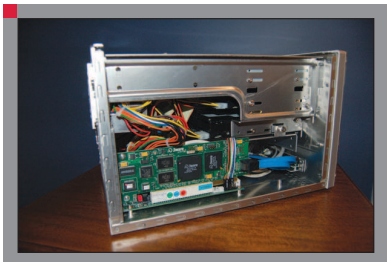
Egy megbízható, automatikus adatmentési megoldás

Ebben a cikkben bemutatom, miként alakíthatunk ki felügyelet nélkül működő, titkosított, redundáns elemekből álló hálózati adatmentési rendszert Linux operációs rendszerrel, a Duplicity szoftverrel, s mindezt közönséges, kereskedelmi forgalomban kapható hardverelemekből.

Manapság teljesen általános, hogy a felhasználók óriási merevlemezeket használnak, amelyeket persze dugig töltenek filmekkel, zenékkel, digitalizált videókkal, szoftverekkel, dokumentumokkal, és mindenféle más formátumú adattal. Ráadásul sokan teljesen elhanyagolják a *CD*-re és *DVD*-re történő adatmentést, aminek lélektanilag talán az lehet a legfőbb oka, hogy ennél az adatmentési megoldásnál rengeteg apró tennivalója akad a felhasználónak. Szinte biztosan meg kell birkóznia például olyan problémákkal, mint az adathordozó méretéből adódó korlátok „kerülgetése”, vagy az adatok épségének ellenőrzése. És akkor a manuális lemezcserélgetést még nem is említettem. Ennek megfelelően ezeknek az adatoknak a többségét nem is mentik a tulajdonosok, vagy legalábbis nem rendszeresen. Jómagam biztonsági szakértőként dolgozom, legfőképpen a szoftverfejlesztés területén, szabadidőmben pedig nyílt forrású szoftvereket fejlesztek. Ez utóbbi munkám keretében immár számos projektben vettem részt. Tekintettel meglehetősen széles érdeklődési területemre otthon egy 12 gépből álló hálózatom van, amelynek tagjain ***Linux***, ***Mac OS X*** és ***Windows*** is fut.
Az elmondottak alapján talán érthető, hogy a munkám eredményének megsemmisülése számomra nem elfogadható alternatíva. Ahhoz, hogy az én munkakörnyezetemben egy adatmentési rendszer valóban jól működhessen, több különböző gépen dolgozó felhasználót, és persze többféle operációs rendszert kell támogatnia. Valamennyi felhasználónak képesnek kell lennie a számára fontos adatok mentésére, illetve szükség esetén az önálló visszaállítására. Röviden tehát a rendszernek képesnek kell lennie a többfelhasználós, felügyelet nélküli működésre. Ha egy ilyen rendszer jól van kialakítva, akkor a felhasználó nem csak egy teljes mentést tud szükség esetén visszaállítani, hanem akár egyes fájlokat is, függetlenül attól, hogy azok mikor kerültek az archívumba. A többfelhasználós működés miatt nyilván szükség lesz egy megfelelően átgondolt biztonsági rendszerre is. Ennek része kell legyen a hozzáférés szabályozása, illetve az adatintegritás felügyelete is. Az előbbi abban akadályozza meg a felhasználókat, hogy egymás bizalmas adataihoz illetéktelenül hozzáférjenek, arról másolatot készítsenek, míg az utóbbi azt hivatott biztosítani, hogy visszaállításkor az adatok valóban a mentéskor érvényes állapotukban kerüljenek vissza rendeltetési helyükre.
A biztonság mellett egy adatmentési rendszernek a megbízhatóság is igen lényeges tulajdonsága. A megoldásnak ennek szellemében tolerálnia kell a hardver esetleges meghibásodásait. Mivel egy adattárolási rendszer legnagyobb valószínűséggel meghibásodó eleme a merevlemez, gondoskodnunk kell a megfelelő hibatűrésről. Végezetül a megoldásnak hatékonyan kell kihasználnia a rendelkezésre álló tárterületet, illetve hálózati sávszélességet. A sávszélesség ügyes elosztásával egyszerre több felhasználó férhet hozzá a rendszer szolgáltatásaihoz, míg a tárterülettel való hatékony gazdálkodás eredménye nyilván a tárolható adatok mennyiségében fog megmutatkozni. Mint minden munkámmal kapcsolatban, természetesen itt is törekedtem arra, hogy a végeredmény vizuálisan is vonzó illetve kellően kicsi legyen, és persze az ára is az „elfogadható” kategóriába essen.
Először azzal próbálkoztam, hogy egy már létező megoldást találjak, amit készen kapok. Találtam is számos jelöltet, amelyek nagyjából két kategóriába sorolhatók: vannak egylemezes hardveres hálózati adatmentési megoldások, és léteznek ugyanilyen ***RAID*** tömbök. Az első kategória egyik maga nemében kiváló képviselője a ***Western Digital NetCenter*** nevű terméke. Ugyanakkor hamar felismertem, hogy egyetlen, ebbe a csoportba tartozó megoldás sem rendelkezik azokkal a tulajdonságokkal, amelyeket az imént felsoroltam. Nincsenek vagy hiányosak a biztonsági szolgáltatások, rossz a sávszélesség kihasználása, vagy egyszerűen csak nem kellően megbízható a termék. A második csoportba tartozó eszközöket láthatóan inkább céges, sem mint magáncélú

1. ábra A Silver Venus 668 ház (elölnézet)

2. ábra A Silver Venus 668 ház (hátsó nézet)



3. ábra A Silver Venus 668 ház (belülről a már beszerelt hardverelemekkel)

felhasználásra tervezték. Ennek egyik kellemetlen mellékhatása, hogy általában jóval drágábbak, mint az első csoport tagjai. A ***Snap Server 2200*** kiváló példa ennek a kategóriának az alacsony árfekvésű részére. Az eszköz ára 1000 dollártól indul, viszont tény, hogy tekintélyes nagyságú tárterület alakítható ki vele. Ezzel együtt a drága készülékekről is azt voltam kénytelen megállapítani, hogy csak hellyel-közzel tesznek eleget a biztonsággal, teljesítménnyel, vagy egyéb szolgáltatásokkal kapcsolatos általános elvárásaimnak.
Mivel a keresés végeredménye az volt, hogy a könnyen és gyorsan elérhető megoldások között egyetlen számomra megfelelő sincsen, úgy döntöttem, hogy magam fogok megépíteni egy ilyen készüléket.
A rendszeremnek biztonságosnak és megbízhatónak kell lennie, titkosítást kell használnia, Linux operációs rendszerrel kell működnie, és végül, de nem utolsó sorban olyan elemekből kell állnia, amelyek közönséges kereskedelmi forgalomban is kaphatók (***Commercial Off-The-Shelf***; ***COTS***).
Azt már az elején eldöntöttem, hogy magát az adatmentést és kezelést a ***Duplicity*** nevű csomaggal fogom megoldani. Ezekkel az eszközökkel és elvárásokkal végül is valóban sikerült megépítenem egy olyan hálózati eszközt, amely egyaránt képes teljes és inkrementális mentéseket készíteni, az adatokat pedig digitális aláírással titkosítva tárolja. Az inkrementális mentés olyan adatmentési megoldás, amelynél csak a legutolsó mentés óta megváltozott tartalmak kerülnek be az aktuális mentésbe.
A teljes mentésnél ezzel szemben az eredeti adathordozó teljes tartalmáról másolatot készítünk, függetlenül az egyes fájlok korától. Ami a visszaállítást illeti, a rendszerem a teljes adatvisszaállítás mellett képes arra is, hogy megadott fájlok egy megadott időben készült másolatát visszaírja az eredeti adathordozóra. Utóbbira számos esetben lehet szükségünk. Tegyük fel például, hogy kaptam egy vírust, de azt is pontosan tudom, hogy a kártevő egy hete még nem volt ott a rendszeremen. Az egyedi adatmentési megoldásommal simán megtehetem, hogy visszaállítom a megfertőződött rendszer egy héttel, egy hónappal korábbi állapotát, vagy akár a legelső mentéskor érvényeset.
A ***Duplicity*** a projekt hivatalos weblapja szerint egy olyan adatmentési megoldás, amely teljes könyvtárakról készít tar formátumú, titkosított mentéseket, majd azokat feltölti egy helyi vagy távoli fájlkiszolgálóra.
Az általam elkészített adatmentési megoldás sarokköve is ez az alkalmazás volt. A ***Duplicity*** támaszkodik a ***librsync*** és a ***GnuPG*** könyvtárakra, valamint számos más fájlátviteli mechanizmusra. Ez volt az a rendszer, amely rendelkezett mindazokkal a képességekkel, melyek az általam megálmodott megoldás funkcionalitásának biztosításához elengedhetetlenek voltak. Volt benne biztonság, hatékonyan működött, egyszerűen mindent tudott, amit csak akartam.
A ***Duplicity*** először a ***librsync*** segítségével elkészít egy tar formátumú kötetet, amely vagy egy teljes, vagy egy inkrementális mentést tartalmaz. Azután ezt az adattömböt a ***GnuPG*** segítségével titkosítja és digitálisan aláírja, ami egyszerre biztosítja az adatok integritását és idegen számára való hozzáférhetetlenségét. Amikor készen van a titkosított mentés, a ***Duplicity*** a megadott helyre továbbítja azt számos adatátviteli mechanizmusainak egyikével.
Jómagam az ***SSH***-n keresztül történő fájlátvitelt használtam, mivel az adatok így a másolás során is titkosított formában áramlanak. Erre tulajdonképpen már nem is lenne szükség, hiszen maguk az átvinni kívánt adatok eleve titkosítottak, viszont a biztonságos átvitel még egy szinttel nagyobb összetettséget kölcsönöz a rendszernek, ami egy esetleges támadó számára értelemszerűen újabb akadály. A döntésnél persze az is lényeges szempont volt, hogy ***SSH*** gyakorlatilag minden gépen fut, így nem kell újabb hálózati szolgáltatásokat, például ***FTP***, ***NFS*** vagy ***rsync*** szervert beüzemelni.

A hardver

Miután eldöntöttem, hogy belevágok, és megépítem magamnak ezt a hálózati adatmentő eszközt, a következő lépésben el kellett döntenem, hogy milyen hardverelemekből fogok építkezni. Figyelembe véve a megvalósítani kívánt funkciókat, valamint a megbízhatósággal, a biztonsággal és a teljesítménnyel kapcsolatos elvárásaimat, világos volt, hogy egy ***RAID 1***-es – tükrözött – tömbön alapuló hálózati megoldást kell megépítenem. Mindez azt jelentette, hogy két merevlemezre volt szükségem, meg persze egy olyan ***RAID*** kártyára, ami legalább két meghajtót képes vezérelni.
Ami az alaplapot illeti, ott rögtön a kis formafaktorú változatok körül kezdtem el nézelődni. Korábban számos esetben használtam már ***Mini-ITX*** rendszerű alaplapokat, így pontosan tudtam, hogy ezek linuxos támogatottsága csaknem teljesnek mondható. Tekintettel arra, hogy ehhez az eszközhöz fölösleges lett volna valamilyen erős processzort választani, az ***EPIA Mini-ITX ML8000A***

alaplap mellett döntöttem, amelyen egy 800 MHz-es processzor, egy 100 Mb-es hálózati csatoló és egy 32 bites *PCI* foglalat található. Ezek a paraméterek tökéletesen megfeleltek a számítási sebességgel és hálózati átvitellel kapcsolatos elvárásaimnak, és ott volt a *PCI* csatoló is a *RAID* kártyának.
Megvolt tehát a megfelelő formafaktorú alaplapom, így választanom kellett egy hozzá illő házat és tápegységet. Itt nyilván fontos szempont volt, hogy a házban a *Mini-ITX* alaplap mellett azért el kell férnie a *RAID* vezérlőnek, meg két teljes méretű merevlemeznek is, miközben az általános esztétikai igényeimről sem feledkezhetek meg. Nos, a választás itt nehéznek bizonyult. Egészen sok *Mini-ITX* ház tulajdonságait böngésztem át, mire ráakadtam az igazira, amely a maga nemében egyetlennek is bizonyult. A nagy ő a *Silver Venus 668* lett, amely kellően sokrétű ahhoz, hogy minden igénynyemnek megfeleljen. Megvolt tehát az alaplap és a ház. A következő lépés a memória kiválasztása volt. Úgy döntöttem, hogy 512 MB DDR266-os *RAM* tökéletesen elegendő lesz.
Azért végül mégis akadt egy kis problémám: furcsa kimondani, de nem volt éppen egyszerű *Mini-ATX* alkatrészeket forgalmazó céget találni az *Egyesült Államokban*. Aztán végül mégis találtam egyet, a *Logic Supply* nevűt, amelynél az alaplapot, a házat és a memóriát is megtudtam rendelni, összesen 301,25 dollárért, amiben már a szállítási költség is benne volt. Ezen a ponton tehát a kezemben volt minden alkatrész, kivéve a *RAID* vezérlőt.
És itt jött az igazi probléma. A minden tekintetben megfelelő *RAI*D kártya megtalálása kifejezetten nehéznek bizonyult. Először is számos olyan *RAID* vezérlő van, amelyik a munka dandárját nem hardverből valósítja meg, hanem az operációs rendszer által futtatott meghajtóval végezteti el. A választás végül *3ware 8006-2LP SATA RAID* vezérlőre esett, amely két *SATA* vezérlővel rendelkezik, a vezérlést pedig teljesen a kártyán levő chipek végzik. Ezt az eszközt a *Monarch Computer Systems*-től vettem meg 127,83 dollárért (az ár ismét tartalmazta a szállítás költségét is).

Most már tényleg csak a merevlemezek hiányoztak. Némi töprengés után végül két 200 GB-os *Western Digital #2000JS SATA300* márkajelű lemezt vettem, melyek 8 MB cachesel rendelkeznek. Ezeket a *Bytecom Systems Inc*-től rendeltem meg, szállítással együtt összesen 176,69 dollárért. Megvolt tehát minden hardverelem a rendszer megépítéséhez a végösszeg pedig 604,77 dollárra rúgott. Összehasonlításként a legegyszerűbb készen kapható *RAID* vezérlővel szerelt hálózati adatmentő egységek 1000 dollárnál kezdődnek, ráadásul nem is nyújtanak annyit, mint amennyit az én rendszerem fog tudni.

A fájlkiszolgáló

Miután megépítettem magát a gépet, el kellett döntenem, hogy milyen operációs rendszer fut majd rajta. Választásom a *Debian stable 3.1r2* terjesztésre esett, elsősorban a kiváló csomagkezelés miatt. Telepítettem egy *SSH* démont is, hiszen ezen keresztül lehet majd elérni a kiszolgálót. Miután ezzel is megvoltam, létrehoztam magamnak egy felhasználói fiókot. A mentett adatok minden felhasználó saját könyvtárába fognak kerülni, vagyis mindenkinek, aki adatokat szeretne menteni a rendszerre, rendelkeznie kell rajta egy fiókkal.

Az ügyfelek beállítása

A kiszolgáló ezzel tulajdonképpen üzemkész is volt, a következő lépésben tehát a hálózat többi gépét kellett megfelelően beállítanom. Mivel a *Duplicity* – mint korábban is említettem – a *GnuPG*-re és az *SSH*-ra támaszkodik, ezeket a szolgáltatásokat úgy kell üzemeltetni, hogy rendszergazdai beavatkozás nélkül legyen képesek együttműködni a *Duplicity*-vel. A következő szakaszokban ismertetem azt a konfigurációt, amit a hálózati adatmentési kiszolgálót használó gépeken valósítottam meg.

A Duplicity telepítése

A *Duplicity* rendszert egyszerűen a *Debian Linux* `apt-get` parancsával telepítettem rendszergazdaként a következőképpen:

```
# apt-get install duplicity
```

Hitelesítés SSH kulcsokkal

Amint telepítettem a *Duplicity*-t, létrehoztam egy *DSA* kulcspárt, és úgy állítottam be az *SSH*-t, hogy ezt használva hitelesítse a felhasználókat. Ebben a módszerben értelemszerűen az a jó, hogy bejelentkezéskor nem kell jelszót megadni. Ezen a ponton talán érdemes megjegyezni, hogy egyesek a kulcspárral való hitelesítést úgy használják, hogy magukat a kulcsokat sem védik jelszóval. Ez biztonsági szempontból rendkívül kockázatos, hiszen bárki, aki megkaparintotta a belépéshez használt kulcsot, attól kezdve ugyanolyan jogosultságokkal rendelkezik, mint mi magunk. A kulcsokat tehát mindenképpen érdemes jelszóval védeni, amit a használat megkezdése előtt a rendszer ellenőriz. Az *SSH DSA* kulcspár előállításához a következő parancsokat futtattam az ügyfélgépen:

```
$ ssh-keygen -t dsa
$ scp ~/.ssh/id_dsa.pub
➥<username>@<server>:
$ ssh <username>@<server>
$ cat id_dsa.pub >> ~/.ssh/
➥authorized_keys2
$ exit
```

A kulcspárt ténylegesen az első parancs állítja elő. A másodikkal a már kész nyilvános kulcsot átmásoljuk a mentéshez használt kiszolgálóra. A harmadikkal elindítunk a kiszolgálón egy távoli parancsértelmezőt, a negyedikkel pedig hozzáfűzzük a most előállított nyilvános kulcsot a használható kulcsok listájához. Ez az a lépés, amivel végül is engedélyezzük, hogy a két gép között a hitelesítés kulcspár használatával történjen. Végezetül az ötödik paranccsal kilépünk a távoli parancsértelmezőből.

A GnuPG kulcsok beállítása

Az *SSH* kulcsok előállítása és beüzemelése után még elő kellett állítanom egy olyan *GnuPG* kulcspárt is, amivel a *Duplicity* a mentett adatokat fogja titkosítani. Ezeket a kulcsokat közönséges felhasználóként, az ügyfélen hoztam létre. Mivel a titkosító kulcsok egy közönséges felhasználói fiókhoz tartoznak, nem lehet velük a teljes fájlrendszert lementeni.
Ha valaha mégis szükség lenne arra,

hogy teljes mentést tudjak készíteni, akkor sincs más dolgom, mint rendszergazdaként bejelentkezve előállítani egy másik *GnuPG* kulcspárt. A *GnuPG* kulcsok létrehozásához tehát a következő parancsot használtam.

```
$ gpg --gen-key
```

A kulcskarika

Amint a *GnuPG* és az *SSH* kulcsok egyaránt elkészültek, az első dolgom az volt, hogy írtam egy *CD*-t, amire kimásoltam valamennyit. Aztán telepítettem és beállítottam a *Keychain* nevű alkalmazást. Ez egy olyan program ami a hosszan futó *ssh-agent* és *gpg-agent* példányokat képes kezelni úgy, hogy ne kelljen megadni a kulcsokhoz tartozó titkosító jelszót minden egyes olyan program indításakor, amelynek szüksége van a kulcsokra. *Debian* előtt először telepítenem kellett a *keychain* és az *ssh-askpass* nevű csomagokat. Aztán a */etc/X11/Xsession.option* nevű fájlban ki kellett kommenteznem az *ssh-agent* kifejezést tartalmazó sort, hogy az ügynökprogram ne indul el, valahányszor elindítok egy új *X* munkamenetet az *Xsession* segítségével. Aztán a saját könyvtáramban található *.bashrc* fájlba beleírtam a következő néhány sort, hogy a *Keychain* megfelelően induljon el:

```
/usr/bin/keychain ~/.ssh/
➥id_dsa 2> /dev/null
source ~/.keychain/`hostname`
➥-sh
```

Végül a *gnome-session* beállításaim közé fölvettem egy *xterm* indítást. Az *xterm* automatikusan elindítja a *Bash* héjat, ez induláskor elolvassa a *.bashrc* tartalmát, és elindítja a *Keychain* alkalmazást. A *Keychain* induláskor ellenőrzi, hogy a kulcsok bekerültek-e már a gyorstárba. Ha nem, akkor egyetlen egyszer megkérdezi a kulcsot titkosító jelszót, valahányszor elindítom a gépemet és bejelentkezem.

A Duplicity használata

Ha a *Keychain* is tökéletesen működik, már nincs is más hátra, mint használatba venni az új rendszert. A *Duplicity* segítségével immár bármely könyvtáramról felügyelet nélkül biztonsági másolatot készíthetek az adatmentési kiszolgálóra úgy, hogy egy *cron* feladattal automatizálom a program indítását. Próbaképpen a teljes *home* könyvtáramról készítettem egy biztonsági másolatot a következő paranccsal:

```
$ duplicity --encrypt-key
➥AA43E426 \
--sign-key AA43E426 /home/
➥username \
scp://user@backup_serv/backup/
➥home
```

Miután lezajlott a mentés, a következőképpen ellenőriztem annak helyességét:

```
$ duplicity --verify --encrypt
➥-key AA43E426 \
--sign-key AA43E426 \
scp://user@backup_serv/backup/
➥home \
/home/username
```

Most tegyük fel, hogy az ügyfélgépen véletlenül letöröltem a *home* könyvtáram teljes tartalmát. Az adatmentési kiszolgálóról való visszaállításhoz a következőt kell tennem:

```
$ duplicity --encrypt-key
➥AA43E426 \
--sign-key AA43E426 \
scp://user@backup_serv/backup/
➥home \
/home/username
```

Igen ám, csakhogy normális esetben az ember a *home* könyvtárában tartja a *GnuPG* és *SSH* kulcsait is, amelyek nélkül ez a művelet sajnos nem fog menni. Először tehát fogom azt a bizonyos *CD*-t, amit rögtön a kulcsok előállítása után megírtam, visszaállítom róla a kulcsokat, és máris működik minden. A *Duplicity* segítségével akár azt is megtehetjük, hogy a kiszolgálóról töröljük egy-egy fájl bizonyos időpontban készült mentését. Ezt kihasználva jómagam a következő parancsot fölvettem a *crontab*-omba, amely minden, két hónapnál idősebb mentést automatikusan töröl a szerverről:

```
$ duplicity --remove-older-
➥than 2M \
--encrypt-key AA43E426 --sign
➥-key AA43E426 \
scp://user@backup_serv/backup/
➥home \
/home/username
```

Ezzel egyik oldalról helyet spórolok, másrészt azonban az intézkedés értelemszerűen korlátozza azt az időt, amennyire a rendszer segítségével „visszanézhetek” és visszaszerezhetem az elveszett adataimat.

Összefoglalás

A cikkben bemutatott adatmentési megoldás nálam a leghatározottabban bevált. Rendelkezik minden olyan funkcióval, ami számomra fontos, és összességében kielégíti valamennyi igényemet. Ugyanakkor azt is látni kell, hogy még ez a rendszer sem tökéletes. A *Duplicity* például egyelőre nem támogatja a közvetlen láncokat (*hard link*), helyette ezeket is közönséges fájlokként kezeli. Ennek megfelelően ha olyan anyagot mentünk, majd állítunk vissza, amelyben ilyen láncok is előfordultak, akkor redundancia keletkezik, hiszen a visszaállításnál egyedi fájlok, és nem hivatkozások jönnek létre.
E fogyatékossága ellenére számomra még mindig a *Duplicity* a legmegfelelőbb megoldás. Ráadásul úgy tűnik, hogy a fejlesztők újabban belelendültek a munkába, tehát lehetséges, hogy az imént említett kellemetlenséget is hamarosan kiküszöbölik. Meg aztán az is lehet, hogy én magam fogom megoldani ezt a dolgot, és közkinccsé teszem. Akárhogy is lesz, az biztos, hogy ezzel a rendszerrel olcsón és viszonylag kevés munkával meg lehet valósítani egy tetszőleges, felügyelet nélkül működő titkosított hálózati adatmentő kiszolgálót.

Linux Journal 2006., 153. szám

Andrew J. De Ponte biztonsági szakértő aki szoftverfejlesztéssel is gyakran foglalkozik. 1997 óta számos UNIX-változattal dolgozott már, és úgy gondolja, hogy a siker kulcsa az, ha az ember megtalálja az egyensúlyt a termelékenység és a szép tervezés között. Az esetleges kérdéseket és megjegyzéseket a cyphactor@socall.rr.com címen várja.