Az Apache biztonságos beállítása

Egy neves IT cég hirdetése szerint naponta többen auditálják hálózatunkat – sajnos nem mindenki a mi megbízásunkból. A biztonság nem fölösleges luxus, hanem az a nélkülözhetetlen kiinduló állapot, ami nélkül ne is reméljünk jövedelmező online jelenlétet.

Az ***Apache*** része a legtöbb ***Linux*** terjesztésnek, így ma már mindenkinek lehet web kiszolgálója. De biztonságos web kiszolgálót nem is olyan egyszerű építeni. Írásomban egy (viszonylag) biztonságos ***LAMP*** (=***Linux, Apache, MySQL*** és ***PHP***) konfiguráció kialakítását mutatom be.
Már rögtön az elején jó szem előtt tartani, hogy a biztonság mindig csak relatív lehet, az abszolút biztonság, mint olyan nem létezik. Tisztában kell lennünk azzal, hogy ha egy támadónak megfelelő erőforrása (idő, pénz, számítási teljesítmény, …) van, akkor be tudja venni a várat.
Egy átlagos betörés jellemzően információ gyűjtéssel kezdődik, a támadó apró, lényegtelennek tűnő mozaikdarabokból összeállítja a cél profilját, kiválasztja a behatolási pontot, majd azon keresztül bejön, elveszi, amit akar, törli a nyomokat, majd angolosan távozik.
A biztonságos kiszolgálónkat ezért úgy alakítjuk ki, hogy a lehető legkevesebb információt szivárogtassuk ki, és a lehető legkevesebb dologhoz engedünk hozzáférést a világnak.
Bruce Schneier elhíresült mondása szerint a biztonság nem egy termék, hanem egy folyamat. A biztonság számtalan láncszemből áll össze, és talán nem árt hangsúlyozni azt a közhelyet, hogy olyan erős a lánc, mint a leggyengébb láncszem.
Sok éve már, hogy egy ifjú titán magabiztosan állította, hogy az ő gépén ***Mandrake*** fut, úgyhogy jobb, ha előre feladja bárki. Én ennél egy kissé óvatosabb vagyok, és azt tanácsolom, ne ringassuk magunkat abban a hamis illúzióban, hogy csak azért, mert ***Linux*** és ***Apache*** fut a gépünkön, nem érhet bennünket egy idegen nyelvű felirat kellemetlen híre egy borús reggelen.
Csábító a lehetőség, hogy telepítsük a ***Linux*** terjesztésünkben található ***Apache*** és ***PHP*** csomagot. Azonban ezek jellemzően (szinte) minden funkcionalitást tartalmaznak, ami nekünk nem jó. ***Wietse Venema*** írta valahol, hogy a ***Postfix*** kódjának minősége kb. 1 hiba (***bug***) 1000 soronként. Ha ezt átlagnak tekintjük, könnyen belátható: minél több (felesleges) funkció van egy programban, statisztikailag annál több (kihasználható) hiba lehet benne. Ezért csomagból csak a ***MySQL*** programot telepítsük.
Kedvenc ***Linux*** terjesztésünk telepítése során alakítsunk ki megfelelő méretű ***/***, ***/usr***, ***/var***, ***/tmp***, ***/home*** partíciókat, és készítsünk egy nagy ***/opt*** partíciót, ez utóbbi fogja tárolni a web kiszolgálónkkal kapcsolatos összes állományt.

Készítsünk ketrecet!

Miként a veszélyes állatokat ketrecben tartják, úgy a veszélyes programot is célszerű bezárni, hogy ha a támadónak sikerül – pl. egy programhiba segítségével – átvenni felette az ellenőrzést, ne okozhasson (még) nagyobb bajt, ne férhessen hozzá a gép más részeihez. A ***chroot()*** rendszerhívás ill. a ***chroot*** parancs hatására az adott program az argumentumban megadott elérési utat látja a továbbiakban gyökérkönyvtárként, és csak az az alatt található dolgokhoz férhet hozzá.
Készítsük el első lépésben a „ketrecet” a ***/opt/jail*** könyvtár alatt! Az *1. Listában* szereplő ***makejail.sh*** héjprogram segít ebben, argumentumként a kezdőkönyvtárat kell megadni, például az

```
sh makejail.sh /opt/jail
```

parancs a ***/opt/jail*** alatt hozza létre a szükséges környezetet. Az Olvasó gépén esetleg más állományokra is szükség lehet.
Ez a héjprogram létrehoz egy minimális környezetet, amely az ***Apache*** futtatásához feltétlen szükséges. Bizonyára feltűnt az Olvasónak, hogy a ***/etc*** könyvtárban nem szerepel jelszavakat tartalmazó állomány (például ***/etc/shadow***), és a ***/etc/passwd*** állományban is csak a nélkülözhetetlen felhasználók szerepelnek. Hogy még keményebb legyen a dió, a ***/etc*** könyvárban mindenre bekapcsoljuk az úgynevezett ***immutable*** attribútumot (`+i`), amely megakadályozza, hogy az itt lévő állományok megváltozhassanak. Ezután a program bemásol néhány könyvtárat (***library***), amelyek az ***Apache*** működéséhez szükségesek.
Nagyon fontos, hogy a ***chroot*** könyvtárba csak a valóban nélkülözhetetlen állományokat helyezzük el. Ne feledjük, minél kevesebb dolgot talál itt a támadónk, annál kevesebb kárt tud okozni. Azt is szeretném itt megjegyezni, hogy a ***root*** felhasználó ill. az ő jogaival futó programok kitörhetnek a ketrecből, ezért hacsak különösen jó okunk nincs rá, semmiképpen ne helyezzünk itt el ***setuid/setgid*** programokat.

1. Lista Az mkjail.sh héjprogram

```
#!/bin/sh
##
## mkjail.sh

if [ $# -ne 1 ]; then echo
➥"usage: $0 <jail directory>";
➥exit 1; fi

if [ ! -d $1 ]; then mkdir -p
➥$1; fi

cd $1 || exit 2;

mkdir dev etc lib usr www

# néhány device is szükséges
mknod dev/null c 1 3
chmod 666 dev/null
mknod dev/hwrandom c 10 183

cp /etc/resolv.conf etc
cp /etc/localtime etc

# csak a legszükségesebb
➥információt tesszük be
➥a kalitkába
echo "root::0:" > etc/group
echo "nobody::98:" >> etc/group
echo "nogroup::99:" >> etc/
➥group
echo "httpd::7002:" >> etc/
➥group

echo "root:x:0:0::/:/bin/sh" >
➥etc/passwd
echo "nobody:x:99:99:nobody:/
➥oblivion:/bin/false" >> etc/
➥passwd
echo "httpd:x:7002:7002::/
➥oblivion:/bin/false" >> etc/
➥passwd

echo "127.0.0.1 localhost" >
➥etc/hosts

# immutable attribútum
# bekapcsolása
chattr +i etc/*

cp /lib/ld-linux.so.2 lib
cp /lib/libc.so.6 lib
cp /lib/libcrypt.so.1 lib
cp /lib/libdl.so.2 lib
cp /lib/libm.so.6 lib
cp /lib/libnsl.so.1 lib
cp /lib/libnss_compat.so.2 lib
cp /lib/libnss_files.so.2 lib
cp /lib/libresolv.so.2 lib
cp /lib/librt.so.1 lib
cp /lib/libtermcap.so.2 lib

mkdir usr/lib usr/local usr/
➥local/bin usr/local/etc

mkdir www/log www/log/.tmp www/
➥data www/phpsessions www/
➥data/www.fiktivceg.hu
chmod 700 www/phpsessions
➥www/log
chown httpd:httpd www/
➥phpsessions
```

Telepítsük az Apache kiszolgálót

Hogy minél kevesebb állományt kelljen a kalitkába másolni, az *Apache*-ot statikusan fordítjuk le. Töltsük le az *Apache* 1.3.37 verzióját a ➲ *http://httpd.apache.org/download.cgi* címről, a hozzá tartozó *mod_ssl* nevű *SSL* foltot ➲ http://www.modssl.org/source/mod_ssl-2.8.28-1.3.37.tar.gz, webhelyről továbbá szükségünk lesz még a *PHP 4.4.4*-es verziójára a ➲ *http://www.php.net/downloads.php* oldalról. A *PHP* számára *MySQL* támogatást is biztosítunk, a példában feltételezzük, hogy a *mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz* állomány a /usr/local/mysql könyvtár alá lett telepítve.

Első lépésben csomagoljuk ki a letöltött állományokat:

```
tar zxvf mod_ssl-2.8.28-
➥1.3.37.tar.gz
tar zxvf apache_1.3.37.tar.gz
tar jxvf php-4.4.4.tar.bz2
```

Aztán alkalmazzuk az *SSL* foltot (*mod_ssl*):

```
cd mod_ssl-2.8.28-1.3.37
./configure --with-
➥apache=../apache_1.3.37
```

A *2. Listában* látható utasításokkal konfiguráljuk az *Apache*-ot, definiáljuk a *MySQL* telepítésének könyvtárát, hozzáadunk néhány extra könyvtárat (*library*), ill. beállítjuk a statikus fordítást. Most még ne fordítsuk le az *Apache*-ot, hanem konfiguráljuk és fordítsuk le a *PHP* modult, ahogyan az a *3. Listában* szerepel. Természetesen egyéb kapcsolókat is használhatunk, ha extra funkciókra is szükségünk van. Másoljuk át a szükséges állományokat az *Apache* megfelelő könyvtárába:

```
cd ../apache-1.3.37
cp ../php-4.4.4/sapi/apache/
```

2. Lista Az Apache konfigurációja

```
cd ../apache_1.3.37
CFLAGS="-static" \
LDFLAGS="-L/usr/local/mysql/
➥lib" \
LIBS="-lcrypt -lcrypto -lssl
➥-lmysqlclient_r -lz
➥-lresolv -lpthread -lm -lgd
➥-lstdc++" \
INCLUDES="-I/usr/local/mysql/
➥include" \
./configure \
-prefix=/usr/local/
➥apache-1.3.37 \
-enable-module=include \
-disable-module=so \
-disable-module=userdir \
-disable-module=info \
-enable-module=status \
-enable-module=ssl \
-activate-module=src/modules/
➥php4/libphp4.a
```

3. Lista Beállítjuk a PHP-t

```
cd ../php-4.4.4
./configure \
--prefix=/usr/local/php \
--with-zlib \
--with-openssl \
--with-config-file-path=/usr/
➥local/etc \
--with-apache=../
➥apache_1.3.37 \
--with-gd \
--with-mysql=/usr/local/mysql
make
```

```
➥mod_php4.* src/modules/php4
cp ../php-4.4.4/sapi/apache/
➥libphp4.module
src/modules/php4
cp ../php-4.4.4/sapi/apache/
➥apMakefile.libdir src/
➥modules/php4/Makefile.libdir
cp ../php-4.4.4/sapi/apache/
➥apMakefile.tmpl
src/modules/php4/Makefile.tmpl
cp ../php-4.4.4/.libs/libphp4.a
➥src/modules/php4/libmodphp4.a
```

Futtassuk újra az ***Apache*** konfiguráló programját, hogy az ***src/modules/php4*** könyvtárban is létrejöjjön a ***Makefile*** állomány, majd fordítsuk le és telepítsük. A figyelmes olvasónak feltűnhet, hogy a ***/opt/jail*** mint gyökérkönyvtár (/) alá telepítjük az alkalmazást:

```
sh config.status
make
su -c 'make install
➥root=/opt/jail'
```

A titkosítás a barátunk, de nem old meg minden problémát

*„**Ez a webhely biztonságos, mert SSL tanúsítvánnyal rendelkezik**”*. Egy időben több web oldalon is láttam ezt a némileg megtévesztő szöveget, amely azt az érzetet kelti a látogatóban, hogy itt nyugodtan megadhatja az adatait, mert az jó kezekben lesz, ott semmi baj nem történhet. A titkosított kapcsolat jó dolog, főleg ha éppen a személyes adatainkat adjuk meg egy vásárlás során, esetleg a bankszámlánkkal kapcsolatos ügyeket intézzük a fotelból. A valóságban azonban a kriptográfia (például ***SSL***) alkalmazása nem tesz egyetlen kiszolgálót sem biztonságosabbá. A kis lakat a böngészőben nem jelent se többet, se kevesebbet, minthogy a látogató gépén futó böngésző és a távoli web kiszolgáló között egy titkosított csatorna jött létre, és az ebben haladó adatok biztonságban vannak egy esetleges hallgatózó harmadik féllel szemben – hacsak nem az egyik kormány szuperszámítógépe érdeklődik valamelyik tranzakciónk iránt. Az elkészített ***Apache***-unk támogatja az ***SSL*** titkosítást, már csak egy kulcsot és egy tanúsítványt (***certificate***) kell készítenünk. A *4. Listában* az ehhez szükséges parancsok, ill. azok képernyőre írt üzenete látható. (A parancsok előtt a *$* prompt áll, amit nem kell begépelnünk) A ***CSR*** állomány készítésekor ügyeljünk arra, hogy a *„**challenge password**”* kérdésnél ne adjunk meg jelszót, ellenkező esetben az ***Apache*** (minden újra)indításakor be kell azt gépelnünk.

4. Lista SSL kulcs és tanúsítvány készítése

```
$openssl genrsa 2048 > /opt/
➥jail/usr/local/etc/
➥www.fiktivceg.hu.key

Generating RSA private key,
➥2048 bit long modulus
......+++
.............................
.............................
...................+++
e is 65537 (0x10001)


$openssl req -new -key /opt/
➥jail/usr/local/etc/
➥www.fiktivceg.hu.key > 1.csr

Country Name (2 letter code)
➥[AU]:HU
State or Province Name (full
➥name) [Some-State]:Hungary
Locality Name (eg, city)
➥[]:Budapest
Organization Name (eg, company)
➥[Internet Widgits Pty Ltd]:
➥Fiktiv Ceg
Organizational Unit Name (eg,
➥section) []:
Common Name (eg, YOUR name)
➥[]:www.fiktivceg.hu
Email Address
➥[]:info@fiktivceg.hu

Please enter the following
➥'extra' attributes
to be sent with your
➥certificate request
A challenge password []:
An optional company name []:


$openssl x509 -in 1.csr -out
➥/opt/jail/usr/local/etc/
➥www.fiktivceg.hu.cert -req
➥-signkey /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.key
Signature ok
subject=/C=HU/ST=Hungary/
➥L=Budapest/O=Fiktiv Ceg/
➥CN=www.fiktivceg.hu/
➥emailAddress=info@fiktivceg.hu
Getting Private key

$chmod 400 /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.key
$chmod 644 /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.cert
```

A példában mi magunk írjuk alá a tanúsítványt, ami arra ugyan jó, hogy titkosított kommunikációt folytassunk egy belső hálózaton, de mindenképpen ajánlott egy hatósággal (***CA***) – például ***NetLock***, ***VeriSign***, ***Thawte*** – aláíratni a tanúsítványunkat, ha az Interneten akarunk megjelenni, ez segít elkerülni a közbenső ember (***man-in-the-middle***) támadásokat. Javaslom, hogy az ***Apache*** konfigurációs állományát (***httpd.conf***) mozgassuk át a ***/opt/jail/usr/local/etc*** könyvtárba, az indító héjprogramot (***apachectl***) pedig az ***/opt/jail/usr/local/bin*** alá, továbbá készítsünk egy szimbolikus linket az

5. Lista Egy apró korrekció az apachectl állományban

```
PIDFILE=/usr/local/apache/
➥logs/httpsd.pid
HTTPD="/usr/local/apache/bin/
➥httpsd -f
/usr/local/etc/httpsd.conf"
```

```
ln -sf /usr/local/apache-1.3.37
➥/usr/local/apache
```

utasítással. Ezek a kényelmünket szolgálják, verzió frissítésnél elég csak a szimbolikus linket módosítani. Az *5. Listában* látható módon módosítsuk az `apachectl` programban az alábbi két változót, hogy a megfelelő állományokra hivatkozzon.

6. Lista A httpd.conf állomány

```
# globalis rész
#

# httpd felhasználóként fog
# futni az Apache
User httpd
Group httpd

ServerName www.fiktivceg.hu
ServerAdmin info@fiktivceg.hu

# nem kérünk névfeloldást
HostnameLookups Off

LogLevel warn

ErrorLog /www/log/.tmp/
➥error.log

LogFormat "%h %v %l %u %t
➥\"%r\" %s %b %{Referer}i
➥\"%{User-agent}i\"" myformat
CustomLog "|/usr/local/apache/
➥bin/rotatelogs /www/log/.tmp/
➥access.log 10800" myformat

# minimális adatot adunk
# a kiszolgálóról
# ServerSignature Off
ServerTokens Minimal

<IfModule mod_alias.c>

    ....

# szigorú korlátozások
# a CGI-BIN könyvtáron
    <Directory "/www/cgi-bin">
        AllowOverride
        ➥AuthConfig Limit
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>


<IfModule mod_mime.c>
    # engedélyezzük a php
    # motort
    AddType application/
    ➥x-httpd-php .php
</IfModule>


# mod_ssl konfiguráció

SSLCertificateKeyFile /usr/
➥local/etc/www.fiktivceg.hu.key
SSLCertificateFile /usr/local/
➥etc/www.fiktivceg.hu.cert

AddType application/x-x509-
➥ca-cert .crt
AddType application/x-pkcs7-
➥crl    .crl

SSLPassPhraseDialog  builtin

SSLSessionCache        dbm:/
➥www/ssl/ssl_scache
SSLSessionCacheTimeout  300

SSLMutex  file:/www/ssl/
➥ssl_mutex

SSLRandomSeed startup builtin
SSLRandomSeed connect builtin

SSLLog none
SSLLogLevel warn

SSLCipherSuite ALL:!ADH:!
➥EXPORT56:RC4+RSA:+HIGH:
➥+MEDIUM:+LOW:+SSLv2:+EXP:
➥+eNULL
SetEnvIf User-Agent ".*MSIE.*"
➥nokeepalive ssl-unclean
➥-shutdown downgrade-1.0
➥force-response-1.0

<Directory "/www/cgi-bin">
    SSLOptions +StdEnvVars
    ➥+CompatEnvVars
</Directory>


#
# most csak egy virtuális
# kiszolgálót definiálunk

NameVirtualHost 1.2.3.4:80

<VirtualHost 1.2.3.4:80>
    ServerName www.fiktivceg.hu
    SSLEngine off

    DocumentRoot /www/data/
➥www.fiktivceg.hu

    <Directory /www/data/
➥www.fiktivceg.hu>
        Options Includes Indexes
        AllowOverride AuthConfig
        ➥Limit
        Order allow,deny
        Allow from all
    </Directory>

    php_admin_value
    ➥open_basedir
"/www/data/www.fiktivceg.hu/"
    php_admin_value doc_root
    ➥"/www/data/
    ➥www.fiktivceg.hu/"
    php_admin_flag
    ➥display_errors On
    php_admin_value
    ➥sendmail_path
"/usr/sbin/sendmail -t -i -f
➥info@fiktivceg.hu"

    # engedélyezzük a fájl
    ➥feltöltést
    php_admin_flag file_uploads
    ➥On
    php_admin_value
    ➥upload_tmp_dir
/www/data/www.fiktivceg.hu/
➥upload
    php_admin_value
    ➥upload_max_filesize 1000k

    ErrorDocument 404 http://
    ➥www.fiktivceg.hu/
    ➥error404.php

</VirtualHost>

NameVirtualHost 1.2.3.4:443

<VirtualHost 1.2.3.4:443>
    ServerName www.fiktivceg.hu
    SSLEngine on

    ....

</VirtualHost>
```

Az Apache és PHP beállítása

Indítás előtt szerkesszük az *Apache* konfigurációs állományát, a *6. Listában* szereplő változók kivételével a többi maradhat az alapértelmezett értékkel.

A *PHP* egy rendkívül sokoldalú programnyelv, amelynek a használata sajnos azzal a (biztonsági szempontból)

7. Lista Bekapcsoljuk az úgynevezett safe mode funkciót

```
safe_mode = On
safe_mode_gid = On

# nem kell mindenkinek tudni,
# hogy milyen verziójú PHP
# fut a gépünkön
expose_php = Off
```

rendkívül kellemetlen mellékhatással is jár, hogy a felhasználók gyakorlatilag olyan ***PHP*** nyelven megírt programokat futtathatnak, amilyet csak akarnak. Néhány óvintézkedést azonban megtehetünk, hogy az ebből eredő esetleges károkat minimalizáljuk. Másoljuk a ***PHP*** forrás könyvtárából a php.ini-recommended nevű állományt a ketrecbe:

```
cp php.ini-recommended /opt/
➥jail/usr/local/etc/php.ini;
➥chmod 600 /opt/jail/usr/
➥local/etc/php.ini
```

Itt szeretném felhívni a kedves Olvasó figyelmét a ***php.ini*** elején található megjegyzésekre, amelyek a javasolt beállításokat tartalmazzák, közülük többnek van biztonsági vonatkozása. Mindenki kedvére szabhatja testre a ***PHP*** konfigurációs állományát, a ***http.conf*** állományban minden virtuális kiszolgáló (***virtualhost***) esetében egyedi értékeket lehet beállítani.
A ***safe_mode*** – amelyet a *7. Listában* kapcsolunk be – több ***PHP*** függvényre is hatással van, korlátozza azok működését, például a ***fopen()*** függvény nem hajlandó megnyitni azokat az állományokat, amelyek kívül esnek az ***open_dir*** változó által megadott könyvtáron. Ez utóbbi igen hasznos funkció, így elkerülhetjük, hogy illetéktelenek megnézzék például a jelszó állományunkat (***/etc/passwd***).
A ***php.ini***-ben állítsuk be a

```
session.save_path = /www/
➥phpsessions
```

sort, hogy a ***PHP*** itt tárolja el az egyes kapcsolatokhoz ***(session)*** tartozó információkat.

8. Lista A modsecurity modul beállítása a httpd.conf állományban

```
<IfModule mod_security.c>

    # bekapcsoljuk a modult
    SecFilterEngine On

    # probléma esetén 403-as
    # HTTP státusz kódot ad
    # vissza, és elutasítja
    # a kérést
    SecFilterDefaultAction
    ➥"deny,log,status:403"

    # néhány józan
    # alapértelmezett beállítás
    SecFilterScanPOST On
    SecFilterCheckURLEncoding
    ➥On
    SecFilterCheckUnicode
    ➥Encoding Off

    # a 0 byte érték
    # kivételével mindent
    # elfogad
    SecFilterForceByteRange 1
    ➥255

    # álcázza a kiszolgáló
    # típusát és verzióját
    # SecServerSignature
    # "Microsoft-IIS/5.0"

    # az ideiglenes
    # állományokat itt tárolja
    SecUploadDir /www/tmp
    SecUploadKeepFiles Off

    # csak a lényeges adatokat
    # naplózza
    SecAuditEngine RelevantOnly
    SecAuditLog /www/log/.tmp/
    ➥modsec_audit.log

    # nem akarunk túl részletes
    # naplózást
    SecFilterDebugLevel 0
    SecFilterDebugLog /www/log/
    ➥.tmp/modsec_debug.log

    # csak azokat a kéréseket
    # fogadjuk el, amelyeket
    # kezelni is tudunk
    SecFilterSelective
    ➥REQUEST_METHOD
    ➥"!^(GET|HEAD)$" chain
    SecFilterSelective
    ➥HTTP_Content-Type
    ➥"!(^application/x-www
    ➥-form-urlencoded$|
    ➥^multipart/form-data;)"

    # GET és HEAD kéréseket nem
    # fogadunk el törzzsel
    SecFilterSelective
    ➥REQUEST_METHOD
    ➥"^(GET|HEAD)$" chain
    SecFilterSelective
    ➥HTTP_Content-Length "!^$"

    # minden POST kérésnél meg
    # kell adni a hosszát is
    SecFilterSelective
    ➥REQUEST_METHOD "^POST$"
    ➥chain
    SecFilterSelective
    ➥HTTP_Content-Length "^$"

    # csak ismert átviteli
    # kódolást fogadunk el
    SecFilterSelective
    ➥HTTP_Transfer-Encoding
    ➥"!^$"

</IfModule>
```

A `display_errors` direktíva bekapcsolását nem javasolja a ***PHP*** kézikönyve, mert az esetleges hibaüzenetek információt adhatnak a támadónak. Azért javaslom mégis, mert ezek a hibák elcsúfítják a honlapunkat, így rákényszerítik a web fejlesztőket, hogy olyan kódot írjanak, ami nem eredményez hibákat.
A `sendmail_path` változó levelezés esetén hasznos, például űrlap kitöltésének visszaigazolásakor, az ***SMTP*** kapcsolat ***MAIL FROM:*** paraméterét rögzíti fix értékre, így nyomon lehet követni, hogy az egyes leveleket melyik virtuális kiszolgáló küldte el. Az is hasznos lehet, ha letiltjuk a `phpinfo()` függvényt, amely túl sok információt tud, még azt is megmutatja, hogy a ***PHP***-t fordító felhasználónak mi volt a `$PATH` változója. Egy adott függvényt a `disable_functions` paraméternél felsorolva tudunk letiltani, például:

9. Lista Valaki bináris kérést küldött

```
==48835b61==================
==========
Request: www.fiktivceg.hu
64.56.74.94 - - [27/Dec/
➥2006:05:11:49 +0100] "\x04\
➥x01" 403 0 "-" "-" - "-"
--------------------

mod_security-action: 403
mod_security-message: Access
➥denied with code 403.
➥Pattern match
➥"!(^application/x-www
➥-form-urlencoded$|
➥^multipart/form-data;)" at
➥HEADER("Content-Type")
➥[severity "EMERGENCY"]

-48835b61-
```

```
disable_functions = phpinfo,
➥shell_exec, system
```

Egy visszatérő probléma a `register_globals` *PHP* változó használata. Noha már számos verzió óta ki van kapcsolva alapállapotban (nem véletlenül!), mégis számtalan web fejlesztő és alkalmazás követeli, hogy kapcsoljuk be. Ahelyett, hogy követnék a *PHP* újabb lehetőségeit, és hozzáigazítanák a programjaikat.
Itt nem foglalkozunk azzal, hogy több kódoló is létezik, amelyekkel az olvasható *PHP* kódból egy – akár titkosított – futtatható ***bytekódot*** lehet készíteni. Ez amellett, hogy megvédi a *PHP* kódunkat az illetéktelen szemektől, azzal az előnnyel is jár, hogy gyorsabban fog futni a *PHP* programunk. Azonban ez sem csodaszer: a rossz és hibás kód ellen ez sem véd meg.
Ha idáig eljutottunk, adjuk ki ***root*** felhasználóként a

```
chroot /opt/jail apachectl
➥configtest
```

parancsot. Ha minden rendben, akkor indítsuk el a ***chroot /opt/jail apachectl start*** utasítással. Gratulálok, elkészült egy relatíve biztonságos web kiszolgáló!

Egy biztonsági modul

Bár hosszú út áll mögöttünk, mégsem merítettük ki az összes lehetőséget, amivel a web kiszolgálónkat biztonságosabbá tehetjük. Időről időre bejárja az Internetet valamilyen egzotikus nevű féreg (***worm***), amelyet a megfertőzött gépek láncreakciószerűen terjesztenek tovább, hogy más kiszolgálókat is megfertőzzenek.
A ***modsecurity*** nevű ***Apache*** modul ➲ ***http://www.modsecurity.org/*** segítségével (amely elérhető az 1.3.x, 2.0.x és 2.2.x verziókhoz is) azonban bizonyos támadásokat megakadályozhatunk. A ***modsecurity*** modul segítségével bizonyos kérésekre 'hozzáférés megtagadva' (403) választ adhatunk. A modul az időponton és a kliens ***IP***-címén túl naplózza magát a problémás kérés adatait, a ***modsecurity*** válaszát és azt is, hogy az adott kérés melyik szabályon akadt fenn. A javasolt konfigurációs részlet a *8. Listában* látható.
Az ***Apache 2.x*** verziójához a ***modsecurity 2.x*** változata használható, amely jobb reguláris kifejezés támogatással és több előre definiált szabállyal rendelkezik, mint az 1.x.
Egy tipikus naplóbejegyzés a *9. Listában* látható.

És még mindig nincs vége!

A téma összetettsége miatt csak utalok néhány dologra, amelyek nélkül nem képzelhető el biztonságos web (tulajdonképpen semmilyen) kiszolgáló. Az első lépés a fizikai biztonság megteremtése, csak a jogosult személyek férhessenek a gép közelébe. A következő lépés az operációs rendszer telepítése. Fontos, hogy csak azokat a csomagokat telepítsük, amelyekre valóban szükségünk lesz. Például fordító (***gcc, make***) biztosan nem kell, inkább egy másik gépen fordítsuk le az ***Apache*** kiszolgálót, ott készítsünk belőle csomagot, és azt vigyük át a biztonságos kiszolgálónkra.
Nem lehet eleget hangsúlyozni, hogy minden biztonsági frissítést azonnal telepíteni kell. Minden terjesztés esetén jó, ha megerősítjük az operációs rendszert, pl. szigorítjuk némely konfigurációs állományhoz való hozzáférést (például ***/etc/lilo.conf***), töröljük a szükségtelen binárisokról a ***setuid/setgid*** jogosultságot (például ***/bin/mount***), eltávolítjuk a szükségtelen felhasználói fiókokat (például ***news, uucp, games***).
Ne feledjük az arany szabályt: minél kevesebb információt mutatunk meg a gépünkről, annak konfigurációjáról, minél kevesebb dologhoz férhetnek hozzá, annál kisebb az esélye egy betörésnek.
A gépen az összes felhasználói fióknak, és különösen a ***root*** felhasználónak erős jelszóval kell rendelkeznie. Itt nem bonyolódom bele például az egyszer használatos jelszavakba (***OTP***) vagy a biometrikus azonosítókba. Sokat segíthet egy körültekintően kialakított kötelező hozzáférés szabályozó (***MAC***) rendszer is. A fizikai biztonság része még az érzékeny adatokat tartalmazó mentésekhez való hozzáférés szabályozása, jó ha páncélszekrényben tartjuk azokat.
Szintén nem esett szó arról sem, hogy hiába egy biztonságosan kialakított kiszolgáló, ha a web fejlesztők nem kellő körültekintéssel írják meg az alkalmazásaikat. Találkoztam egy olyan kiszolgálóval, amelyiken egy sérülékeny verziójú ***phpBB*** fórum futott. Egy kreatív látogató módosította az alkalmazás konfigurációját tartalmazó ***SQL*** táblát, és egy furcsa dizájnt állított be. Magát a gépet ugyan nem törte fel, de pont elég kárt okozott azzal, hogy megváltoztatta a nyitó oldalt. Sokszor nem is szükséges teljes ellenőrzést szerezni a célpont felett, az pont elég lehet a vállalkozásunk csődjéhez, ha bizalmas üzleti információkat szerez meg a konkurencia. Amihez az is elég lehet, ha le tud másolni egy adatbázist.
Bár nem a legújabb ***LAMP*** verziókat használtam ebben a példában, de ezek az elvek a ***PHP 5.x*** ill. az ***Apache 2.x*** verziókkal is használhatóak. Sok sikert kívánok minden Olvasónak a saját biztonságos kiszolgálójához!

Sütő János
(jsuto@freemail.hu)
1997 óta használ Slackware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spamszűrőjét polírozza.