# Hálózati szolgáltatások futtatása Felhasználói Módú Linuxon (1. rész)

Hálózati háttérprogramok elszigetelése a Linux kernel virtualizációs képességeinek kiaknázásával.

2006. májusában a ***Linux Journal Paranoid Pingvin*** rovatában már kifejtettem a ***Debian 3.1*** kitűnő képességeit virtualizációs környezetek támogatására, ide értve a ***Felhasználói Módú Linuxot*** (***User Mode Linux; UML***) is. Ugyanebben a számban ***Matthew Hoskins „Felhasználói Módú Linux”*** című cikkében olvashattunk egy gyors és boszorkányos receptet a ***Felhasználói Módú Linux*** kipróbálására, előre lefordított ***UML*** kernel és előre elkészített gyökér-fájlrendszer képmásfájlok segítségével. (***UML: User-Mode Linux, Felhasználói Módú Linux, FML***)

Ha ezek a cikkek felkeltették étvágyukat egy átfogóbb és biztonságosabbra tervezett ***FML*** rendszerre, akkor lássanak hozzá: az elkövetkezőkben belecsobbanunk a ***Felhasználói Módú Linux*** átélésébe, gondosan megvizsgálva minden mozzanatot (beleértve a kiadandó parancsokat is), hogy ki-ki elkészíthesse legszemélyesebb Felhasználói Módú Linux változatát, amivel például hálózati szolgáltatások végezhetőek.

## Célok

Miért is tesszük mindezt, minek az elérésében reménykedünk? Ahogy azt egy korábbi írásomban említettem, a virtualizáció hasonlít a ***chroot jail*** elvéhez („elkülönített börtön”, melyben a látszólagos gyökérkönyvtár eltér a fájlrendszer tényleges gyökérkönyvtárától). Ez bezár egy processzt vagy háttérprogramot (démont) az operációs rendszerkörnyezet egy részébe, ezáltal igencsak megnehezítve a támadó dolgát, aki hatalma alá szeretné vonni a teljes operációs rendszert. Ezt akkor sem fogja tudni megtenni, ha sikerül rést találnia a processz vagy háttérfolyamat működésében.

A ***chroot*** leszűkíti tehát a folyamat hatáskörét a gazdagép (valódi) fájlrendszerének egy részére.

A virtualizáció ehhez hasonló, de ott a leszűkítés egy teljesen virtuális számítógépre történik, amely a (valódi) gazdagépen fut. Ez azt jelenti, hogy a merevlemez, a memória és a kernel, sőt még az olyan rendszereszközök, mint a hálózati és hangkártyák is csak virtuálisan léteznek a számára. A ***Felhasználói Módú Linuxban*** ez úgy valósul meg, hogy a vendég (virtuális) rendszermag a gazdagépen levő (valódi) rendszermag felhasználói terében lévő processzként fut.

Mivel mind a vendég, mind a gazda rendszermag ***Linux*** kernel, a ***Felhasználói Módú Linuxszal*** történő virtualizáció gyors és hatékony.

A vendég kernelnek nem kell adminisztrátori jogosultságokkal futnia a gazdagépen, ezért a vendég rendszeren valamely háttérfolyamatot megtörő és ebből (a vendég rendszeren) adminisztrátori jogokat megszerző támadó is csak alacsony jogosultságú héj elérést tud szerezni a gazdagépen.

Ez nem jelenti azt, hogy lehetetlen adminisztrátori jogosultságot szerezni a gazdagépen. Ha a támadónak valóban sikerül odáig eljutnia, hogy héj elérése legyen a gazdagépen, ezt erősebb jogosultságúvá alakíthatja, ha talál valami alkalmas sebezhetőséget a gazdagép rendszermagjában vagy valamely, ott futó felhasználói programban. (Emlékeztetőül: semmilyen helyi sebezhetőség nem pusztán ***helyi***, ha már hálózatba van kötve a rendszer!)

De ezzel eléggé megnehezítettük a támadó dolgát efféle sebezhetőségek kiaknázására, különösen akkor, ha ez a rés nincs ott a vendég (virtuális) rendszerben.

Most már sejthető, hogy hogyan érdemes terveznünk: a vendég rendszernek oly csupasznak kell lennie, amennyire csak lehetséges. Semmilyen felesleges programot nem kell rá telepítenünk, semmi felesleges erőforrást nem kell rendelkezésére bocsátanunk, hogy ezáltal a minimálisra csökkentsük a támadásra alkalmas felületet. Ha például ***DNS*** szerverként működik a gépünk, akkor csak az alapvető hálózati támogatást, ***BIND***-ot (vagy más ***DNS***-kiszolgáló csomagot) kell feltennünk rá, és lehetőleg semmi mást. Sem ***X Window*** rendszert, sem Apache webszervert – semmi többet, mint amire a szoros értelemben vett ***DNS***-kiszolgálónak szüksége van.

Ha igazán paranoiások vagyunk, akkor még az ***SSH*** kiszolgálót is kihagyhatjuk, ehelyett egy virtuális soros konzolról intézve az adminisztrációt. Bár, őszintén szólva, egy olyan megoldás, hogy csak egy adott ***IP***-címről: a gazdagépéről fogadja el a szerver az ***SSH*** kapcsolatot, talán egy ésszerű középutat jelenthet. Az is egy járható út, hogy ***SELinux*** alól futtatjuk a ***Felhasználói Módú Linuxot***; azonban ennek tárgyalása már túlmutatna jelen sorozatunk keretein.

Tekintsünk egy olyan helyzetet, hogy egy „bástya” szervernek több hálózati szolgáltatást is futtatnia kell: ***Apache***-ot, ***BIND***-ot. Ekkor érdemes két különböző vendég rendszert futtatni ugyanazon a gazdagépen. Az egyiken csak az Apache (és függőségei) fussanak, a másikon hasonlóképpen: csak a ***BIND*** legyen telepítve. Ily módon egy ***BIND*** biztonsági rés nem vezet közvetlenül weboldalunk elcsúfításához, vagy egy ügyetlenül elkészített webes alkalmazás megtörése nem nyit utat a ***BIND*** meghamisításához.
***Összefoglalásként***: két fő tervezési elvünk az lehet, hogy minden hálózati szolgáltatásnak külön virtuális gépet érdemes készíteni, valamint, hogy ezek mindegyike legyen olyan puritán és biztonságos, amennyire csak lehetséges. Remélhetőleg ez olyan szakaszolt, bástyaerős szerverfelépítéshez vezet, amely a lehető legtöbb absztrakciós réteget kihúzza a támadó és a teljes adminisztrátori birtokba vétel közé.
Jelen cikksorozat hátralevő részeiben egy olyan példát fogok bemutatni, amelyben egyetlen vendég rendszer működik: ez ***BIND***-ot futtat. Mind a gazdagép, mind a vendég rendszer ***Debian 3.1***-esen alapul, mivel a ***Debian*** igen népszerű az ***FML*** vendég rendszerek terén (sokszor nevét is adja más, lecsupaszított telepítéshez is, akárcsak a ***Slackware***). Ezzel együtt a továbbiak alkalmazhatóak egyéb disztribúciókra is, mind gazdagép, mind a vendég rendszer tekintetében.

**A teendőink:**

1. Gazda rendszermag készítése ***Felhasználói Módú Linux*** vendégrendszerek futtatására optimalizáltan
2. A gazdagépen futó egy (vagy több) vendég-rendszermag készítése
3. A vendég rendszerek számára készült gyökér-fájlrendszer testre szabása
4. Vendég rendszerek futtatása, konfigurálása és lezárása biztonságos ***DNS*** szolgáltatáshoz

## A gazdagép előkészítése

Először is meg kell győződni arról, hogy a gazdagépen megfelelő kernel fut-e. Nagy valószínűséggel újat kell majd fordítani a rendszermagot. Néhány ***Linux*** disztribúció rendszermagja alapértelmezetten tartalmazza a ***Felhasználói Módú Linuxot***. Nem biztos azonban, hogy a kiválasztott disztribúcióba a ***SKAS*** foltot is beleforditották. (A ***SKAS*** betűszó: ***Separate Kernel Address Space***, elkülönített kernel-címtér.) Az alapértelmezett kernelben sajnos általában nincs ***SKAS*** támogatás. Annak ellenére, hogy a kernel forrásában már a 2.6.9-es verziótól kezdve jelen van az ***FML*** támogatás, a ***SKAS*** foltot még külön tartják karban (***Linus Torvalds*** ugyanis nem fogadta el a beillesztését).
Fontos szerepe van a ***SKAS*** foltnak. Nagyban megnöveli az ***FML*** teljesítményét és biztonságát, ha a vendég rendszer kernelét az egyéb folyamataitól elkülönített címtérben futtatjuk (mint ahogy ezt a gazdagép is teszi). A ***Felhasználói Módú Linux SourceForge***-on található weboldala részletesen elmagyarázza a ***SKAS*** folt szükségességét. (Lásd a cikkhez tartozó forrásokat. ***Lényege, hogy segítségével a gazdagép kernelének nem kell foglalkoznia az FML processzeivel; az FML nem adja át a processzadatokat – a ford.***)

## A kernelek és a vendég rendszerek elkülönítése

A ***Felhasználói Módú Linux***, a ***VMware*** és más virtualizációs rendszerek szövegkörnyezetében sajátos jelentése van a gazdagép és vendég rendszer kifejezéseknek. A gazdagép az a rendszer, ami a virtualizációs környezetet futtatja; azaz végső soron egy vagy több virtuális gép gazdájaként működik. A vendég rendszerek olyan virtuális gépek, amelyek a gazdagép felett futnak.
A gazda kernel és a vendég kernel említésekor tehát tudnunk kell, hogy a vendég kernel a gazda kernelen fut. A ***Felhasználói Módú Linuxban*** a gazda kernel a normál kernel, melyet célszerűen az adott hardverre optimalizáltak (***Intel x86***, ***IBM PowerPC*** stb.), és amelybe beleforditották a ***Felhasználói Módú Linux*** támogatást (beleértve az opcionális ***SKAS*** foltot).
A vendég kernelt úgy kell lefordítani, hogy fusson a virtuális „hardveren”: az ***UM*** (***Felhasználói Módú***) architektúrán. De ezen túl ennek a kernelnek ***nincs*** szüksége a SKAS foltra vagy a Felhasználói Módú Linux támogatásra. Hacsak nem akar valaki a vendég rendszeren belül is újabb vendég kernel(eke)t futtatni, amely, bár lehetséges (ezt beágyazásnak (***nesting***) hívják), ennek tárgyalása túlmutat ezen cikk keretein.
Három összetevőt különíthetünk el minden ***FML*** virtuális gép példányban: a vendég kernelt, a vendég gyökér fájlrendszert és a ***COW*** fájlt (***Copy On Write***, íráskor létrejövő másolat). A gyökér fájlrendszer lehet egy lemez-képmásfájl, mely magán a kernelen kívül a virtuális gép minden fájlját tartalmazza. A vendég kernel futtatásakor a gyökér fájlrendszer fájlja pontosan úgy csatolható fel, mint ahogy bármely más képmásfájlt, például egy ***CD .iso*** fájlt is csatolni szoktunk. Ahogy a ***CD-ROM***, ez is csak olvasható lesz. Minden változtatás, ami a virtuális fájlrendszert érinti az ***FML*** munkamenet során (beleértve fájlok törlését és létrehozását is), a ***COW*** fájlban tárolódik. Ennek a „varázslatnak” köszönhetően lehetséges több példányban is futtatni ugyanazt a vendég kernelt akár ugyanazzal a vendég gyökér fájlrendszerrel együtt – elegendő pusztán annyit tenni, hogy különböző ***COW*** fájlt nevezünk ki az egyes ***FML*** példányok számára.
A kernel forrásának beszerzéséhez a legjobb választás valószínűleg az adott disztribúcióhoz elkészített kernelforrás-csomag telepítése. Arra azonban figyelni kell, hogy legalább 2.6.9-es rendszermag álljon rendelkezésre, mert az ***FML*** támogatás csak ettől kezdve tekinthető biztonságosnak. A ***Debian 3.1*** még ***2.6.8***-as rendszermagot használ, így arra az elhatározásra jutottam, hogy a hivatalos ***Debian*** kernelforrás-csomag helyett letöltöm a ***2.6.17***-es változatot a ***kernel.org*** webhelyről. A ***kernel-package*** csomag segítségével ebből a hivatalos kernelforrásból normál ***Debian*** (***.deb***) csomagot lehet készíteni.
A rendszermag forrásán kívül még a ***SKAS*** foltra is szükségünk lesz, melyből a legfrissebb elérhető a ***Blaisorblade*** webhelyen

(lásd a források között). A folt letöltésekor szem előtt kell tartani, hogy mely sorszámú kernel forrását szeretnénk megfoltozni.
*Debian* gazdagépemen kicsomagoltam a letöltött hivatalos kernelforrást a */usr/src/linux-2.6.17.3* könyvtárba, majd azon nyomban át is neveztem */usr/src/linux-2.6.17.3-host* nevűre, finom utalásféleképpen a készületben lévő gazdarendszerre. Ezek után a *SKAS* tarlabdát is a helyére tettem (a *skas-2.6.17-rc5-v9-pre9.patch.bz2*-t a */usr/src*-be). Meg kellett változtatnom a */usr/src/linux-2.6.17.3-host* könyvtár jogosultságát egy nem-adminisztrátori felhasználóéra azon lelkiismereti elv érvényesítése miatt, hogy „csak akkor legyünk adminisztrátorok, ha nagyon muszáj": a kernel felépítését ugyanis egy normál felhasználó is vígan el tudja végezni.
Az alábbi parancsokat adtam ki adminisztrátorként:

```
host:/usr/src/# tar -xjvf
➥./linux-2.6.17.3.tar.bz
host:/usr/src/# mv ./linux-
➥2.6.17.3 ./linux-2.6.17.3-host
host:/usr/src/# chown mick
➥./linux-2.6.17.3
host:/usr/src/# su - mick
```

A *SKAS* folt „odavarrásához" nem-adminisztrátor felhasználóként a */usr/src/linux-2.6.17.3-host* könyvtárba lépve az alábbi parancsot adtam ki:

```
host:/usr/src/linux-2.6.17.3
➥-host$ bunzip2 -c ../skas-
➥2.6.17-rc5-v9 -pre9.patch.bz2
➥| patch -p1
```

Majd ugyanebből a könyvtárból kiadtam egy `make menuconfig` parancsot is. A rendszermag Felhasználói Módú Linux számára történő konfigurálásához az alapértelmezett értékek többnyire megfelelőek – „csak" arra kell figyelni, hogy az adott gép hardveréhez is illeszkedjenek. Ezen kívül bölcs dolog kétszer is ellenőrízni az alábbiakat:

- A „Processzor típusa és egyéb jellemzői" részen a */proc/mm* támogatása legyen bekapcsolva.

- A „Hálózati beállítások" alatt az *IP:tunneling* (*IP-alagút*) és a *802.1d Ethernet Bridging* (*Ethernet híd létrehozása*) is szükséges. Ha az *iptables* által is szabályozni szeretnénk a vendég rendszerünk viselkedését, akkor ellenőrizni kell a „Hálózati csomagszűrés" szekciót; be kell kapcsolni az alábbiakat: *Core Netfilter Configuration, IP: Netfilter Configuration* és *Bridged IP/ARP packets filtering* opciókat *(Általános Hálózati Szűrő Beállítása, IP:Hálózati Szűrő Beállítása, Hídkapcsolt IP/ARP csomagszűrés)*.

- A „Hálózati eszközök"-nél kapcsoljuk be a *„Universal TUN/TAP device driver"* támogatást (*„Általános TUN/TAP eszköz meghajtó"*).

- Végül bizonyosodjunk meg arról, hogy bele legyen drótozva (ne pedig csak modulként elérhetően létezzen) azon fájlrendszerek támogatása, ami a rendszerünk gyökér-fájlrendszere (például *ext3*, *ReiserFS*).

Ettől a ponttól kezdve ugyanúgy folytatódik minden, mint bármely más kernelfordításkor: adjuk ki a

```
make bzImage; make modules;
```

parancsokat, majd adminisztrátorként a

```
make modules, make
➥modules_install
```

és

```
make install
```

utasításokat. *Debian* esetén használható a `make-kpkg` parancs is (azaz *„make Kernel-PacKaGe"*, *„csinálj kernelcsomagot"*) ugyanilyen céllal, majd `dpkg`-vel telepíthető a kapott kernel csomag.
Ha elkészült friss-ropogós gazdarendszermagunk, indítsuk újra számítógépünket. A lábra álló gazdarendszer már alkalmas lesz *Felhasználói Módú Linux* vendég rendszerek futtatására.

## A vendég-rendszermag előállítása

Ezek után, hogy készen áll gazdagépünk az *FML* használatára, még szükségünk van vendég-rendszermagra is. Ez egy fokkal egyszerűbb, mint a gazda-rendszermag előállítása, mivel nem kell foglalkoznunk a *SKAS* folttal.
Először térjünk vissza ahhoz a könyvtárhoz, ahol a kernelforrásunk tarlabdáját tartjuk, és újra tömörítsük ki. Talán emlékszünk még arra, hogy ezek után egy átnevezéssel utaltunk a kernelforrás hovatartozására – ezt most is hasonlóképpen fogjuk tenni. A gazda- és vendég-rendszermagunk könyvtárait elkülönítetten kell használnunk.
*Debian* tesztrendszeremen a */usr/src/linux-2.6.17.3* könyvtárba tömörítettem ki a tarlabdát, majd ezt a könyvtárat */usr/src/linux-2.6.17.3-guest*-nek neveztem át. Újfent gyengítsük le a könyvtár jogosultságát egy kiskaliberű felhasználóéra, és váltsunk bele.
Ezúttal eltekinthetünk a SKAS folt beépítésétől. Mivel kernelünk nem egy normál (például *x86*-os) architektúra számára készül, hanem egy speciális *UM* (*User Mode, Felhasználói Módú*) architektúra számára, a forrás megfelelő előkészítéséhez javaslom az alábbi parancsok kiadását:

```
host:/usr/src/linux-2.6.17.3-
➥guest$ make mrproper ARCH=um
host:/usr/src/linux-2.6.17.3-
➥guest$ make defconfig ARCH=um
host:/usr/src/linux-2.6.17.3-
➥guest$ make menuconfig ARCH=um
```

A `make mrproper` parancs mindenféle konfigurációs előzményt és objektumot eltakarít a kódkönyvtárból; a `make defconfig` elkészíti a friss alapkonfigurációs fájlt az *um* architektúra számára; végül a `make menuconfig`, elvárásainknak megfelelően, lehetővé teszi eme konfigurációs fájl finomhangolását.

Különösen is figyeljünk az alábbiakra:

- Egyszerűbb az élet monolitikus kernellel (modulok betöltögetése helyett). Ha netán mégis ez lenne a cél, érdemes elolvasni a *User-Mode Linux HOWTO* (*Felhasználói Módú Linux Hogyan*) 2.2-es fejezetét (lásd a források közt).

- A „Processzor típusa és egyéb jellemzői" részen győződjünk meg (kétszer is) arról, hogy architektúraként az *um*-et (*User Mode,*

*Felhasználói Módú*) adtuk-e meg, s hogy a */proc/mm* támogatás be van-e kapcsolva.

- A „Hálózati beállítások” alatt az ***IP:tunneling*** (***IP-alagút***) és a ***802.1d Ethernet Bridging*** (***Ethernet híd létrehozása***) legyen bekapcsolva.

- A „Hálózati eszközök”-nél kapcsoljuk be a „***Universal TUN/TAP device driver***” támogatást („***Általános TUN/TAP eszköz meghajtó***”).

- Az összes lehetséges hardver kernelmodult kapcsoljuk ki; ez a kernel virtualizált hardveren fog futni, így nem lesz szükség semmi egzotikus drótnélküli LAN támogatásra, sem őskori párhuzamos portra stb.

A friss konfigurációs fájl elmentése után az alábbi parancsot mezei felhasználóként adjuk ki (tehát nem adminisztrátorként):

```
host:/usr/src/linux-2.6.17.3-
➥guest$ make linux ARCH=um
```

Talán feltűnt, hogy ***nem*** javasoltam a (***zip*** vagy ***bz2*** módon) tömörített rendszermag-képfájl (***image***) előállítását. Gondoljunk arra, hogy ez a kernel úgy fog futni, mint bármely más, normál felhasználó által kiadott parancs – nem érdemes tehát tömöríteni. A kész rendszermag a forráskód főkönyvtárában helyezkedik el a fordítás befejeztével (***/usr/src/linux-2.6.17.3-guest*** a fenti példában), és nemes egyszerűséggel ***linux*** lesz a neve. Ezt célszerű átnevezni valami kifejezőbbre, például ***uml-guestkernel-2.6.17.3***-re, valamint érdemes át is mozgatni valahova máshová, pl. a ***/usr/local/uml/*** könyvtárba. Ne ijesszen meg senkit a vendégrendszermag mérete. A bithalmaz mérete nagyrészt szimbólumokból áll, ami nem töltődik be a memóriába a végrehajtáskor.

## Tanulság

Gazdagépünk immár teljes mértékben támogatja a ***Felhasználói Módú Linuxot***, és egy futtatásra készen álló vendég-rendszermagot is elkészítettünk. Már csak egy megfelelő gyökérfájlrendszer képmásfájl beszerzése vagy elkészítése van hátra, melyet a vendég-rendszermaggal fogunk tudni használni. Innen folytatjuk következő alkalommal.

*Linux Journal 2006., 151. szám*

***Mick Bauer***
(darth.elmo@wiremonkeys.org)
az Egyesült Államok egyik legnagyobb banki hálózatában felelős a hálózati biztonságért. Ő a szerzője az O'Reilly által kiadott Linux szerverek biztonsága, Linux Server Security c. mű második kiadásának. (Korábban a Biztonságos szerverek építése Linuxon, Building Secure Servers With Linux címet viselte a könyv.) Időnként előad információ-biztonsági konferenciákon. Ő a szerzője a „Hálózatmérnöki Polkának” („Network Engineering Polka”).

## A CIKK FORRÁSAI

➲ www.linuxjournal.com/article/9260