A Linux fájlrendszer biztonsága (2. rész)

A múlt hónapban megismerkedtünk a jogosultságok alapjaival. Most itt az ideje, hogy megvizsgáljunk néhány hasznos bitet a felhasználók kényelmes és biztonságos együttműködésének megteremtéséhez.

Múlt alkalommal az alapoktól indulva vizsgáltuk meg a fájl- és könyvtárjogosultságok rendszerét: megismertük a felhasználók és csoportok fogalmát valamint a fájlokra és könyvtárokra vonatkozó olvasási, írási és futtatási jogok beállításának és törlésének módját. Ebben a részben a jogosultságok néhány fejlettebb formájára irányítjuk figyelmünket, felderítjük a jogosultságok numerikus módszerét, az `umask` parancsot és a root jogosultságainak átadását a `su` és `sudo` parancsokkal. A mostani cikk a múlt havinál több középszintű információt tartalmaz, de remélhetőleg még akkor is érthető lesz, ha csak annyit tudunk a témáról, amit az előző alkalommal olvastunk.

A sticky bit

Idézzük fel a múlt hónapban tanulmányozott `extreme_casseroles` könyvtár listájának eredményét:

```
drwxr-x--- 8  biff drummers 288 Mar 25 01:38
➥extreme_casseroles
```

Biztosan emlékszünk még arra, hogy a könyvtárra vonatkozó csoport-jogosultságokat r-x értékre állítottuk be, vagyis a csoport által olvasható és futtatható értékre, úgyhogy a `drummers` csoport tagjai beléphetnek ebbe a könyvtárba és olvashatják a benne található recepteket. Tegyük fel, hogy dobos barátunk, `Biff`, szeretné, ha zenésztársai nem csak olvasni tudnák ezeket a recepteket, hanem a sajátjaikat is elhelyezhetnék itt. Ahogy az elmúlt alkalommal láthattuk, ehhez nem kell mást tennie, mint beállítani a könyvtár csoportra vonatkozó írási bitjét a következőképpen:

```
chmod g+w ./extreme_casseroles
```

Ezzel csak egy gond van: az írási jog magában foglalja az adott könyvtárban új fájl létrehozásának, illetve egy már létező fájl törlésének a jogosultságát is. Mi akadályozza meg az egyik dobos cimborát abban, hogy mások receptjeit letörölje? A ***sticky bit***, semmi más.
Régebben a ***sticky*** bitet arra használták, hogy a fájlt (programot) a memóriába olvassanak be, így a hívása esetén sokkal gyorsabban tudott betöltődni. A ***Linuxon*** azonban más a szerepe. Amikor egy könyvtár esetében beállítjuk a ***sticky*** bitet, akkor ezzel az adott könyvtárban korlátozzuk a felhasználók törlési lehetőségeit. Ez konkrétan azt jelenti, hogy a könyvtárban egy adott fájl letörléséhez tulajdonosi jogokkal kell rendelkeznünk vagy a fájl vagy a könyvtár esetében, még akkor is, ha a tulajdonos csoporthoz tartozunk, és a csoport írási lehetősége be van állítva.
A ***sticky*** bit beállításához az alábbi parancsot kell futtatnunk:

```
chmod +t könyvtárnév
```

A példánkban a parancs formája `chmod +t extreme_casseroles` lenne. Ha most az `ls` parancsot a `-d` kapcsolóval használva lekérdezzük magának a könyvtárnak a részletes jellemzőit, hogy a könyvtárhoz kötődő jogosultságok jelenjenek meg és nem a könyvtár tartalma, vagyis kiadjuk a `ls -ld extreme_casseroles` parancsot, az alábbi eredményt kapjuk:

```
drwxrwx--T 8  biff drummers  288  Mar 25 01:38
➥extreme_casseroles
```

Figyeljük meg a jogosultságokat jelző karakterlánc végi `T` betűt. Normál esetben itt egy `x` vagy `-` jelet várnánk attól függően, hogy a könyvtáron az egyéb felhasználók rendelkeznek-e írási joggal. A `T` azt jelöli, hogy a könyvtáron az egyéb felhasználóknak nincs futtatási joga és a ***sticky*** bit be van kapcsolva. A korlátozás hatásának bemutatásához tegyük fel, hogy az `extreme_casseroles` könyvtár tartalmának a kiíratása az *1. listán* látható eredményt adja.
Tegyük fel továbbá, hogy a `crash` nevű felhasználónak nem tetszik a `pineapple_mushroom_surprise.txt` fájl és ezért megpróbálja letörölni. Arra számít, hogy ez sikerülni is fog neki, hiszen a `drummers` csoporthoz tartozik, és a csoport rendelkezik ezen a fájlon írási joggal. Ne feledjük azonban, hogy biff bekapcsolta a szülőkönyvtár ***sticky*** bitjét. Ez az oka annak, hogy `crash` kísérlete kudarcba fullad, ahogy azt a *2. listán* láthatjuk is. Még egy megjegyzés a ***sticky*** bittel kapcsolatban: a beállításnak csak az adott könyvtárszinten érvényesül a hatása. Talán észrevettük, hogy az 1. lista az `extreme_casseroles` könyvtárban lévő két élmelyítő recept mellett egy `src` nevű könyvtárat is tartalmaz. Az `src` könyvtár tartalmára nem lesz hatással az `extreme_casserole` sticky bitjének beállítása, habár magára a könyvtárra igen. Ha `biff` az `src` könyvtár tartalmát is meg szeretné óvni a csoport tagjai által végrehajtott törléstől, az `src` könyvtár saját ***sticky*** bitjét is be kell állítania.

1. lista Az extreme_casseroles könyvtár tartalma

```
drwxrwxr-T   3 biff drummers   192 2004-08-10 23:39 .
drwxr-xr-x   3 biff drummers  4008 2004-08-10 23:39 ..
-rw-rw-r--   1 biff drummers    18 2004-07-08 07:40 chocolate_turkey_casserole.txt
-rw-rw-r--   1 biff drummers    12 2004-08-08 15:10 pineapple_mushroom_surprise.txt
drwxr-xr-x   2 biff drummers    80 2004-08-10 23:28 src
```

2. lista Törlési kísérlet a sticky bit bekapcsolt állapotában

```
crash> rm pineapple_mushroom_surprise.txt
rm: cannot remove
➥`pineapple_mushroom_surprise.txt´:
Operation not permitted
```

A setuid és setgid beállításai

Elérkeztünk a ***UNIX*** világának legveszélyesebb jogosultságbitjeihez, a ***setuid*** és ***setgid*** bitekhez. Egy futtatható bináris fájl esetén a ***setuid*** bit beállítása azt eredményezi, hogy a program a futtató felhasználótól függetlenül úgy fut, mintha a tulajdonosa futtatná. Hasonlóan, a ***setgid*** bitet beállítva egy futtatható fájl esetén az úgy fog futni, mintha a tulajdonos csoport tagja futtatná, függetlenül futtató személytől. Amikor azt a kifejezést használom, hogy úgy fut, azon azt értem, hogy ugyanazokkal a jogosultságokkal fut a program. Tegyük fel, hogy `biff` megír és lefordít egy olyan `killpms` nevű *C* programot, amelynek a hatása megegyezik az

```
rm /extreme_casseroles/
➥pineapple_mushroom_surprise.txt
```

parancséval, továbbá a

```
chmod +s ./killpms
```

paranccsal bekapcsolja a fájl ***setuid*** bitjét és beállítja a csoport-futtathatóságát is. A ***killpms*** fájl részletes jellemzőit kiíratva az alábbi sort láthatnánk:

```
-rwsr-xr--   1 biff drummers    22 2004-08-11 23:01
➥killpms
```

Ha `crash` lefuttatja ezt a programot, végül is sikerül elérnie, hogy törölje a ***Pineapple-Mushroom Surprise*** receptet – a `killpms` ugyanis olyan jogosultságokkal fut, mintha `biff` futtatta volna. Amikor a `killpms` megpróbálja letörölni a `pineapple_mushroom_surprise.txt` fájlt, sikerrel is jár, hiszen a fájl a tulajdonosa számára írható, a killpms pedig olyan jogosultságokkal rendelkezik, mint a tulajdonosa, `biff`. Ha egy programot a ***setuid*** bit beállításával akarunk futtatni, annak futtathatónak kell lennie a csoport és egyéb felhasználók számára – remélem nem kell magyaráznom, hogy miért. További jellemző, hogy a ***Linux*** rendszermagja a héjprogramok esetében nem veszi figyelembe a ***setuid*** és ***setgid*** biteket, ezek csak bináris (lefordított) futtatható fájlok esetében működnek.

A ***setgid*** ugyanúgy működik, csak a csoport-jogosultságokat használja. Ha egy futtatható fájl esetén a `chmod g+s filenév` paranccsal beállítjuk a ***setgid*** bitet és a fájl futtatható az egyéb felhasználók számára (`-r-xr-sr-x`), akkor a fájl futtatásakor a csoportazonosítót (***group ID***) használja a futtató felhasználóé helyett.

A fenti példa esetén, ha a `killpms` egyéb felhasználói jogosultságait r-x beállításra változtatjuk meg (`chmod o+x killpms`), és ugyanakkor beállítjuk a ***setgid*** bitet is (`chmod g+s killpms`), akkor függetlenül attól, hogy ki futtatja a `killpms`-t az mindenképpen a `drummers` csoport jogosultságaival fog rendelkezni, hiszen a `drummers` a fájl csoport-tulajdonosa.

A setgid és a könyvtárak

De mi a helyzet a könyvtárakkal? Nos a ***setuid*** nincs hatással a könyvtárakra, ellenben a ***setgid*** igen, ami nem teljesen magától értetődő tény. Rendes körülmények közt, ha létrehozunk egy fájlt, annak tulajdonosaként önműködően a saját felhasználói és (elsődleges) csoport-azonosítónk kerül beállításra. Például ha biff létrehoz egy fájlt, annak tulajdonosa `biff` lesz, a csoport-tulajdonosa pedig a `drummers` csoport, amennyiben a `drummers` `biff` elsődleges csoportja az */etc/passwd* lista alapján.

Beállítva azonban egy könyvtár ***setgid*** bitjét, az adott könyvtárban létrehozott fájlok a könyvtár csoport-tulajdonosát fogják örökölni. Ez akkor hasznos, ha a rendszerünk felhasználói jellemzően másodlagos csoportoknak is tagjai és rendszeresen hoznak létre olyan fájlokat, amelyeknek e csoportok többi tagja számára is elérhetőknek kell lenniük. Például ha az `animal` nevű felhasználó az */etc/group* fájlban úgy szerepel, mint a `drummers` csoport másodlagos tagja, de az `/etc/passwd` listában az elsődleges csoportja a `muppets`, akkor `animal` minden gond nélkül létrehozhat az `extreme_casseroles/` könyvtárban fájlokat `drwxrwx--T` jogosultság-beállítással. Mivel alapértelmezésben `animal` fájljai a `muppets` csoporthoz tartoznak, nem pedig a `drummers` csoporthoz, a `drummers` csoport többi tagjának sem olvasási, sem írási joga nem lesz `animal` receptjein, amíg `animal` kézzel át nem állítja a fájljai csoport-tulajdonosát (`chgrp drummers újfájl`) vagy meg nem változtatja az egyéb felhasználókra vonatkozó jogosultságokat (`chmod o+rw újfájl`).

Másrészről viszont ha `biff` vagy a rendszergazda beállítja az `extreme_casseroles/` könyvtár ***setgid*** bitjét (`chmod g+s extreme_casseroles`), akkor `animal` itt létrehozott új fájljainak csoport-tulajdonosa a `drummers` lesz a könyvtár tulajdonos csoportjának megfelelően. Minden egyéb jogosultság változatlan marad. Ha a kérdéses könyvtár már eleve nem írható a csoport által, akkor a `setgid` bitnek semmilyen hatása nincs, hiszen a csoporttagok nem hozhatnak benne létre fájlokat.

Ezzel megismertük az összes jogosultsági lehetőséget: az olvasási, írási, futtatási jogokat, a ***sticky***-bitet, a ***setuid*** és

3. lista Az alapértelmezett jogosultságmaszkunk ellenőrzése

```
mick@localhost:/home/mick> umask
0022
```

4. lista Egy 0022-es maszkkal létrehozott fájl és könyvtár

```
-rwxr-xr-x 2 mick users 48 2004-08-13 08:31 default_dir
-rw-r--r-- 1 mick users  4 2004-08-13 08:31 default_file
```

setgid biteket. Ha mind a hatot megértettük, valószínűleg a ***Linux*** felhasználók kisebbik feléhez tartozunk. De ezzel még nincs vége!

Numerikus módszerek

Eddig kódokat használtunk a jogosultságok jelölésére: r jelentette az olvasási jogot, w az írásit és így tovább. Mondanom sem kell, hogy a minden mással együtt a rendszerünk a jogosultságokat is számokként tárolja. A `chmod` felismeri mind a kódokkal (`u+rwx,go-w`), mind pedig a számokkal megadott jogosultság-módosítókat.

Egy számkód négy számjegyből áll, amelyek balról jobbra haladva a különleges jogokat, tulajdonosi jogokat, csoportjogokat és az egyéb jogokat jelentik. Emlékezzünk rá, hogy az egyéb jogok azokra a felhasználókra vonatkoznak, akik nem tartoznak sem a tulajdonos, sem a csoporttagok kategóriájába. Például a 0700 számkód szerint nincs különleges jogosultság, minden tulajdonosi jog be van kapcsolva, és semmilyen csoport- vagy egyéb jogosultság sincs.

Minden jogosultság-típushoz egy meghatározott számérték tartozik, az egyes jogosultság-típusokhoz tartozó értékek pedig az egyes helyiértékeken összeadódnak: a számjegy az összes beállítani kíván bit összegét jelenti. Ha például a tulajdonosi jogosultságok értéke 7, ez a 4-nek (az olvasási jog értéke), a 2-nek (az írási jog értéke) és az 1-nek (a futtatási jog értéke) összegeként áll elő.

Ahogy az előbb említettem, az egyes jogokhoz tartozó értékek: 4–olvasás, 2–írás és 1–futtatás. (Én úgy jegyeztem meg ezeket, hogy ismételgettem magamban az „olvasás-írás-futtatás, 4-2-1” mondókát.) Miért nem 3, kérdezhetné valaki? Azért, mert így biztosítható, hogy ne legyen két olyan jogosultság-kombináció, amihez ugyanaz az érték tartozik.

A különleges jogosultságok a következők: 4 jelenti a ***setuid*** bitet, 2 a ***setgid*** bitet, az 1 pedig a ***sticky*** bitet. Például a 3000 a ***setgid*** és ***sticky*** bit bekapcsolását jelenti, miközben semmilyen más jogosultság nincs – ennek a beállításnak nincs persze sok értelme.

Nézzünk még egy példát a numerikus megadásra. Ha futtatom a `chmod 0644 mycoolfile` parancsot, az *1. ábrán* látható jogosultságokat állítom be a `mycoolfile` fájl számára.

A numerikus módszer teljesebb leírását a `coreutils` parancs ***Numeric Modes*** részében találhatjuk meg, vagyis ha kiadjuk az `info coreutils numeric` parancsot.

Az umask parancs

Mielőtt más témába kezdenék, szeretném bemutatni az utolsó jogosultsággal kapcsolatos parancsot. Az `umask` a ***Bash*** héjba beépített parancs, amely kiírja vagy beállítja az alapértelmezett jogosultságmaszkunkat. A sajátunkat az umask parancs paraméterek nélküli beírásával nézhetjük meg, ez egy négyjegyű számot fog eredményül adni.

A rendszeremen a 2. listának megfelelő választ kaptam.

A 0022 jelentése a következő: nincs különleges jogosultság, nincs tulajdonosi jogosultság, a csoport- és egyéb jogosultságok közül pedig az írási van bekapcsolva. Ez meg hogy lehetséges?

A válasz, hogy valójában az `umask` nem a kóddal, hanem maszkokkal dolgozik. A 0022 az a szám, amit a 0777 értékből kivonva dönti el a rendszer, hogy a létrehozott fájlokhoz milyen jogosultságokat rendel: `0777-0022 = 0755`.

Aha! Szóval a létrehozott fájljaim tulajdonosa rendelkezni fog olvasási, írási és futtatási jogokkal (7 = 4+2+1), a csoport- és egyéb jogosultságok közül pedig az olvasás és futtatás lesz beállítva (5 = 4+1). Igaz ez? Majdnem. Valójában az `umask` a futtatási jogosultságot önműködően csak a könyvtárakra állítja be. Még ha a jogosultság-maszkunk szerint szerepel is a futtatási jog, a létrehozott közönséges fájlok futtatás-bitje nem kerül önműködően bekapcsolásra.

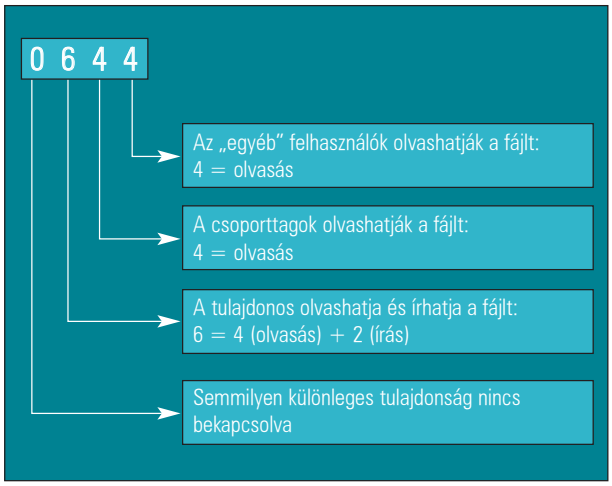
Tehát a 0022 jogosultságmaszkommal, amely az alapértelmezett jogokat 0755 értékre állítja be, a `default_dir` könyvtárban létrehozva egy `default_file` nevű fájlt, a két tételnek a részletes adatai a *4. listán* lévő képet fogják mutatni.

Az alapértelmezett jogosultságmaszkunk megváltoztatásához egyszerűen futtassuk az `umask` parancsot, paraméterként megadva a kívánt maszkot. Például ha azt szeretném, hogy az összes fájlom rendelkezzen csoport-olvasási joggal, de semmilyen más jogot nem szeretnék beállítani, ez a 0740 numerikus kóddal fejezhető ki. Ezt kivonva a 0777 kódból a 0037 maszkot kapom, vagyis az alapértelmezett maszkom beállítására az `umask 0037` parancsot kell használnom. Ez az új maszk azonban csak az aktuális munkafolyamatra és az ebből indított héjakra érvényes, ha álladósítani szeretném a beállítást, a ***.bashrc*** fájlomba új sorként be kellene írnom az `umask 0037` parancsot.

A su és sudo parancsok

Most, hogy már mindent tudunk a jogosultságokról, numerikus kódokról és a maszkokról, szólnom kell néhány szót arról, hogy mi a helyzet a felhasználókkal, csoportokkal és ezek jogaival a gyakorlatban. A ***UNIX*** biztonsági rendszerével nagyon gyakran az a gond, hogy egy adott rendszeren a hatókörök nagyjából úgy néznek ki, hogy a root mindent megtehet, a felhasználók meg szinte semmit.

Sajnos sokkal könnyebb kiadni egy gyors `su` parancsot, és ezzel egy időre megszerezni a root-jogosultságokat, mint létrehozni egy jól meghatározott jogosultságokkal rendelkező csoportrendszert, amelyben a rendszergazdák és az alrendszerek felügyelői pontosan azokkal a jogosultságokkal rendelkeznek, amelyekre szükségük van. Használhatjuk persze a `su` parancsot a `-c` kapcsolóval is, ami csak egy bizonyos parancs root-ként való futtatását teszi lehetővé az egész héj helyett (például `su -c rm fájlnév.txt`), de ehhez szükség van a root-jelszó beírására. Az pedig soha nem jó,



1. ábra A mycoolfile jogosultságainak magyarázata

FONTOS FIGYELMEZTETÉS!

A ***setuid*** és ***setgid*** bitek nagyon veszélyesek lehetnek, ha root vagy valamilyen kiemelt jogosultságokkal rendelkező felhasználó vagy csoport tulajdonában lévő fájlokon kerül beállításra. Azért mutatom be a ***setuid*** és ***setgid*** működését, hogy megértsük, mire való – nem gondolom, hogy a gyakorlatban valami fontos felhasználási célja lenne számotokra. A **sudo** parancs, amit a cikk további részében ismertetek majd, sokkal alkalmasabb eszköz arra, hogy valakit root-jogosultságokkal ruházzunk fel.

ha néhány személyen kívül más is ismeri a root-jelszót. A root-mindent-visz probléma megoldásának egy másik megközelítési módja a ***SELinux***-hoz hasonló szerep-alapú hozzáférési rendszer (***RBAC***) használata, amely olyan hozzáférési rendszert léptet hatályba, ami csökkenti a root hatáskörét. Ez azonban talán még bonyolultabbá teszi a helyzetet, mint a megfelelő csoportok és csoport-jogosultságok beállítása, amivel nem akarom azt mondani, hogy a ***SELinux*** és társai ne lennének nagyon hasznosak – én nagyon szeretem az ***RBAC***-t.

Egy jó középutat jelenthet a `sudo` parancs használata. A `sudo` lehetővé teszi a felhasználók számára, hogy egyes parancsokat root jogosultsággal futtassanak anélkül, hogy ismerniük kellene a root-jelszót. A `sudo` mára a legtöbb Linux rendszercsomag alapelemét képezi.

A `sudo` beállításait az */etc/**sudoers*** fájl tartalmazza, de ennek közvetlen szerkesztésére nincs szükség. Ehelyett kiadhatjuk a `visudo` parancsot, amely megnyitja a fájl szerkesztőjét, ami alapesetben a `vi`. Az `EDITOR` környezeti változó megváltoztatásával más karakteres szerkesztőprogramot is beállíthatunk. Például a */**usr**/**bin**/**gedit*** használatához az alábbi parancsot kell kiadnunk:

```
export EDITOR=/usr/bin/gedit
```

Hely hiányában nem tudom részletezni a ***sudoers*** fájl szerkezetét, a ***sudoers(5)***, ***sudo(8)*** és ***visudo(8)*** súgóoldalakon olvashatjuk el a teljes leírást. Nézzünk inkább egy gyors példát.

Emlékszünk még a `crash` felhasználó próbálkozására, hogy megszabaduljon a ***Pineapple-Mushroom Surprise*** recepttől? Bár ebben az esetben nem kellene ilyen durva eszközökhöz nyúlnunk – a bemutatott jogosultság-technikák megfelelő eszközt biztosítanak ehhez –, használhatjuk a `sudo` parancsot arra, hogy `crash` elérje célját, feltéve, hogy `biff` rendelkezik root-jogosultsággal. Először is váljunk root felhasználóvá (`su -`). Ezután futtassuk a `visudo` parancsot. Ennek hatására a `vi` szerkesztőprogramba kerülünk, amelyben az */**etc**/**sudoers*** fájl van épp megnyitva (ha a `vi` használatában újoncok vagyunk, a ***vi(1)*** súgóoldalt érdemes elolvasnunk). Menjünk el a fájl végéig, majd adjuk az alábbi sort a fájlhoz:

```
crash localhost=/bin/rm /home/biff/
➥extreme_casseroles/pineapple_mushroom_surprise.txt
```

Mentsük a fájlt, majd lépjünk ki. Most `crash` a feladat elvégzéséhez beírja a következő parancsot:

```
sudo rm /home/biff/extreme_casseroles/
➥pineapple_mushroom_surprise.txt
```

A parancs a felhasználó jelszavának beírását kéri. Miután hibátlanul beírta, az alábbi parancs fut le, mégpedig root felhasználóként:

```
/bin/rm /home/biff/extreme_casseroles/
➥pineapple_mushroom_surprise.txt
```

Ezzel megtörtént a kérdéses fájl törlése. Egy másik megoldás, ha az */**etc**/**sudoers*** fájlban lévő sor a következőképpen fest:

```
crash localhost=/bin/rm /home/biff/
➥extreme_casseroles/*
```

Ebben az esetben `crash` bármit letörölhet az `extreme_casseroles` könyvtárban függetlenül a ***sticky*** bit beállításától.

A `sudo` nagyon kényelmes eszköz, de legalább ilyen veszélyes is lehet, bánjunk vele körültekintően – a root jogosultságaival nem szabad játszadozni. Tényleg okosabb, ha a felhasználói és csoportjogosultságokat állítjuk be a célnak megfelelően, mint ha kiadjuk a root jogosultságait, még ha csak a `sudo` parancson keresztül is. Még az is jobb, ha egy olyan ***RBAC***-alapú rendszert használunk, mint a ***SELinux***, amennyiben rendszer védelme ezt megkívánja. Ez alkalommal ennyi fért bele. Remélem hasznos volt ez az ismertető. Legyetek óvatosak a következő alkalomig!

Linux Journal 2004. november, 127. szám

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél.