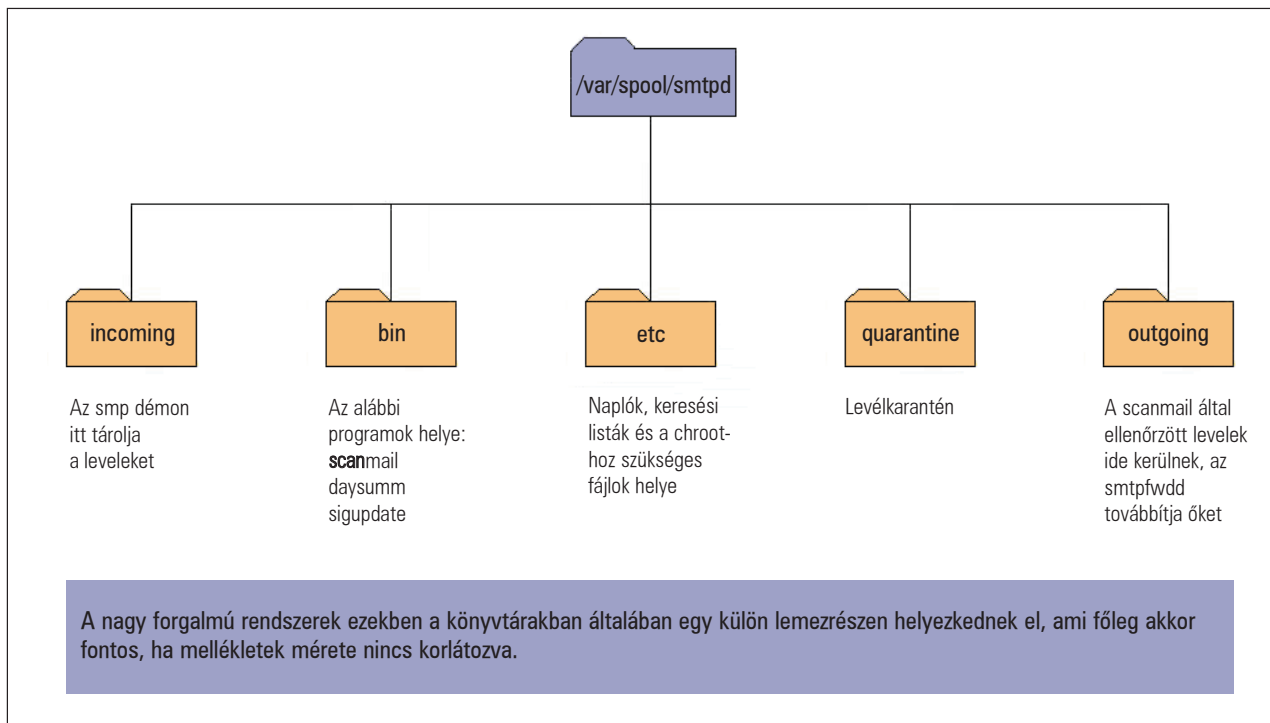
Az `smtpd` ezeket a szöveges fájlkat beolvassa, és továbbítja őket a célkiszolgálónak. Sikeres továbbítás után a fájl törlődik. De hogy is kerül a levél az *incoming* könyvtárból az *outgoing* könyvtárba? Nos, ez az a pont, ahol levélvizsgáló parancsfájlunk belép a képbe. A parancsfájl az *incoming* könyvtárban található fájlkat vírusra utaló nyomokat keresve egytől-egyig végignézi. Ha a fájl nem tartalmaz vírust, az *outgoing* könyvtárba kerül, ha viszont igen, a *quarantine* könyvtárba helyeződik át. Ilyen egyszerű az egész. A levélvizsgáló parancsfájlhoz azonban előbb szükségünk lesz egy működő víruskeresőre, tehát előbb erre a feladatra összpontosítsunk. Mindenféleképpen parancssoralapú víruskeresőre van szükségünk. Én a magam részéről – mint már említettem – a McAfee Virus Scan for UNIX/Linux rendszert választottam, így itt most erről fogok írni. A McAfee termék igen széles kilépőkód-listával rendelkezik, ezáltal a parancsfájlokba meglehetősen könnyen beilleszthető. A terméket egyszerűen beszerezhetjük <http://www.nai.com> honlap webloltján keresztül. A termék telepítése után bizonyosodjunk meg arról, hogy a `uucp` felhasználó által végrehajtható legyen. Ehhez be kell lépniünk a `/usr/local/uvscan` könyvtárba, majd ki kell adnunk a következő parancsot:

```
chmod -R 755
```

Próbáljuk ki, hogy működik-e! Váltuk át `uucp`-felhasználóra, és írjuk be a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Amennyiben a próba sikeresnek bizonyult, továbbléphetünk, és felfrissíthetjük a vírusujjlenyomatokat. Az ujjlenyomatfájlok (avagy a „definíciós” fájlok – ahogyan gyakran emlegetik őket) alkotják az összes víruskereső velejét. Ezért fontos, hogy



2. ábra A könyvtárszerkezet

a frissítés időről időre önműködő módon megtörténjen. Nagyon fontos, hogy ezeket a frissítéseket legalább kéthetente ellenőrizzük, mivel havonta legalább három új változat jelenik meg. Ehhez a megoldáshoz először egy `sigupdate` Bash-parancsfájl fogunk alkotni, amelyet majd a `/var/spool/smtpd/bin` könyvtárba helyezünk el. Akárcsak eddig, most is győződjünk meg arról, hogy az `uucp`-felhasználó-e a parancsfájl birtokosa, és hogy a fájl végrehajthatónak jelöltük-e be. A parancsfájl tartalmát az 1. listában (24. CD Magazin/Vírus könyvtár) találjuk, elég könnyen követhető. A `sigupdate` fájl hetente egyszer fogjuk futtatni, lehetőség szerint a nap valamilyen kevésbé terheltségi szakában. Ezt később tesszük meg egy `crontab` sor beillesztésével. Továbbá készítenünk kell a `uucp`-felhasználó saját könyvtárában egy `.netrc` nevű fájl. Szerkesszük át úgy, hogy a következőképpen nézzen ki:

```
machine ftp.nai.com
login anonymous
password admin@domain.com
macdef init
cd pub/antivirus/datfiles/4.x
bin
prompt
mget dat-*.tar
close
bye
```

A `.netrc` fájl előre meghatározott gazdagépek FTP-elérését vezérli. Ez a legjobb módja annak, hogy FTP-folyamatunkat önműködővé tegyük. A `.netrc` írásmódról az FTP súgóoldalon olvashatunk. Lássunk neki, és amint elkészült a két fájl, futtasuk le a `sigupdate`-et.

A frissítés végén futtassuk le a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Nézzük meg a vírusadatfájl létrejöttének dátumát. Valamilyen közeli időpontot kell látnunk, általában egy hónapnál nem régebbit. Ha nem így lenne, parancssorból kell ellenőriznünk, hogy a `sigupdate` helyesen működik-e.

Most lépünk tovább a fő levélvizsgáló parancsfájlokra. Ezt a fájl `scanmail`-nek fogjuk hívni és a `/var/spool/smtpd/bin` könyvtárba kerül. Ez a parancsfájl fogja végrehajtani az összes közvetlen műveletet az `smtpd` által létrehozott levélszövegfájlok között. Készítsünk egy fájl, majd tegyük végrehajthatóvá és adjuk át az `uucp`-felhasználónak. Első körben azokat a csatolt fájlokat szűrjük ki, amelyek nyilvánvalóan rosszak. Az ehhez a kereséshez tartozó minták a `matches.bad` fájlban tárolódnak, amelyet később fogunk létrehozni. Ha a `grep` talál valamit, a levél a karanténba kerül, és egy levelet küldünk a rendszergazdának, amelyben megtalálható a dátum, a fájlnev, és hogy a levél kinek, illetve kitől érkezett.

A 19. sortól kezdve a `scanmail` előbb belép az `incoming` könyvtárba, és az ott található fájlneveket egy vektorban helyezi el. Ezután minden egyes fájlneven végiglépdel, és a `grep` segítségével a fájlokban adott mintákat keres. Minden levél tartalmát kétszer ellenőrizzük. Első körben azokat a csatolt fájlokat szűrjük ki, amelyek nyilvánvalóan rosszak. Az ehhez a kereséshez tartozó minták a `matches.bad` fájlban tárolódnak, amelyet később fogunk létrehozni. Ha a `grep` talál valamit, a levél a karanténba kerül, és egy levelet küldünk a rendszergazdának, amelyben megtalálható a dátum, a fájlnev, és hogy a levél kinek, illetve kitől érkezett. Ha nem volt találat, jöhet a második kör. Ez esetben a `grep` a `matches.doc` nevű fájl fogja használni, hogy kiszűrje azokat a csatolt állományokat, melyek makróvírusokat vagy beágyazott

vírusokat tartalmazhatnak. Ha talál valamit, a csatolt részt a `metamail` program segítségével egy dinamikus létrejövő ideiglenes könyvtárba bontja ki. Az ideiglenes könyvtár neve a csatolt állomány neve lesz "`_d`" utótaggal kiegészítve. Az ideiglenes könyvtár tartalmát ezután a parancssoros víruskeresőnkkel végignézzük.

Ha valamilyen vírust találtunk (ezt a kereső visszatérési értékéből tudjuk meg), a `scanmail` a levelet és a csatolt állományokat karantén alá helyezi, majd figyelmeztető levelet küld a rendszergazdának. Egyúttal udvarias levelet küldünk a levél feladójának, amelyben értesítjük, hogy érdemes lenne végignézzni a rendszerét, illetve megadjuk a talált vírus nevét.

Ha ez idáig nem találtunk vírust, a levél az `outgoing` könyvtárba kerül, ahonnan az `smtpd` a belső levélkiszolgálóhoz továbbítja majd. Az `smtpd` a kimenő könyvtárat ötpercenként egyszer ellenőrzi.

A következő lépés annak a fájlnevlistának az elkészítése, amelyet a `scanmail` a gyanús csatolt állományok felderítéséhez fog használni. A `scanmail` a fájlok közt a `grep` eszköz segítségével keres. Kihaszaljuk a `-f` kapcsoló nyújtotta előnyöket, mivel így a `grep` a kereséshez használt mintákat egy megadott szöveges fájlból fogja kiolvasni. A szöveges fájl szerkezete igen egyszerű, minden sorban egy minta található. A `grep` a fájlban felsorolt bármely mintával való egyezést találatként fogja értékelni. Váltunk a `/var/spool/smtpd/etc` könyvtárba és hozzunk létre két fájl `matches.bad` és `matches.doc` néven. A `matches.bad`-be az olyan fájlok névmintáit helyezzük, amelyeket semmiféleképpen nem szeretnénk anélkül a rendszerbe eresztetni, hogy a rendszergazda meg ne vizsgálta volna őket. A `matches.doc` fájlnak ezzel szemben azokat a dokumentummintákat kell tartalmaznia, amelyek beágyazott vírusokat tartalmazhatnak, ilyenek például a Word-dokumentumok és a táblázatkezelők állományai. Amikor ezeket a fájlneveket hozzuk létre, minden sorban a `filename=.*\.` formátumot használjuk. Ez azért szükséges, hogy ne kapjunk hamis riasztásokat a mimekódolás olyan véletlen karaktorsorozatai miatt, amelyek történetesen megegyeznek a `grep` által keresett mintával. Figyeljünk arra is, hogy ez a fájl semmiképpen ne tartalmazzon üres sort, hiszen a `grep` az üres sort is keresendő mintának fogja venni, és minden levélre találatot fogunk kapni. A Vim jó szerkesztőprogram e célra, mivel könnyen láthatjuk a benne szereplő üres sorokat. Az általam használt fájlok tartalmát a 3. listában (24. CD Magazin/Vírus könyvtár) lelhetjük fel.

A másik felhasznált parancsfájl neve `daysumm` lesz és a `/var/spool/smtpd/bin` könyvtárban helyezzük el. Hozzuk létre ezt az állományt a 4. listának megfelelően.

A `daysumm` a napi tevékenységről értesítőt küld a rendszergazdának. Megmutatja, hány levél érkezett aznap, közülük hány volt vírusos, illetve melyek voltak ezek a vírusok. A cronban állítjuk be, hogy minden este 11:59-kor fusson le.

A `daysumm` parancsfájl a `/var/spool/smtpd/etc` könyvtárban elhelyezett `virus.$date` és `email.$date` fájloktól függ. Ezek szöveges fájlok, amelyek dinamikusan jönnek létre, a `scanmail` frissíti őket és dátumfüggők.

Emiatt a `daysumm` programot mindig éjfél előtt kell lefuttatnunk, különben az időbélyeg (timestamp) megváltozik és rossz fájlok kerülnek beolvasásra.

Biztos észrevették már, hogy valahányszor magáról a tűzfalról küldtünk ki üzenetet – például a parancsfájlokban is –, mindig egy `sendmail -q` parancsot is kiadunk. A `-q` ugyanis azt mondja meg a `sendmail`-nek, hogy induljon el, és nézzen körül, van-e kimenő üzenet, ha van, küldje el, majd lépjen ki. Ez hatékonyan kiüríti az összes kimeneti sort, ami azért szük-

4. lista A daysumm parancsfájl

```
#!/bin/sh
mailto="admin@domain.com"
timestamp='date +%A - %B %d %Y'
etc="/var/spool/smtpd/etc"
date='date +%m%d%Y'
email_total='cat $etc/email-log.$date'
virus_total='cat $etc/virus-log.$date'

mail -s "DAILY E-MAIL SUMMARY" $mailto <<eoi

Date:      $timestamp
E-Mail total: $email_total
Virus total: $virus_total

eoi

/usr/sbin/sendmail -q

exit 0
```

séges, mert többé semmilyen MTA nem fut a gépünkön. Az *smtpd* csomag nem MTA, hanem egy tároló- és továbbító-csomag. Úgy is elképzelhetjük, mint egy kifejezetten levéltovábbításra kihegyezett programot. E külön parancs nélkül tehát soha egyetlen levelet sem kaphatnánk meg a tűzfalról. Itt az ideje, hogy az egész folyamatot a cron démon segítségével önműködővé tegyük. Ezt a uucp-felhasználó személyes crontab állományának felhasználásával fogjuk megtenni. Lépjünk be uucp-felhasználóként, majd adjuk ki a crontab -e parancsot, ami az uucp-felhasználó cron tábláját nyitja meg szerkesztéshez. A scanmail, daysumm és sigupdate parancsfájlok részére hozzuk létre a következő bejegyzéseket:

```
MAILTO=""

# A scanmail parancsfájl minden kőt percben
# fusson le
*/2 * * * * /var/spool/smtpd/bin/scanmail

# A daysumm parancsfájl minden nap 11:59-kor
# induljon el
59 23 * * * /var/spool/smtpd/bin/daysumm

# A sigupdate minden cs t rt k n 4:00-kor
# fusson le.
0 16 * * 4 /var/spool/smtpd/bin/sigupdate
```

Természetesen a futásidőpontokat megváltoztathatjuk úgy, hogy megfeleljenek az igényeinknek. Az, hogy a scanmail-t milyen sűrűn futtassuk le, főként a napi levélforgalmunktól függ. Ha a napi mennyiség tízezer levél felett van, a magam részéről az időközök két percben határoznom meg, az smtpfwd-t pedig ötpercenkénti futásra állítanom be. Így nem küldünk egyszerre hatalmas levélcsomagokat a belső kiszolgálóra. Ha naponta 1000 vagy ennél kevesebb levéllel kell csak számolnunk, elég, ha a scanmail minden tízedik percben fut le, az smtpfwd pedig ötpercenként néz körül. Ne feledjük el a MAILTO=""

kifejezést kitenni a crontab-bejegyzések elejére! Ez azért szükséges, hogy a crond a végrehajtott cron-feladatokról ne küldjön levelet az uucp-felhasználónak. Ha minden két percben egy levél érkezik, az gyorsan felgyülemlik, és az uucp-felhasználó soha nem ellenőrzi a leveleit. Felállítottam egy Samba-megosztást is, hogy a windowsos gépekről is hozzáférhessek a */var/spool/smtpd* könyvtárszerkezet-hez. A főként Windowst használó rendszergazdáknak ez a beállítás hasznos lehet – így nem kell mindig SSH-kapcsolatot nyitnom, valahányszor vírusra figyelmeztető levelet ellenőrzök. A következő sorokat kell a */etc/smb.conf* fájlba illeszteni:

```
[mail-gate]
Comment = Levél-tűzfal k nyvtárak
Path = /var/spool/smtpd
Valid users = nev nk
Admin users = nev nk
Browseable = no
Read only = no
```

ahol a „nevünk” természetesen a Samba-felhasználói nevünket jelenti. Ami még hiányzik ahhoz, hogy tűzfalunkról az összes bejövő SMTP-kapcsolatot az új levélszűrő kiszolgálónkra irányítsuk, az, hogy tűzfalunknak IP-álcázást kell használnia. Egyszerűen adjuk ki a következő parancsot:

```
ipmasqadm portfw -a -P tcp -L tbfzal 25
-R c0lg0p 25
```

ahol a tbfzal a tűzfalunk címe, a c0lg0p pedig az új levélszűrő gépünk címe. Amennyiben levélszűrő tűzfalunkat közvetlenül a már meglévő tűzfalunkon szeretnénk futtatni, semmin sem kell változtatnunk. Ha más típusú tűzfalrendszert használunk, olvassuk el a leírást, hogy megtudjuk, miképpen állíthatjuk be a kapuátírányítást. Remélhetőleg nem okoz nagy gondot, de előfordulhat, hogy kapcsolatba kell lépniünk a termék készítőjével. Ne feledjük el ezt az átírányító parancsot betenni az indító parancsfájlokba, hogy túlélje a rendszerindításokat. Ha minden jól ment, végre működő levélvírusszűrő tűzfallal rendelkezünk. Próbáljunk meg küldeni magunknak néhány próbaüzenetet valamelyik ingyenes webes levelezőszolgáltatótól, hogy lássuk, minden jól működik-e. Én például adott időközönként küldök magamnak egy makróvírussal fertőzött állományt, hogy lássam, a rendszer még mindig helyesen működik-e. Sok dologgal lehetne még bővíteni ezt az alaprendszert. Különösen ígéretes a daysumm parancsfájl, amelyet jócskán fel lehetne még fejleszteni. Jelenleg épp egy CGI-parancsfájlon dolgozom, amely a pillanatnyi átlagokat – például az átlagos napi mennyiséget – az Interneten keresztül jelenítené meg. Természetesen ez csak egy út a több százból, ahogyan ez a rendszer levelezőrendszerünket megvédeheti, anélkül, hogy befolyásolná a cég költségvetését. Ha valaki esetleg kitalálna valamilyen ügyes továbbfejlesztést a rendszerhez, kérem, tudassa velem. Nagyon szeretnék hallani róla.



Dave Jones
(davidashleyjones@hotmail.com)
három évig volt hálózati rendszergazda az alabamai Birminghamben. Amikor éppen nem a számítógép előtt ül, egy-egy szál jófajta dohányt szív el, vagy a feleségével és a lányával X-aktákat néz a tévében.