Az Apache2 börtönbe zárása manuálisan!

Hát akkor készítsünk először egy alap rendszer struktúrát a kiválasztott jail könyvtár alá a debootstrap utasítással.

```
server:/#debootstrap --arch i386 etch /var/apache2chroot
```

Másoljunk át, ellenőrizzünk pár fontos konfigurációs állományt:

```
server:/var/apache2chroot#cp /etc/apt/sources.list
/var/apache2chroot/etc/apt/sources.list
server:/var/apache2chroot#cat
/var/apache2chroot/etc/resolv.conf
server:/var/apache2chroot#cat
/var/apache2chroot/etc/hostname
```

Adjunk új sorokat az /etc/fstab állományunkba:

```
proc /var/apache2chroot/proc proc defaults 0 0
```

A temp-et is fel lehet mountolni a jailbe újra, de nem kötelező.

```
/dev/sda1 /var/apache2chroot/tmp tmpfs,nodev,nosuid,noexec
defaults 0 2
```

Sőt a /var/www-t is akár read-only módban is ha a weboldalunk megengedi.

Mountoljuk be az előzőleg megadott pontokat:

```
server:/var/apache2chroot#mount /var/apache2chroot/proc
server:/var/apache2chroot#mount /var/apache2chroot/tmp
```

Nah most chroot-oljuk be magunk az új környezetünkbe:

```
server:/var/apache2chroot#cd /var/apache2chroot/
server:/var/apache2chroot#chroot .
```

Telepítsük fel jailen belül az apache2 csomagot és többi csomagot amit a jailben akarunk használni.

```
server:/#apt-get update
server:/#apt-get upgrade
server:/#apt-get install apache2 libapache2-mod-php4 php4-
mysql php4-ldap php4-imagick php4-gd php4-gd2 php4-pear
```

Ugye a fő indoka jailbe zárásnak eme utolsó csomaghalmaz sebezhetőségi pontjai.
Ugorjunk ki a jailből és linkeljük vissza a telepített könyvtárakat a rendes rendszerbe a későbbi könnyű kezelhetőség miatt.

```
server:/#exit
server:/var/apache2chroot#ln -s
/var/apache2chroot/etc/apache2 /etc/apache2
server:/var/apache2chroot# ln -s
/var/apache2chroot/var/www/ /var/www
server:/var/apache2chroot# ln -s
/var/apache2chroot/var/log/apache2 /var/log/apache2
```

Csináljunk egy inditó szkriptet a jailen belüli apache indítására(ami egyszerűen bepasszolja az öszes indító paramétert a jailen belülre.)

```
nano /etc/init.d/apache2
#!/bin/sh -e
#
# Ez a Skript indítja el JAIL-en bellül az apache2-öt.
case $1 in
start)
chroot /var/apache2jail /etc/init.d/apache2 start
;;
stop)
chroot /var/apache2jail /etc/init.d/apache2 stop
;;
reload)
chroot /var/apache2jail /etc/init.d/apache2 reload
;;
restart | force-reload)
chroot /var/apache2jail /etc/init.d/apache2 restart
;;
*)
echo "Usage: /etc/init.d/apache2
start|stop|restart|reload|force-reload"
;;
esac
```

Majd hozzuk létre rá az indító linkjeinket:

```
update-rc.d apache2 start 91 2 3 4 5 . stop 91 0 1 6 .
```

Ezzel adjuk meg, hogy milyen runlevelen, milyen indítási sorszámmal induljon el, valamint a 0,1,6 runlevelnél milyen sorszámmal álljon meg a daemon.

Az Apache2 börtönbe zárása jailer szkript segítségével.

A balabit Kft. 2004-ben készített egy szkriptet ahhoz, hogy a fent említett rengeteg lépést elkerüljük. A lényeg hogy egy xml-hez hasonló konfigurációs állomány segítségével legenerálja a szükséges környezetet ahhoz, hogy az adott daemon zökkenőmentesen fusson. Ehhez némi hibakeresési ismeret szükséges. Azon kívül, hogy a rengeteg lépést átugorhatjuk (vagyis a jalier megcsinálja) a kialakított környezet mérete is jelentősen csökkenthető. A manuálissal konfigurálással ellentétben nem egy egész rendszert, hanem kifejezetten csak a szükséges állományok kerülnek be a chroot-olt környezetbe.

Két nagyon fontos „tool” áll rendelkezésre egy ilyen konfigurációs állomány elkészítsünk (a linux ismereteinek felül, hisz nyilvánvaló, hogy például a proftpd deamonnak kell a */etc/passwd*, valamint a

/etc/hostname fájl). Az első ilyen program az **ldd**, mely egy adott indító binárisból (pl.: */usr/sbin/apache2*) kilistázza az indításához szükséges dependenciánkat.

A fent említett példa az apache2-re nézve:

```
dev2:/# ldd /usr/sbin/apache2
        linux-gate.so.1 =>  (0xffffe000)
        libm.so.6 => /lib/tls/i686/cmov/libm.so.6 (0xb7f01000)
        libpcre.so.3 => /usr/lib/libpcre.so.3 (0xb7edb000)
        libaprutil-1.so.0 => /usr/lib/libaprutil-1.so.0 (0xb7ec0000)
        libldap_r.so.2 => /usr/lib/libldap_r.so.2 (0xb7e8b000)
        liblber.so.2 => /usr/lib/liblber.so.2 (0xb7e7f000)
        libdb-4.4.so => /usr/lib/libdb-4.4.so (0xb7d84000)
        libpq.so.4 => /usr/lib/libpq.so.4 (0xb7d67000)
        libsqlite3.so.0 => /usr/lib/libsqlite3.so.0 (0xb7d0c000)
        libexpat.so.1 => /usr/lib/libexpat.so.1 (0xb7ceb000)
        libapr-1.so.0 => /usr/lib/libapr-1.so.0 (0xb7cc8000)
        libuuid.so.1 => /lib/libuuid.so.1 (0xb7cc5000)
        librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0xb7cbc000)
        libcrypt.so.1 => /lib/tls/i686/cmov/libcrypt.so.1 (0xb7c8e000)
        libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0xb7c7c000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7c77000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7b46000)
        /lib/ld-linux.so.2 (0xb7f32000)
        libresolv.so.2 => /lib/tls/i686/cmov/libresolv.so.2 (0xb7b33000)
        libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0xb7b1d000)
        libgnutls.so.13 => /usr/lib/libgnutls.so.13 (0xb7aaf000)
        libssl.so.0.9.8 => /usr/lib/i686/cmov/libssl.so.0.9.8 (0xb7a6f000)
        libcrypto.so.0.9.8 => /usr/lib/i686/cmov/libcrypto.so.0.9.8 (0xb7935000)
        libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0xb78b9000)
        libcom_err.so.2 => /lib/libcom_err.so.2 (0xb78b6000)
        libtasn1.so.3 => /usr/lib/libtasn1.so.3 (0xb78a3000)
        libz.so.1 => /usr/lib/libz.so.1 (0xb788e000)
        libgcrypt.so.11 => /usr/lib/libgcrypt.so.11 (0xb783d000)
        libgpg-error.so.0 => /usr/lib/libgpg-error.so.0 (0xb7839000)
        libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0xb7814000)
        libnsl.so.1 => /lib/tls/i686/cmov/libnsl.so.1 (0xb77fe000)
        libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0xb77f8000)
```

Látható, hogy a legtöbb ilyen dependencia library-kból áll ezért célszerű a rendszerben található **`lib`** könyvtárakat belerakni a konfigurációs állományba, ez nem csökkenti a környezet biztoságát.

A másik ilyen eszköz a **strace** mely kiírja a indító bináris rendszerhívásait.

Használata:

```
strace /usr/sbin/apache2
```

FIGYELEM: az strace nyomon követhetetlen, ha a binárisnak megvan minden működéshez szükséges környezeti változója, illetve konfigurációs állománya. Használata csak nem működő daemonnál javasolt, sokkal több információt nyújthat, mint például a syslog.

További információkat az *apt-cache show <csomagnév>* paranccsal tudunk szerezni, arról milyen csomagok kellenek még az adott daemon működéséhez.
Maga a jailer szkript telepíthető a debian tükörről, mely alapértelmezetten létrehoz egy general konfigurációs állományt a */etc/jailer.conf*-ban. Mivel a létrehozott chroot-olt környezet sok esetben symlinkeket hoz létre a „gazdarendszerből" ezek érvényre juttatásához a jailer csomag által tartalmazott **updatejail** parancsot használhatjuk.

Ezek után egy példa mely létrehoz egy chroot-olt környezetet az apache2 számára:

```
<general>
Junk: /usr/doc/* /usr/man/* /usr/share/man/* /usr/share/doc/*
/etc/init.d/* /usr/share/zoneinfo/* /sbin/ldconfig.new /etc/*
</general>

<apache2>
Root: /var/chroot/apache2
Conf: /etc/apache2/*
Debs: apache2 strace
Junk-Debs: tcpd arpd
Extra: /dev/null /dev/log /etc/hostname /etc/resolv.conf
/etc/localtime /etc/mime.types /usr/lib/lib* /etc/passwd
/etc/nsswitch.conf /etc/group /lib/apache2/* /usr/lib/apache2/*
/lib/libgcc_s.so.1 lib/tls/i686/cmov/libm.so.6

Junk: /usr/share/* /sbin/* /usr/sbin/arp /usr/sbin/arping
/usr/sbin/inetd /usr/sbin/ipautofw /usr/sbin/ipmasqadm
/usr/sbin/tzconfig /usr/sbin/updateinetd/usr/sbin/zic /usr/lib/*
/bin/* /usr/bin/tzselect /usr/bin/ldd /usr/bin/getent /usr/bin/zdump
/lib/libwrap* /lib/libm* /lib/libcrypt* /lib/libthread* /lib/libutil*
/lib/librt* /lib/libpthread* /lib/libnss* /lib/libdb* /lib/libdl*
/lib/libBrokenLocale* /lib/libnsl* /lib/libSegFault* /lib/libresolv*
/usr/sbin/iconv /usr/sbin/local /usr/bin/rpc* /usr/bin/trace*
</apache2>
```