Brochettes de Sécurité

Marcel bemutatja, hogyan állíthatunk vissza törölt állományokat, miként rejthetjük el titkos adatainkat, illetve hogyan készíthetünk róluk biztonsági másolatokat.

Á, François! Gyere, ezt nézd meg! Eszmefuttatásom tárgya a biztonság, és sikerült is összeállítanom a témáról egy étlapot. Quoi? Természetesen nem a hálózatokról van szó. Tudod, mon ami, egy étterem sikeres vezetéséhez mi a legfontosabb? Bien sur, egy gazdagon felszerelt borospince nagyon fontos, ebben egyetértünk, de most az ételre gondoltam. Tudod, a vendégeink sok mindenre éhesek és a mi kötelességünk, hogy olyan ízekkel kápráztassuk el folyamatosan az ínyüket, amilyenekre nem számítottak.

Qu'est-ce que tu dis? François, a biztonság a Linuxszal való főzőcskézés igen sokoldalú kérdése. A hálózati biztonság is egy ilyen kérdés, s nyilván mindenkinek ez jut először az eszébe. Ahhoz azonban, hogy az ízlésünk friss és éber maradjon, érdemes egy kicsit válogatni a hozzávalók között. Nézd meg például az első fogást! Nem tagadhatod le, hogy köze van a biztonsághoz, továbbá – François, miért nem figyelsz? Hogyan? Á, mais pardon, megérkeztek a vendégeink. Kérek mindenkit, foglaljon helyet! François azonnal hoz egy kis bort nekünk! Akad még a pincében egy 1998-as Clos de Vougeot, mely azt hiszem, ez mindenki ízlésének megfelel, és különleges légkört teremt. Egy finom Burgundy, mes amis. Vite, François, vite! François-val éppen a biztonság kérdésének különböző nézőpontokból történő megközelítéséről beszélgettünk. Amikor Linuxszal főzünk, a „biztonság” fogalmát emlegetve mindnyájan a hálózati biztonságra gondolunk, pedig a szó monsieur Roget híres fogalomtára szerint számos egyéb dolgot is jelenthet. Olyan szavakat lelhetünk fel benne, mint kötvény, letét, adóslevél, óvadék, kezes, hitelesítés, nyugta, hogy csak néhányat említsünk közülük. És mi a helyzet azzal a biztonsággal, amit rendszeresen végzett biztonsági mentésként ismerünk? Önök minden bizonnyal gyakran készítenek biztonsági másolatokat, ily módon épségben tudhatják a munkájukat, és ezt a fajta biztonságot a rendszeresen végzett adatmentés nyújtja. De még ebben az esetben is előfordulhat, hogy elvész egy állomány. Mi a helyzet akkor, ha az utolsó biztonsági mentés tegnap este vagy akár két órával ezelőtt történt? Jó szokásunk és a biztonságos munkához való egészséges vonzódásunk ellenére mégis elvesztettük az állományunkat. Mit tehetünk ebben a helyzetben? Már hallom a hétköznapi bölcsességet: ha egyszer letöröltünk egy állományt Linux-rendszerünkben, annak vége, örökre elveszett. Ez azonban nem teljesen van így, lehet még esélyünk a megmentésére.

Tesztrendszeremen egy kis meghajtó van csatlakoztatva a /mnt/tinydrive útvonalon. A `df /mnt/tinydrive` parancsra az alábbiak jelennek meg:

```
/dev/hdd6   42913   9063   31634   22%
/mnt/tinydrive
```

Mondtam, hogy kicsi, igaz? Létrehoztam néhány könyvtárat és állományt ezen a meghajtón. Az egyik állomány neve ***cabernet*** és a családom Cabernet-titkait tartalmazza Az állományt kilistázva ezt kapom eredményül:

```
$ ls -l
-rw-r--r--   1 marcel   wines   77 Jun 21 04:38
cabernet
```

77 bájt mint látjuk a titkok mindig elegánsak. Ha most az `rm cabernet` paranccsal az állományt, eltűnik, és nem tudjuk olvasni. Ennek ellenére egy kis kitartással és szerencsével visszaállíthatjuk, ahogy mindjárt be is mutatom. Fontos, hogy a kérdéses meghajtót azonnal leválasszuk, vagy kapcsoljuk ki a gépet, ha a gyökérkönyvtárról van szó:

```
unmount /mnt/tinydrive
```

Minél több idő telik el ugyanis az állomány törlésétől kezdve (következésképp minél több adat íródik a lemezre), annál kisebb az esélyünk arra, hogy az állományt maradéktalanul sikerül visszaállítani. A mentési kísérlethez a ***Debugfs*** nevű programot használom. Csak semmi fejetlenség, mes amis, minden rendszeren megtalálható ez a parancs az ***e2fsprogs*** csomag részeként:

```
# debugfs /dev/hdd6
debugfs 1.14, 9-Jan-1999 for EXT2 FS 0.5b,
95/08/09
debugfs:
```

Ha a `debugfs` parancssorába beírom, hogy `lsdel`, egy listát kapok, amely valahogy így néz ki:

```
7665    0 100644   5248    6/   6 Thu Apr 19
18:05:30 2001
  32 500 100600 12288   12/ 12 Thu Jun 21
04:38:39 2001
 158 500 100644     78    1/   1 Thu Jun 21
04:38:39 2001
 157 500 100644     77    1/   1 Thu Jun 21
04:40:25 2001
```

Az első oszlop a fájlleíró (inode). A második oszlopban az állomány tulajdonosának azonosítója található (UID). Mivel az én azonosítóm 500 és a véletlen törlés 21-én, szerdán következett be, úgy tűnik, hogy a keresett állomány a felsoroltak valamelyike. Továbbá arra is emlékszem, hogy a fájl 77 bájt hosszúságú volt. Mivel fájlnevek nincsenek a listán, minél több adat áll rendelkezésre az eredeti állományról, annál könnyebb a dolgunk. Még mindig a `debugfs` parancssoránál maradva, beírom a következő sort:

```
debugfs: dump <157> /home/marcel/cabs
```

A 157-es szám a fájlleíró, a /home/marcel/cabs pedig egy állomány egy másik, már csatlakoztatott fájlrendszerben. A `dump` parancsot használva a `debugfs`-ben a 157-es fájlleírójú állomány tartalmát átirányíthatom egy új, ***cabs*** nevű állományba. Nézzük a művelet eredményét:

```
$ cat /home/marcel/cabs
MiØrt is pr bÆlkoznÆnk j  Cabernet
eliÆll tÆsÆval, amikor meg is vÆsÆrolhatjuk
Henri's Fine Wines  zletØben?
```

Mais non! A titok felfedve!

Egy másik eszköz, ami hasznosnak bizonyulhat törölt állományok visszaállításánál, *Tom Pycke* ***Recover*** nevű programocskája. A ***Recover*** a törölt állományok akár hihetetlenül

nagyra duzzadó listájával való munkát könnyíti meg, sőt, szinte fájdalommentessé teszi. Töltsd le a legfrissebb forrásállományt, a (➲ http://recover.sourceforge.net/linux/recover) csomagold ki és fordítsd le:

```
tar -xzvf recover-1.3.tar.gz
cd recover-1.3
make
make install
```

A program a `recover` parancs begépelésével indul, ezután a rendszeren elérhető különböző fájlrendszereket keresi meg, illetve a megfelelő fájlrendszer parancssorban történő megadásával is indítható. Ezt egy kérdés-felelet rész követi, amellyel a törölt állományok hatalmas listáját szűkíthetjük kezelhető méretűre. Egy példa:

```
# recover /dev/hdd6
Recover v1.3 by Tom Pycke
<Tom.Pycke@advalvas.be>

Getting inodes (this can take some time)...
debugfs 1.14, 9-Jan-1999 for EXT2 FS 0.5b,
95/08/09

In what year did you delete the file? (eg.
1999): 2001
```

Ezután a program megkérdezi a hónapot, amelyet a keltezés- és időtartam, valamint az állomány lehetséges méretének alsó és felső határa követ. A következő kimenetet kaptam:

```
=> 158 78 JUN THU 21 4:38:39 2001
=> 157 77 JUN THU 21 4:40:25 2001
2 inodes found. Where shall i dump them?
(directory):  /tmp
```

A kapott állományok a `dump157` és `dump158` névre hallgatnak. Ezután nincs más teendőnk, mind a `cat`-tel megvizsgálni a tartalmukat. Gondolva rá, hogy esetleg futtatható állományokról van szó, érdemes lehet a `file` paranccsal ezt előtte megvizsgálni. Mint jól tudjuk (és ahogyan monsieur Roget is rájött), a biztonságnak más nézőpontja is létezik. Megnyugtat a tudat, hogy bármikor vissza tudunk állítani egy véletlenül törölt állományt az `ext2fs` fájlrendszerünkben, de mi a helyzet az ellenkező oldalról nézve a dolgot, ha például az állomány tartalma szigorúan titkos, és nem akarjuk, hogy az életben még egyszer valaki belepillanthasson? Ha az állományt inkább valami nagyon sötét, a szőlőművelésre teljesen alkalmatlan vidékre szeretném száműzni? Amennyiben rendszereden megtalálható a ***fileutils*** csomag legújabb kiadása (olyan programokkal, mint a `cp`, `mv` stb.), máris kéznél van a megoldás. Ebben található a `shred` nevű program, amely olvashatatlanná szabdalja az állományt:

```
$ shred secret_instructions
```

Ebben a formában kiadva a parancsot a `secret_instructions` állomány tartalma valóban titokban marad, maga az állomány azonban nem törlődik, csak összekeveredik, mint a jól felvert tojáshab. Igazi biztonságérzet azonban inkább a parancs alábbi formájával érhető el:

```
$ shred -z -u secret_instructions
```

Ennek hatására a program nullákkal írja felül az állományt (a `-z` kapcsoló hatása), majd törli (`-u`). Titkainkat ezennel biztonságba helyeztük.

Konyhai felfedező utunkat ma néhány könnyű, a biztonsági mentéssel kapcsolatos javaslattal kezdtük. Tételezzük fel, hogy adatainkat rendszeres időközönként szalagra mentjük amelyeket azután más, biztonságos helyre szállítunk. Így azokat a titkos állományokat is tároljuk, amelyeket oly elszántan igyekszünk a rendszerünkről eltávolítani. Tegyük fel, hogy a `tar` paranccsal elmentettem az /etc/profile állományt:

```
tar -cvf /tmp/profile.tar /etc/profile
```

Egy ily módon előálló állomány meglehetősen nyílt:

```
$ more /tmp/profile.tar
etc/profile0100644000000000000000000000001163072
353262310117170
ustar  rootroot
# /etc/profile
```

Amint látható, a tárállományban lévő fájlok tartalma könnyedén olvasható. Természetesen tudom, hogy az állományokat egy pofonegyszerű

```
tar -xvf filename.tar
```

paranccsal egyszerűen ki is csomagolhatnám, de most mást szeretnék megmutatni. Mi lenne, ha kódolnánk az tárállományt? Éppen ez a gondolat hívta életre *Brian Wagener* és *Katrina Illari* ***Sectar*** nevű programját, amely a GNU `tar` biztonsági kiterjesztése. Könnyen hozzájuthatunk a ➲ sourceforge.net/projects/star címről.

Két csomagra lesz szükségünk: az első a GNU `tar` javított változata, a második maga a `sectar`. A csomag kibontásával és fordításával kell kezdenünk:

```
tar -xzvf tar_sectar.tar.gz
cd tar-1.13
make
make install
```

Ez majdnem megegyezik az eredeti `tar`-csomaggal, amely a rendszerünkön található, kivéve egy nagyon fontos beállítási lehetőséget. Ha kiadjuk a

```
/usr/local/bin/tar -help
```

parancsot, az alábbi sort is láthatjuk:

```
-e, --encrypt    filter the archive through
                 encryptor
```

E tulajdonság használatához azonban a `sectar` csomag lefordítása is szükséges. A legújabb forrást használva az alábbi lépéseket követtem:

```
tar -xzvf sectar-1.02.tar.gz
cd sectar-1.02
make
make install
```

Ezt a most felfedezett titkosítási lehetőséget kihasználva újra előállítom szigorúan titkos tárállományomat:

```
$ tar -ecvf /tmp/profile.tar /etc/profile
The 256 bit key created is keyfile.993515072
```

Most ezt a kulcsállományt (amely a munkakönyvtáramban található) lemezre vagy egyéb biztonságos helyre másolom addig az időpontig, amikor majd vissza kell állítanom az állományt. Ha a kulcsállomány rendelkezésre áll, az adatok `tar`-állományból való kicsomagolása gyerekjáték, hiányában viszont az adatok titkosak maradnak. Ha a kulcsállomány nélkül nézzük át az állományt, zavarosnak, olvashatatlannak és nyomtathatatlannak találjuk.

Örömömmel láttunk vendégül benneteket, a következő alkalommal se felejtsetek el csatlakozni hozzánk! A votre santé! Bon apétit!

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. rendszerépítéssel és hálózati tanácsadással foglalkozó cég elnöke. A Világhálón elérhető honlapján sok hasznos dolog ➲ http://www.salmar.com/marcel/ lelhető.

Roget fogalomtára ➲ http://www.thesaurus.com