FreeBSD – a szomszéd vár (9. rész)
A várbörtön cellái

Gyakran kerülhetünk olyan helyzetbe, amikor meg kell védenünk a rendszerünket egy szolgáltatástól, esetleg önmagától, vagy a többi programtól, a külső-belső támadásoktól, illetve a sérülékeny vagy könnyen támadható az adatokat mindezektől. A probléma megoldása egyszerű, a szolgáltatásokat bezárjuk egy börtön cellájába: ez a jail alrendszer.

Mi az a jail?

A ***Linux*** esetén már ismert `chroot` program távoli rokona, amellyel ***FreeBSD*** alatt tudunk programokat elzárni a rendszer többi részétől; egy kicsit másképp és egy kicsit jobb körülmények között. A `jail` ugyanis ***FreeBSD*** virtuális gépeket hoz létre, amelyeknek közös rendszermaggal bírnak, viszont minden mást külön kezelnek. A gépen kívülről nézve egy `jail` alrendszer olyan, mintha lenne még számítógépünk, amelyeken ugyan az a ***FreeBSD*** van, mint a mellette lévőkön (holott csak egy gépünk van). Minden virtuális gépnek van egy külön ***IP*** címe, amelyre hallgat, a rendszer többi részével csak (virtuális) hálózati felületeken tud kommunikálni, leszámítva a megadott könyvtárstruktúrát, amelyek a rendszermagon keresztül képes kezelni. Azt tudom mondani, hogy a ***FreeBSD*** `jail` megvalósítása többet tud, mint a ***Linux*** chroot (teljesen elszeparált virtuális gép); kevesebbet tud, mint az ***UML (User Mode Linux)***, mivel a rendszermag közös, az nem cserélhető.

Miképp készül?

A bebörtönzést elő kell készíteni, amely tevékenység alatt az alaprendszer könyvtárainak és állományainak a `jail` könyvtárába való másolását értjük; ezen túl néhány beállítás is szükséges lehet, hogy a bebörtönzött folyamat megfelelően tudjon működni.

Általában két módszer közül választhatunk, mind a kettő célravezető, de más-más módon éri el a biztonságos üzemeltetést. Megtehetjük, hogy a teljes telepítés után eltávolítjuk az állományok és könyvtárak egy részét, egészen addig, amíg ***még*** működőképes a bebörtönzött program (kövér modell). Esetleg egyenként másoljuk át az állományokat, és a könyvtárakat, egészen addig, amikor ***már*** működőképes a program (***sovány*** modell). Akár a kettő módszert kombinálhatjuk is, felmásolunk egy-egy könyvtárat, majd elkezdjük törölgetni a mappából az állományokat. Az interneten keresgélve nagyon valószínű, hogy megtaláljuk a kész recepteket az adott szolgáltatás működéséhez minimálisan szükséges állományok listájával együtt.

A börtön felépítését sok különféle módon tehetjük meg, ebből három módszert szoktunk használni: ***mezei*** másolás, a `make world` parancs segítségével, illetve egy külön telepített programot használva.

A mezei másolás okán egyszerűen átmásoljuk a `jail` könyvtárába azokat az állományokat, amelyek szükségesek a bezárandó programunk futásához (és magát a futtatandó programot is). Gyakorlatilag ez a legegyszerűbb megoldása a börtönünk létrehozásának, bár kétségkívül nagyobb szakértelmet kíván. Sajnálatos, hogy a börtön karbantartása is nehezebb így, mint a többi módszert használva.

A klasszikus (leginkább támogatott) módszer szerint egy új alaprendszert építünk fel a börtön egy-egy cellájában. Teljesen azonosan kell eljárnunk, mint az alaprendszer frissítésénél, egyetlen paraméterrel kell többet megadnunk: hol készüljön el az új alaprendszer, illetve néhány más parancsot is végre kell hajtanunk.

```
# export JAILDIR=/usr/local/jails/10.1.1.213
# cd /usr/src
# mkdir -p $JAILDIR
# make world DESTDIR=$JAILDIR
# cd $JAILDIR
# ln -sf dev/null kernel
```

Ezzel a tevékenységgel számítógépünk ugyan annyi időt tölt el, mint az alaprendszer frissítésével, tehát akár több órán át is tarthat. Ha gyakran kell `jail` rendszert elkészítenünk, akkor érdemes egy példányt eredeti állapotában megtartani, s így csak át kell másolnunk a teljes könyvtárszerkezetet. Ezzel a módszerrel egy börtöncella mérete azonos lesz a teljes alaprendszer méretével, vagyis közel 200MBájt helyet fog elfoglalni. Ezen a hatalmas méreten sokat tudunk csökkenteni, ha a felesleges részeket eltávolítjuk (dokumentáció, kézikönyv oldalak, stb.).

Második legnagyobb szakértelmet kívánó kihívás a `sysinstall` program használata a `jail` beállításához. Ekkor alapvetően egy programot kell felmásolni a börtönünk könyvtárába: a `/stand/sysinstall` állományt.

```
# mkdir $JAILDIR/stand
# cp /stand/sysinstall $JAILDIR/stand
```

A bebörtönzés elindításához szükséges a helyi hálózathoz hozzáadni egy IP címet (például `10.1.1.213`), ahol elérjük majd a börtönükbe bezárt programjainkat.

```
# ifconfig vr0 alias 10.1.1.213/32
```

Két fontos könyvtár van, amit nem tudunk könnyedén létrehozni, a `dev` és a `proc`, amelyek speciális fájlrendszert hordoznak: a `devfs` és a `procfs` nevűt. Ezt a két fájlrendszert fel kell csatolnunk msinden egyes jail könyvtárba.

```
# mkdir dev
# mount_devfs devfs $JAILDIR/dev
# mkdir proc
# mount_procfs proc $JAILDIR/proc
```

Az alaprendszer elkészítése nem vonja maga után a */etc* könyvtár elkészítését (amely teljesen érthető a `make world` alapvető feladatát tekintve), így ezt külön parancs segítségével kell nekünk elkészítenünk.

```
# cd /usr/src/etc
# make distribution DESTDIR=$JAILDIR
```

Nincs más hátra, mint beállítani a börtön kényelmi szolgáltatásait, ehhez már be kell térnünk a cellába is. A `jail` parancs szolgál a bebörtönzés elindítására, amelynek az első paramétere az elkészített könyvtárszerkezet belépési pontja, a második paramétere a börtön neve, harmadik paramétere a kiválasztott IP cím, majd ezeket követi a végrehajtandó parancs elérési úttal együtt.

```
# jail /usr/local/jails/10.1.1.213/ private213
➥10.1.1.213 /rescue/sh
```

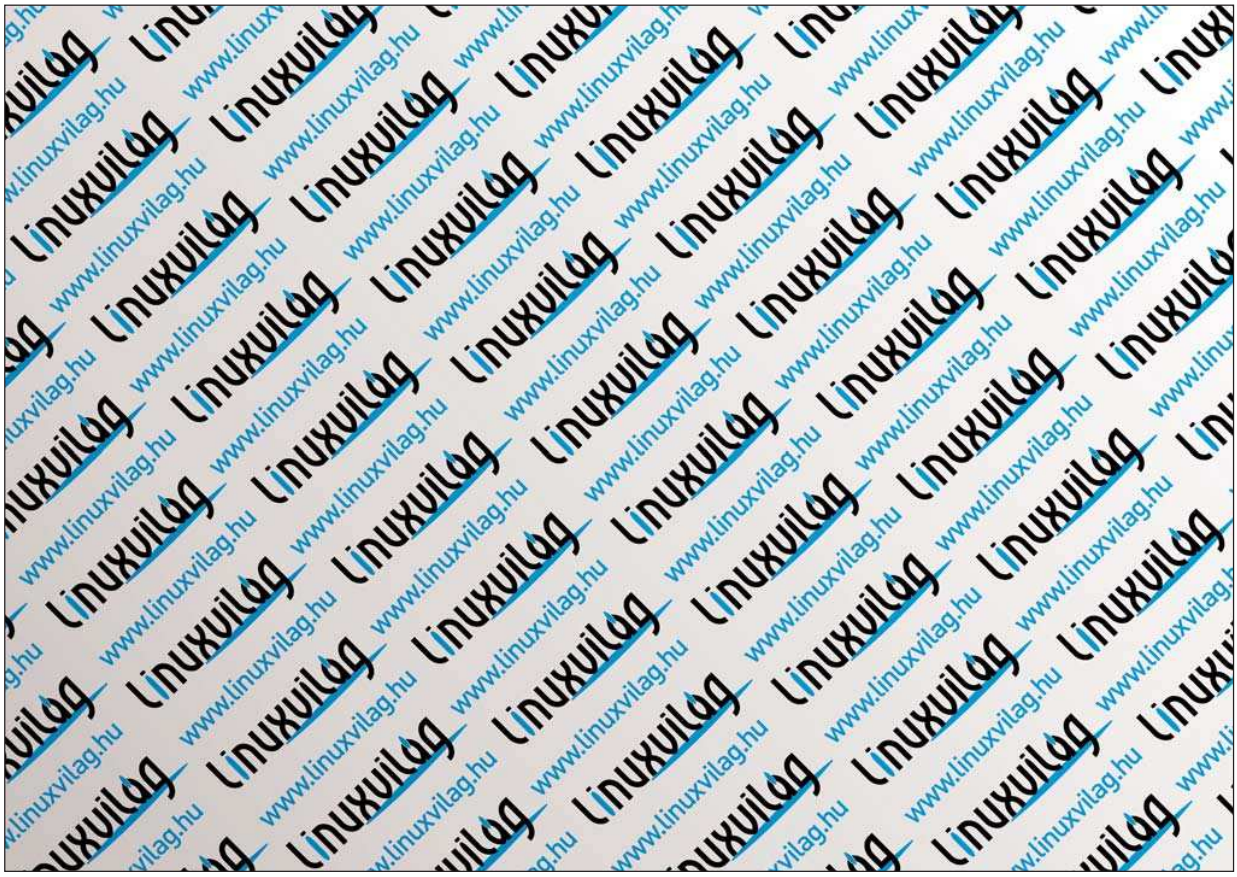
A kapott üres parancssorba kezdésképp a `sysinstall` programot írhatjuk.

```
$ /stand/sysinstall
```

Nem kell meglepődni, megkapjuk a ***FreeBSD*** telepítésénél is kapott programot, amellyel most újra feltelepíthetjük a ***FreeBSD*** rendszerünket: a `jail` alrendszer ugyanis olyan, mint egy teljesen új telepítés. A rendszermag már fut, az alaprendszer rendelkezésre áll, de semmi egyéb nincs beállítva, engedélyezve: mindent újra be kell állítanunk és engedélyeznünk. Az első használatbavétel előtt – amikor már a bebörtönzött programunkat szeretnénk futtatni – érdemes a `/sbin/init` programot elindítani a `jail` első beállításához (***SSH*** kulcsok elkészítése, stb.). Nem kell megijedni, a képernyőkép teljesen azonos lesz, mint amikor a ***FreeBSD*** rendszerünk elindul, hiszen a virtuális gépünkben is egy ***FreeBSD*** indul el. Programok telepítése és karbantartása azonos módon történik, mint azt a börtönünkön kívül tettük, így az alaprendszer frissítése és a `ports` adatbázis is járható út.

Segédprogramok

Érdemes néhány segédprogramot telepíteni, amelyek sokat segítenek a börtönünk felügyeletében. Néhány



programot a börtöncellába kell telepíteni, némelyiket a börtönön kívülre. A legfontosabb a ***jailutils*** csomag, amely a börtönön belül futó programok kezelésében segít (`jps`, `jtop`, `jkill`), mivel a `jail` belső mechanizmusai ezen programok normál futását jelentősen befolyásolják. A `jailadmin` vagy a `jailctl` csomag a börtönök elindításában, leállításában; illetve elkészítésében is segítséget nyújtanak.

A börtöncella beállítása

Fontos tulajdonsága a `jail` rendszernek, hogy nem hallgatózik a számítógépünk összes elérhető hálózati felületén, csak a parancssorban megadott IP címen. Ha megnézzük a börtönünkön kívül a `vr0` eszközhöz rendelt címeket, akkor láthatjuk, hogy több címe is van, azonos fizikai címmel.

```
# ifconfig vr0
vr0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
➥mtu 1500
        inet 10.1.1.211 netmask 0xffffff00
        ➥broadcast 10.1.1.255
        inet6 fe80::211:5bff:fe09:782a%vr0
        ➥prefixlen 64 scopeid 0x2
        inet 10.1.1.213 netmask 0xffffffff
        ➥broadcast 10.1.1.213
        ether 00:11:5b:09:78:2a
```

A `jail` rendszeren belül szintén megtaláljuk ezt az eszközt, de már csak a saját ***IP*** címével, a többi cím nem is látszik.

```
$ ifconfig vr0
vr0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
➥mtu 1500
        inet 10.1.1.213 netmask 0xffffffff
        ➥broadcast 10.1.1.213
        ether 00:11:5b:09:78:2a
```

Ha ki szeretnénk próbálni a jail hálózatának működőképességét, nem várt problémába ütközhetünk.

```
$ ping 10.1.1.212
ping: socket: Operation not permitted
```

A `ping` működéséhez igazzá kell állítani a `security.jail.allow_raw_socket` rendszerváltozót még a `jail` elindítása előtt.

```
# sysctl -a security.jail.allow_raw_sockets=1
security.jail.allow_raw_sockets: 0 -> 1
# jail /usr/local/jails/10.1.1.213/ private213
➥10.1.1.213 /rescue/sh
$ ping 10.1.1.212
PING 10.1.1.212 (10.1.1.212): 56 data bytes
64 bytes from 10.1.1.212: icmp_seq=0 ttl=64
➥time=7.635 ms
```

E nélkül csak a ***TCP*** és az ***UDP*** csomagokat jutnak ki a börtönből, ha például ***ICMP*** csomag is szükséges, akkor ezt a beállítást tegyük meg. Ez az engedélyezés viszont biztonsági kockázatot is jelent, így csak végszükség esetén használjuk; célravezetőbb egy olyan helyettesítő eszköz használata, amely ***TCP*** porton át is képes a hálózat működőképességét ellenőrizni (például az ***echoping*** csomag).
A hálózat többi beállítása (gazdagép neve, útválasztás, névfeloldás, stb.) azonos módon történik az alaprendszer telepítéskori beállításaival, használjuk nyugodtan erre a célra a `sysinstall` programot.

Lemezkezelés

A börtönben futó program nem látja a külsőleg felcsatol állományrendszereket, csak a saját főkönyvtárát tartalmazó fájlrendszert tudjuk lekérdezni.

```
$ df
Filesystem  1K-blocks      Used      Avail Capacity
➥Mounted on
/dev/ad0s1f 150307738 16267994 134039744     11%
➥/usr
```

Néhány tanács és pár megjegyzés

A `jail` nem óvja meg a gépünket a feltöréstől, mindössze – helyes tervezés esetén – az érzékeny adatokat védi meg a támadótól. Minél több `jail` készül el a gépünkön, annál nehezebb lesz azok karbantartása, hiszen egy-egy alaprendszert érintő biztonsági hiba kijavítását az alaprendszerünkön és az összes `jail` rendszerünkön is végre kell hajtanunk. Érdemes a `jail` belső felhasználóit leredukálni minimálisra, hiszen a legtöbb inaktív lesz. A `jail` alkalmas arra, hogy a teljes ***FreeBSD*** rendszer rendszergazda (root) jogú adminisztrálását kiadjuk egy másik személynek. Ő képes belépni a megadott IP címen, minden frissítést, telepítést és beállítást képes elvégezni, viszont nem képes az alaprendszer is érintő beállítások elvégzésére (particionálás, fájlrendszer elkészítése, más fájlrendszer felcsatolása, hálózati eszközök átállítására, stb.). Igazából egy olyan virtuális gépet tudunk készíteni a kiszolgáló gépünkön, amelyet egy ***FreeBSD*** rendszert ismerő személy képes kezelni, s úgy tekinthet rá, mint egy önálló számítógépre.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja: ➲ http://www.freebsd.org

A magyar FreeBSD honlap: ➲ http://www.freebsd.hu

A magyar BSD honlap: ➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/