Webes környezet kialakítása szabad szoftverek segítségével

Szabad szoftver keretrendszer

Készítette a Közigazgatási és Igazságügyi minisztérium E-közigazgatási
Szabad Szoftver Kompetencia Központja
Budapest, 2013

Kódszám: EKOP–1.2.15

A dokumentum legfrissebb változata letölthető a honlapunkról:
http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

Tartalomjegyzék

Webes környezet kialakítása szabad szoftverek segítségével

Egy átlagos webes kiszolgáló a legtöbb esetben[1] egy úgynevezett LAMP[2] környezetből áll, amelynek részei az operációs rendszer, a webszerver, valamilyen szerveroldali szerver szkriptnyelv, mint például a PHP és az adatbázis-kezelő. Egész pontosan így kapjuk meg a LAMP rövidítés betűit. A legtöbb statisztikai mérés szerint a Linux+Apache+MySQL+PHP környezet a leginkább elterjedt (lásd az adatbázis fejezetet[3]). Egy átlagosan feltelepített környezetben tehát a fenti megoldás segítségével kialakíthatunk dinamikus és statikus, szabad szoftverekkel működtetett oldalakat. Ez a fejezet segít a webszerverek és a hozzájuk szorosan kapcsolódó szolgáltatások kiválasztásában. Ilyen például az FTP szerver vagy az SFTP szerver, hiszen a legtöbb esetben az oldalak karbantartására ma is ezt a megoldást használják azok karbantartói, természetesen a számos egyéb lehetőség mellett. Ez a fejezet a webes kiszolgálás köré csoportosítható eszközöket ismerteti.

[1] http://news.netcraft.com/
[2] http://hu.wikipedia.org/wiki/LAMP_(szoftvercsomag)
[3] http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

Webszerverek

Apache

Az Apache[4] hosszú évek óta a legnépszerűbb[5] webszerver a független kutatások szerint. Az Apache projektet az ASF alapítvány keltette életre, és többek között[6] a webszervert is ők fejlesztik, tartják karban. Az alapítvány védőernyője alatt tevékenykedik számtalan szabad szoftveres programozó, akik az 1999-es indulás óta fejlesztik a webszervert. A kezdetekben a cél az volt, hogy az NCSA HTTPd[7] bővítésén dolgozzanak, majd 1999. június 1-jén hivatalosan is létrejött az ASF. Az Apache egy igen kifinomult, moduláris és rugalmasan bővíthető webszerver, amely kompatibilis a HTTP/1.1 (RFC 2616) protokollal. A projekt nyilvánosságra bocsátása után dinamikusan fejlődött, és hamar valós alternatívát kínált az akkor előnyös pozícióban lévő Netscape Communications Corporation webszerverrel szemben. Azóta is folyamatosan fejlődik, és gyakorlatilag uralja a piacot. Moduláris felépítésének köszönhetően mi magunk dönthetünk arról, hogy az alap telepítés után milyen funkciókat akarunk használni, például az SSL vagy a mode_rewrite modult. Megfelelő hardvertámogatás mellett jól skálázható, kis- és nagyvállalati megoldásokra is. A legtöbb szkriptnyelvet támogatja, mint a PHP[8], Perl[9], Python[10], Ruby on Rails[11]. Több kereskedelmi cég felhasználja az Apache-ot termékeiben: Oracle, IBM WebSphere, OSX beépített webszerverként. Számos router, gateway, NAS megoldás szerves része. Teheti mindezt a nagyon is megengedő Apache Szoftver Licenc miatt. Telepítése Ubuntu LTS alatt is igen egyszerű, az:

```
apt-get install apache2
```

parancs kiadásával az alap függőségek kezelése mellett fog települni a webszerver. Gyakorlatilag a parancs lefuttatása után van egy teljesen alap szinten működő web szerverünk, amely létrehozza a /var/www könyvtárban az alap „It works!” oldalt, és innentől kezdve a webszerver alap beállítások szerint a 80-as porton fogadja a kéréseket. Mint oly sok hálózati kiszolgáló szolgáltatást, az Apache-ot sem érdemes alapbeállításokkal használni, hiszen egyrészt nem túl biztonságos, valamint jellemzően több tartományt vagy oldalt szolgál ki az Apache, így mindenféleképpen nézzük meg, milyen beállításokat érdemes átállítanunk. A konfigurációs fájlok az /etc/apache2/ könyvtárban találhatóak.

```
apache2.conf httpd.conf mods-enabled/ sites-enabled/ conf.d/ magic ports.conf ssl/
envvars mods-available/ sites-available/
```

Az apache2.conf fájl

Az ismerkedést kezdjük az apache2.conf fájllal, amelyben a legfontosabb központi paraméterek[12] megtalálhatóak, itt paraméterezhetünk az Apache alapvető viselkedésével kapcsolatban sok mindent, valamint igazából ide van belinkelve a többi konfigurációs fájl is, azaz az Apache innen olvassa be például a mods/site-enabled konfigokat is. Nézzük, mihez érdemes hozzányúlni és miért:

[4] http://www.apache.org/
[5] http://news.netcraft.com/archives/2013/09/05/september-2013-web-server-survey.html
[6] http://projects.apache.org/
[7] http://hu.wikipedia.org/wiki/NCSA_HTTPd
[8] http://www.php.net/
[9] http://www.perl.org/
[10] http://www.python.org/
[11] http://rubyonrails.org/
[12] http://httpd.apache.org/docs/current/mod/core.html

```
Timeout 300
```

Másodpercben kifejezve megmondjuk, hogy az Apache mennyi időt várjon mielőtt Timeout hibával lezárja a kapcsolatot. Főként nagy forgalmú, vagy I/O terhelt szerverek esetében lehet ez érdekes. Alapbeállításon célszerű hagyni, illetve ha szükséges változtatni (általában lefele irányban, azaz némileg csökkenteni) akkor minden esetben tesztelni érdemes, hogy az adott terheltség és az adott kiszolgálási stílus mellett számunkra mi a megfelelő: nem mindegy, hogy egy nagy terhelésű fórum szervert paraméterezünk, vagy egy 200 MB-os letöltőközpontot.

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

Ezen 3 paraméter segítségével állítható be, hogy a szerver egy TCP kapcsolat nyitva tartása mellett több kérést is kiszolgál. Ilyen eset, amikor egy oldalon sok kép van, és az Apache ilyen beállítások mellett csak egy TCP kapcsolatot használ fel arra, hogy esetleg az összes adatot a kliens felé továbbítsa. Az értékeket ezredmásodpercben tudjuk megadni. Megint csak: nagy terhelés esetén az oldal felépítése és a felhasználók szokásai alapján érdemes próbálkozni a ki/be kapcsolásával, és Munin segítségével folyamatosan monitorozni a végeredményt.

```
<IfModule mpm_prefork_module>
  StartServers       5
  MinSpareServers    5
  MaxSpareServers   10
  MaxClients       150
  MaxRequestsPerChild  0
</IfModule>
```

Alap telepítés szerint az mpm_prefork_module-t használjuk, hiszen a libapache2-mod-php5 kiterjesztés ezzel kompatibilis. Amit érdemes tudni róla, hogy ebben az esetben a kéréseket egy szál szolgálja ki, és a memóriaigénye is magasabb a workerhez[13] képest. Beállítása és üzemben tartása átlagos esetben egyszerűbb, és javasolt is a prefork+libapache2-mod-php esetén. Érdemes tudni róla, hogy ha mégis a workert akarnánk választani, akkor lehetőségünk lenne a mod_fcgi-vel PHP-futtatásra. Nézzük a paramétereket:

StartServers: meghatározza, hogy az Apache indulásakor hány szerverfolyamat induljon el. Általában érdemes az alapértelmezett beállításon hagyni, azaz 5-ön.

MinSpareServers, MaxSpareServers: a minimum és maximum SpareServer opciókkal szabályozhatjuk, hogy hány tartalék kiszolgáló fusson a háttérben, amelyek a várakoznak az új csatlakozók kiszolgálására. Így ha a szabad folyamatok száma eléri a MaxSpareServers értéket, akkor az Apache le fogja kapcsolni a feleslegesen futó folyamatokat.

MaxClients: meghatározhatjuk, hogy mennyi klienst szolgáljon ki az Apache egy időben maximum. Tipikusan ha egy kliensnek sokat kell várakozni az oldal betöltéséig, akkor vélhetőleg ez a szám túl alacsony.

MaxRequestsPerChild: ezzel szabályozzuk, hogy egy gyermek folyamat összesen mennyi kérést szolgálhat ki. Alapértéke nulla, de néha – főleg memóriaszivárgás esetén – érdemes ezt az értéket terheléstől függően 5000-10000 közé állítani, de csak teszt jelleggel.

```
AccessFileName .htaccess
<Files ~ "^\.ht">
  Order allow,deny
  Deny from all
  Satisfy all
</Files>
```

[13] http://httpd.apache.org/docs/2.2/mod/worker.html

Rögzítjük, hogy a .htaccess fájlt hogyan ismerje fel a szerver, valamint megmondjuk, hogy a .htaccess és a .htpasswd fájlok kliensek által való lekérhetőségét akadályozzuk meg. A .htpasswd fájlokat amúgy sem praktikus publikus könyvtárba elhelyezni, de erről később még bővebben értekezünk.

```
HostnameLookups Off
```

A direktíva segítségével megmondhatjuk, hogy a kliensek IP-címének rögzítését hogyan akarjuk látni a naplóban. Paraméterei: On|Off|Double. ON esetén az IP-hez kapcsolódó reverse érték is rögzítve lesz. Érdemes azonban figyelembe venni, hogy akár már egy kis látogatottságú weboldalon is rengeteg nslookup-ot eredményezhet, azaz nem véletlen, hogy az alapértelmezett paramétere az OFF.

```
ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Itt határozzuk meg a naplózással kapcsolatos formai és fájl igényeinket. Azaz, hol legyen a központi error.log. Ebbe a fájlba fogja rögzíteni az Apache a működése során fellépő hibákat, ide fognak kerülni (ha másként nem rendelkezünk) a PHP-ból eredő hibák, illetve a nem létező fájlok és egyéb természetű hibák. Ezért célszerű oldalanként definiálni egy error.log fájlt. A többi paraméter segítségével pedig a látogatói kliensekről megszerezhető információk rögzítési formátumát állítjuk be. Mivel alapesetben a felhasználók gépei és böngészői túlságosan közlékenyek (User Agent[14]), ezért ha nincs szükségünk minden információra, itt tudjuk szűkíteni a kívánt információhalmazt.

```
Include ports.conf
```

Itt történik a fejezet elején felsorolt konfigurációs fájlok meghívása, azaz gyakorlatilag ezek is az apache2.conf részét képezik, csak egy strukturált módon.

A ports.conf fájl

A következő konfigurációs fájl tehát a ports.conf, amelyben azt rögzítjük, hogy a webszerver milyen IP-címeken és milyen portokon hallgasson a világ felé:

```
NameVirtualHost *:80
NameVirtualHost *:443
```

Jellemzően egy webszerver nem csak egy tartományt szolgál ki, hanem többet. Ebben az esetben be kell kapcsolnunk a NameVirtualHost direktívát, amely azonban a 2.3.11-es verzió óta szükségtelenné vált, mivel már az Apache úgy értelmezi, hogy bármelyik IP vagy port több tartományt szolgál ki. Itt jelenleg azért szerepel, mivel a jelenleg használatos Ubuntu LTS verzióban még kötelező direktíva.

```
Listen 80
<IfModule mod_ssl.c>
Listen 443
</IfModule>
```

Ezzel megmondjuk a webszervernek, hogy a standard 80-as és 443-as portokon működjön. A 80-as port természetesen az alap, a 443-ashoz viszont SSL tanúsítványt kell készíteni és be is kell konfigurálni az Apache számára, amelyet később teszünk meg, ezért van feltételek között. Azaz ha az

[14] http://hu.wikipedia.org/wiki/User_agent

SSL modult bekapcsoltuk, akkor a Listen paraméter is élni fog, ha nem akkor természetesen nem. Megadható még így is:

```
Listen 192.168.1.1:80
```

A security fájl

A következő fájl, amelyet az Apache az /etc/apache2/conf.d-ből olvas fel a security. Nevéből adódóan a biztonsági beállítások egy jó részét tartalmazza:

```
ServerTokens Prod (Full | OS | Minimal | Minor | Major | Prod)
```

Meghatározza, hogy a webszerver milyen fejlécet adjon vissza a kliensek felé. Értelemszerűen itt is igaz az, hogy ha nem tesztelés jelleggel szükséges a teljes információ visszaadása, akkor a lehető legkevesebbet kell magunkról elárulnunk, azaz érdemes Prod-ra állítani az értéket, amely így csak a Product értéket fogja elárulni, azaz Apache lesz.

Nézzük meg mi történik, ha Full értéken van:

```
telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD // HTTP/1.0 \n\n

HTTP/1.1 401 Authorization Required
Date: Mon, 13 May 2013 11:01:07 GMT
Server: Apache/2.2.22 (Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1
WWW-Authenticate: Basic realm="ByPassword"
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
ServerSignature Off
```

Bekapcsolva (jellemzően hiba esetén van így) verzió- és egyén-szenzitív információt (signature) szolgáltat, természetesen nem hibakereső üzemmódban ezt is érdemes kikapcsolva tartani, főként biztonsági megfontolásból.

```
TraceEnable Off
```

Főként teszt és diagnosztikai esetekben érdemes bekapcsolva tartani.

A charset fájl

Ugyancsak a conf.d/ alatt található a charset fájl, amely ma már igen egyszerű, hiszen jellemzően UTF-8 kódolás támogatásával dolgozunk:

```
AddDefaultCharset UTF-8
```

Itt azonban szükség esetén megadható egyéb kódolás külön támogatása is.

Az Apache kiterjesztése

A következő fontosabb lépés az Apache kiterjesztési lehetőségeinek megismerése, azaz a hozzá kapcsolható modulszerkezet, illetve azok ki-be kapcsolása. Ha például PHP-t telepítünk a rendszerre – mint ahogy azt a fejezet vége felé a PHP részletezésénél fogunk is – az Ubuntu telepítőrendszere, miután feltelepítette a szükséges függőségeket, köztük az apache2 modult is, az

/etc/apache2/mods-available alá fogja elhelyezni a PHP modulfájl helyét és a betöltéshez szükséges konfigurációt tartalmazó fájlt, amely így fog kinézni:

```
cat /etc/apache2/mods-available/php5.load
LoadModule php5_module /usr/lib/apache2/modules/libphp5.so
cat /etc/apache2/mods-available/php5.conf
<IfModule mod_php5.c>
  <FilesMatch "\.ph(p3?|tml)$">
    SetHandler application/x-httpd-php
  </FilesMatch>
  <FilesMatch "\.phps$">
    SetHandler application/x-httpd-php-source
  </FilesMatch>
  # To re-enable php in user directories comment the following lines
  # (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
  # prevents .htaccess files from disabling it.
  <IfModule mod_userdir.c>
    <Directory /home/*/public_html>
      php_admin_value engine Off
    </Directory>
  </IfModule>
</IfModule>
```

A PHP-vel nincs teendőnk, hiszen azt majd a telepítőrendszer elintézi, azonban az /etc/apache2/mods-available könyvtárban van sok olyan modul, amely telepítve van, működésre kész, de jelenleg nem használja a webszerver. Az egyik ilyen például az SSL, amelyet egyszerűen az Apache számára használhatóvá tehetünk, csak létre kell hoznunk egy szimbolikus linket az /etc/apache2/mods-enabled/ssl.load -> /etc/apache2/mods-available/ssl.load között, azaz a mods-enabled alá linkeljük a modes-available/ alatt lévő ssl.log és ssl.conf fájlokat, majd az apache2ctl configtest és az apache2ctl restart parancsok segítségével érvényesítjük. A ports.conf-ban rögzítettek szerint így már fel fogja ismerni az Apache, hogy az SSL modul be van töltve. Az SSL további konfigurációját pedig a normál Virtualhost konfigurálás után részletezzük.

Egy tartomány kiszolgálásának beállítása

Nézzük, hogyan tudunk bekonfigurálni egy www.iskola.hu és iskola.hu tartományt. Első lépésben szükséges, hogy a DNS szerverünkben beállítsuk (vagy kérjük annak karbantartójától, hogy a www.iskola.hu a szerverünk IP-címére mutasson – „A” rekord), valamint célszerű az Origin (@)-t, azaz az iskola.hu (www nélküli) címet ugyanerre az IP-re irányítani. Ha ezzel megvagyunk, akkor az /etc/apache2/sites-available könyvtárba helyezzük el a soron következő virtualhost leírófájlját. Ha ez lesz az első, akkor célszerű (a nullás foglalt, az az alapértelmezett) a 001-iskola.hu nevet adni neki. Az egyszerűség kedvéért, mivel itt egyszerre értelmezhető a konfiguráció, minden sor után #-tel kezdve olvasható a magyarázat, értelemszerűen ezeket nem kell a végleges fájlba másolni:

```
<VirtualHost *:80>
#Itt határozzuk meg, hogy milyen IP-címekre és portokra vonatkozzon ez a virtualhost. Mi most a csillag karaktert használtuk, hogy ne okozzon gondot, ha a gépnek több IP-címe van, vagy le kell cserélni. Megadhatunk IP-címet is, például: <VirtualHost 192.168.1.1:80>

ServerAdmin webmaster@iskola.hu
#Meghatározzuk a virtualhost által hirdetett rendszergazdai címet
```

```
ServerName www.iskola.hu
ServerAlias iskola.hu
#A ServerName direktíva mondja meg, hogy milyen DNS névre hallgat majd, a ServerAlias-hoz pedig felvehetjük a szükséges többit, jelen esetben a www nélküli címet, de lehetne itt a mail.iskola.hu is.
DocumentRoot /var/www/iskola/
#A DocumentRoot direktívával meghatározzuk, hogy honnan kezdődik a virtualhost nyilvános könyvtárszerkezete. Ide rakjuk az index.html, index.php stb. fájljainkat.

<Directory />
Options FollowSymLinks Indexes
AllowOverride AuthConfig All
</Directory>
#Az adott könyvtárra meghatározzuk az Options[15] segítségével, hogy ha nincs index.html vagy index.php fájl az adott könyvtárban, akkor kilistázza-e a könyvtár fájljait (Indexes opció), illetve beállítjuk, hogy kövesse a szimbolikus linkeket. Majd az AllowOverride opcióval lehetővé tesszük, hogy ha .htaccess fájl található egy könyvtárban, akkor az abban lévő paramétereket figyelembe vegye, mintha csak például az Options-ök között szerepelt volna. Ez a paraméter barátunk és ellenségünk is lehet. Fontos, hogy akkor engedjük All opciókkal, ha erre a PHP programozóknak egyedileg szükségük van és megbízunk bennük, minden más esetben érdemes virtualhostonként és könyvtáranként egyénileg szabályozni, engedni vagy tiltani bizonyos opciókat. Az AuthConfig opcióval pedig a .htaccess-ből megvalósított felhasználó/jelszó bekérését engedélyezzük.
<Directory /var/www/>
Options FollowSymLinks MultiViews
AllowOverride Authconfig
Order allow,deny
allow from all
</Directory>
#Mint látható a /var/www könyvtárra, amely a DocumentRoot alatt helyezkedik el, részben eltérő hozzáférési beállításokat eszközöltünk. Tiltottuk az Indexes beállítást, azaz nem engedjük a listázást, illetve hiányzik az All paraméter, azaz ott már nem dönthet a .htaccess-ben utólag beállított felülíró paraméter. Az egészre akkor lehet szükség például, ha a weboldalunknak szüksége lehet a DocumentRoot-on kívülről írni vagy olvasni, ilyen eset amikor 2 oldal egy szerveren részben közös PHP kódot használ, vagy csak biztonsági okokból elkülönítünk bizonyos dolgokat a DocumentRoot-tól szeparálva.

ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/iskola.hu-access.log combined
#A korábban már tárgyalt naplózási lehetőségeket rögzítjük, azaz hol legyen a site error.log fájlja, milyen részletesen legyen a napló, illetve a sima hozzáférési napló hol tárolódjon.

</VirtualHost>
#itt ér véget a virtualhostunk.
```

Ha elmentettük, akkor következő lépésként be kell linkelnünk most létrehozott fájlt a sites-enabled könyvtárba, hogy az Apache tudja, most már használnia kell:

```
ln -s /etc/apache2/sites-available/001-iskola.hu /etc/apache2/sites-enabled/001-iskola.hu
```

[15] http://httpd.apache.org/docs/current/mod/core.html#options

Majd futtassuk le a tesztet, hogy nem írtunk-e el valamit, vagy véletlenül nem hagytunk ki esetleg egy lezáró </Virtualhost> címkét:
sudo apache2ctl configtest

Ha a válasz a következő:

Syntax OK

Akkor minden rendben van, ebben az esetben kiadhatjuk a sudo apache2ctl restart parancsot. Ha az is hiba nélkül futott le, akkor az iskola.hu és a www.iskola.hu tartományokat is ki fogja szolgálni az így beállított Apache.

Egy biztonságos tartomány beállítása

Nézzük, mi a teendő, ha HTTPS alatt szeretnénk az ugyancsak a példában szereplő www.iskola.hu és iskola.hu oldalakat konfigurálni, értelemszerűen itt most csak az előző normál 80-as porton működő Apache-hoz viszonyított különbséget fogjuk taglalni. Azt tehát már beállítottuk korábban, hogy a mods-enabled könyvtárba be legyen linkelve az SSL config és load fájl is, valamint az Apache port szinten is tudja, hogy ha a modul be van töltve, akkor a 443-as porton szükséges figyelnie, azért másoljuk az iskola.hu konfigurációs fájlját egy új fájlba, amelyet hívhatunk /etc/apache2/sites-available/002-iskola.huSSL-nek. A következő változtatásokat eszközöljük:

```
A Virtualhost direktíva értelemszerűen ne 80-as hanem 443-as legyen:
<VirtualHost *:443>
```

A szerver név és alias meghatározás után, de még a DocumentRoot előtt a következő sorokat helyezzük el:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/iskoal.hu.crt
SSLCertificateKeyFile /etc/apache2/ssl/iskola.hu.private.key
```

Értelemszerűen előtte az SSL fejezetben (a fejezet jelenleg készítés alatt van, amint elkészülünk a weboldalunkon megtalálható lesz) leírtak szerint készítsük el a kulcspárokat. Ezek után ne felejtsük belinkelni a sima iskola.hu-hoz hasonlóan az iskola.huSSL fájlt ugyanúgy a sites-enabled könyvtárba, majd az

```
apache2ctl configtest
```

parancs futtatásával győződjünk meg róla, hogy semmit nem írtunk el, és csak ezek után indítsuk újra az Apache-ot, amelyet ezúttal az SSL konfiguráció bővülése miatt egy teljes értékű restart kiadásával érdemes megejteni:

```
service apache2 restart
```

Ha mindent jól csináltunk, és az error.log fájlban sincs jelezve semmi hiba, akkor a 443-as porton a https://www.iskola.hu oldalon meg fog jelenni pontosan ugyanaz a tartalom, mint a sima iskola.hu oldalon. Mivel a HTTP és a HTTPS virtualhost konfigurációja nem egy fájlba lett belerakva, ezért figyeljünk oda arra, hogy később amikor változtatni kell a HTTP konfiguráción, akár egy PHP-beállítást, akár egy log vagy Directory bejegyzést, akkor ezt a HTTPS fájlba is vezessük át.

Lighttpd

A Lighttpd[16] egy kicsi és rendkívül gyors alternatív webszerver motor. A fejlesztését 2003-ban teljesen az alap koncepció lefektetésével kezdték, a C10K[17] probléma köré építve. Fő jellemzője, hogy kifejezetten biztonságos és rendkívül gyors, az erőforrásokat teljes mértékben kíméli, így például egy statikus tartalom kiszolgálásához jóval kevesebb RAM- és processzorkapacitást igényel. A

[16] http://www.lighttpd.net/
[17] http://en.wikipedia.org/wiki/C10k_problem

mostanában igen elterjedt bérelhető VPS[18] megoldásokban gyakran használják, a szűkös erőforrások optimális kihasználása érdekében. Támogatja a FastCGI, CGI, Auth, Output-Compression, URL-Rewriting, SSL opciókat és még számos más megoldást is. Sok esetben használják kirakat webszervernek, azaz felépítenek egy LAMP környezetet, amely csak a dinamikus tartalmat szolgálja és a statikus tartalmat egyfajta terheléscsökkentő pajzsként ugyanezen a gépen a Lighttpd veszi át. De ugyanezt a megoldást használják fordítva is, vagy csak képek kiszolgálására. Jelenleg olyan oldalak használják kiszolgálásra mint a YouTube, Wikipedia vagy a meebo.

Nginx

A Nginx[19] elveiben nagyon hasonló a Lighttpd-hez. 2004-ben Igor Sziszojev, egy orosz rendszer- és szoftvermérnök publikálta az első verziót belőle. Elképesztő terhelhetőség jellemzi, pontosan ugyanazon szempontok mentén, mint a Lighttpd-t. Felépítése moduláris, és elképesztően sok funkciót támogat[20]. Felhasználása nagyon hasonló a Lighttpd-éhez, főként cache proxy és terheléselosztó funkciókat valósítanak meg vele. Gyakori felhasználás, hogy a statikus képeket kiszervezik egy külön Nginx szerver alá, ezzel sokszorozva meg a hardver lehetőségeit. A szakmai fórumok tele vannak ajánlásokkal és gyakorlati tapasztalatokkal, hogy milyen feladatra melyik szervert érdemes és lehet használni. Mindenképpen érdemes átolvasni[21] ezeket egy bonyolultabb rendszer esetén, mert nem egyértelmű, hogy hosszú távon melyik a jobb. Jelenleg a Nginx a világ második legtöbbet futtatott webszervere (a saját dokumentációja szerint).

Webszerver üzemeltetéséhez szükséges komponensek

PHP

A PHP[22] (Personal Home Page Tools) egy szerver oldali szkriptnyelv, amely egyike az első olyan szkriptnyelveknek, amelyet HTML kódba lehet ágyazni külső fájl használata helyett. A webszerverhez kapcsolódó modulként egy PHP-feldolgozó értelmezi a kódot. Az évek során akkora felhasználói és fejlesztői bázisra talált, hogy Rasmus Lerdorf az alkotója a The PHP Group[23]-ra bízta a PHP-t. Mára a legelterjedtebb szkriptnyelv lett, amely messze a legtöbb lehetőséget támogatja modulrendszerével, mint például adatbázis-kezelés számtalan változata, képmanipuláció, cache lehetőségek, távoli szerver kapcsolatok stb. A kezdetekben csak egy szkript gyűjtemény volt, de mára iparági standard lett, amelyet számtalan területen használnak fel. A kezdeti „Personal Home Page Tools” megnevezést megváltoztatták, és ma már a PHP: Hypertext Preprocessor elnevezést használják. Lehetőség van nem csak webszerveren keresztüli használatára, hanem kiszolgáló oldali parancssori munkára is, ezzel nagyrészt átvéve az előtte egyeduralkodó shell és Perl szkriptes megoldásokat, hiszen ugyan úgy crontab-ból[24] is ütemezhető a PHP. A PHP biztonsága híresen kétes, pedig főként nem a PHP-t fejlesztők által elkövetett tervezési hibák jellemzőek, hanem inkább a PHP-t használó fejlesztők nem tartják be a megfelelő direktívákat, és alapvető biztonsági szabályokat (bevitelkezelés stb.). Sokáig a PHP beépített lehetősége volt a safe_mode és a köré csoportosuló egyéb függvé-

[18] http://en.wikipedia.org/wiki/Virtual_private_server
[19] http://nginx.org/
[20] http://nginx.org/en/docs/
[21] http://www.wikivs.com/wiki/Lighttpd_vs_nginx
[22] http://www.php.net/
[23] http://hu1.php.net/manual/en/history.php.php
[24] http://en.wikipedia.org/wiki/Cron

nyek. Sajnos vagy szerencsére az 5.3-as verzió kiadásával ezek kivezetése fokozatosan megkezdődött, így az üzemeltetőknek és a programozóknak kell más megoldások után nézni. A safe_mode megoldás egyébként egy igen kötött programozói viselkedést követelt meg, viszont nagyobb látszólagos biztonságot nyújtott, mint amilyen az a valóságban volt.

A PHP telepítése

Telepítése igen egyszerű, ugyanakkor sokan nem fordítanak elég figyelmet a telepítés utáni részletes beállítására, az Apache-csal való összehangolásra. Ugyanis alapbeállításokkal a PHP igen megengedő üzemmódban fut, amelyen érdemes szigorítani. Az alapszintű telepítése tehát Ubuntu LTS rendszeren:

```
apt-get install libapache2-mod-php5 php5
```

A php5 ezzel automatikusan települ, és a telepítőrendszer be is linkeli a PHP moduljait az Apache megfelelő könyvtárába, majd újra is indítja az Apache-ot, azaz a telepítés után a PHP rendszer működőképes. Ekkor kell elkezdenünk a finomhangolást:

A PHP Apache-hoz kapcsolódó beállításait az /etc/php5/apache2/php.ini fájlban találhatjuk, a fontosabb beállítások, amelyeket érdemes módosítani a telepítés után:

```
engine = On
```

Ezzel ki-be kapcsolhatjuk globális szinten a PHP-t. Ezt megtehetjük virtualhostonként is az Apache-ból, úgy hogy abba a virtualhostba, amelyben le szeretnénk tiltani a PHP-t, beírjuk a következőt: php_flag engine off

Ennek főként akkor van jelentősége, ha egy webhelyet átmenetileg karbantartó üzemmódba teszünk, például egy nagyobb átállás időtartamára azt akarjuk, hogy a látogatók egy index.html-ből tudomást szerezzenek róla, hogy az átállás meddig fog tartani, de ne érjék el a könyvjelzőzött direkt linkeken a PHP tartalmat.

```
safe_mode =Off
```

A Safe Mode egy remek próbálkozás arra, hogy rákényszerítse a programozókat a játékszabályok betartására. Sajnálatosan azonban a PHP 5.2-es változatától már nem támogatott, de még részben használható. Természetesen a Safe Mode nem kínált jó megoldást a biztonsági hibás kódokra és részben olyan érzetet kellett az üzemeltetőkben, hogy a PHP-n keresztül az esetleges támadók nem tudnak majd kitörni a DocumentRoot-ból. Azaz sok esetben illúzió volt, de sok esetben egy fajta alternatíva volt arra az esetre, ha egy webszerveren eltérő tulajdonosú, vagy fejlesztőjű oldalak futottak. Ilyenkor egy bizonyos szeparációs biztonságot jelentett az üzemeltetőnek és igen nagy kötöttséget a fejlesztőknek. Mivel használata a továbbiakban nem tanácsos, itt eltekintünk a részletezésétől[25].

```
disable_functions = pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority
```

A disable_functions egy remek változó arra az eshetőségre, ha szeretnénk tiltani számos programozók által kedvelt, de általunk nem szükségesnek vélt (vagy éppen biztonsági megfontolásból) funkciót. Erre remek példa lehet a phpinfo[26] függvény, amely sok esetben a fejlesztéshez elengedhetetlen, de véletlen hanyagságból kint felejtett phpinfo kimenete gyakorlatilag a szerver összes lényeges és támadható paraméterét kiszolgáltatja, azaz érdemes a fejlesztési szakaszt követően tiltani. A disable_classes is hasonló opció.

```
expose_php = Off
```

[25] http://php.net/manual/en/features.safe-mode.php
[26] http://php.net/manual/en/function.phpinfo.php

Nagyon hasonló az Apache-nál már alkalmazott beállításhoz, azaz a HTTP Header-ben megjelenő tényleges PHP fő- és alverziót lehet vele elrejteni. Alapvetően érdemes Off értéket megadni.

max_execution_time = 30

Meghatározhatjuk, hogy minden egyes PHP folyamat maximum mennyi időt futhat, mielőtt az elemző kilövi. Sok esetben a 30 másodperc kevés szokott lenni, ezért érdemes a saját igényeinkhez igazítani, de figyelve arra, hogy maximum annyit adjunk, amennyi ténylegesen szükséges.

max_input_time = 60

Az előző opcióhoz hasonlóan másodpercben határozzuk meg, hogy az INPUT meddig tarthat.

memory_limit = 128M

Ezzel meghatározzuk, hogy a PHP folyamatok mennyi tényleges memóriát fogyaszthatnak a rendszerünkön. Sok esetben a 128MB kevés, szintén kis lépcsőkben emeljük, és arra a tényleges maximumra ami még éppen elég, de a rendszert sem lehet kikészíteni még vele. A jó beállításhoz a Munin grafikonjait és a top, htop parancsokat is segítségül hívhatjuk.

error_reporting = E_ALL & ~E_NOTICE

Ezzel meghatározhatjuk, hogy a hibanaplózás milyen szinteken legyen bekapcsolva. Fejlesztési időszakban az alap beállítást érdemes meghagyni, később pedig szigorítani[27] kell rajta, mivel túl sok információt árulhatunk el így egy esetleges támadó számára.

display_errors = Off

Ezzel engedhetjük vagy tilthatjuk, hogy a felhasználó a PHP szkript által okozott hibát megjelenítheti-e. Megint csak fejlesztési szakaszban, amikor az oldal nem publikus (például jelszóval védett), akkor érdemes bekapcsolva tartani, de később kikapcsolni és inkább naplófájlba terelni az ilyen üzeneteket.

log_errors = On

Ennek használatával megmondhatjuk, hogy az error.log-ban szeretnénk inkább látni a PHP szkriptek hibaüzeneteit.

file_uploads = On

upload_max_filesize = 20M

Ezekkel a beállításokkal engedhetjük vagy tilthatjuk a fájlfeltöltéseket PHP-n keresztül, valamint meghatározhatjuk azok maximális méretét. Érdemes összehangolni a max_execution_time és a max_input_time = 60 beállítással, mivel ha engedünk több száz MB adatfeltöltést, akkor az vélhetőleg nem fog beleférni az alap beállításoknál meghatározott 30/60-as keretbe, figyelembe véve az itthoni aszimmetrikus internetsebességeket (azaz a feltöltés a legtöbb esetben lassabb mint a letöltési irány).

allow_url_fopen = Off

A PHP szkript számára tilthatjuk vagy engedhetjük például a HTTP vagy FTP távoli fopen hívást. Ha nincs rá szükségünk (a programozók nem igénylik), akkor érdemes Off értékre állítani.

A php.ini fájlban, illetve az Apache virtualhostonként állítható változói még számos lehetőséget kínálnak számunkra, amelyeket most nem részletezünk[28].

Phpmyadmin

A Phpmyadmin[29] egy nyílt forrású, PHP nyelven írt eszköz, amely képes webes felületen a MySQL adatbázis-kezelő majdnem teljes körű menedzselésére. Használatával lehetséges adatbázist kezelni és törölni, illetve a különböző egyéb adatbázissal kapcsolatos műveletek nagy részét is elérhetjük ezen a felületen. Híres a biztonsági incidenseiről, ezért kifejezetten fontos, hogy ne alapbeállításokkal használjuk. A szerverem-neve.hu/phpmyadmin alap hivatkozást ne hagyjuk meg neki (na-

[27] http://php.net/manual/en/function.error-reporting.php
[28] http://hu1.php.net/manual/en/
[29] http://www.phpmyadmin.net/home_page/index.php

gyon könnyen megváltoztatható, hiszen a legtöbb esetben a telepítés a disztribúció csomagkezelőjével zajlik. Így ebben az esetben csak egy Apache Alias beállítás módosítása szükséges. Ha pedig letöltjük, akkor ne olyan könyvtárba rakjuk be, ami a neve.), illetve érdemes eleve HTTPS felület alatt engedélyezni a használatát csak, valamint egy .htaccess fájlal[30] pluszban védeni. A legtöbb esetben használata szükségszerű, hiszen a távoli MySQL port engedélyezése a programozó számára, ha lehet még kevésbé ajánlatos, mint a phpmyadmin maga.

OpenSSL

Az OpenSSL[31] egy nyílt forrású programcsomag- és könyvtár-gyűjtemény, amely segítségével nagyon sokfajta kriptográfiai műveletet végezhetünk el. A legtöbb Linux disztribúció alaptelepítésének része, valamint Windows alatt is elérhető. Az Apache számára tudunk a segítségével egy önaláírt kulcsot készíteni, amelyet a mode_ssl bekapcsolása után megfelelően befűzve lehallgatással szemben ellenállóbbá tehetjük a hálózati kapcsolatot. Azaz a gyakorlatban egy HTTPS felületet hozhatunk létre, illetve ugyanezt az elkészített kulcsot akár az FTP kapcsolat biztonságosabbá tételére is felhasználhatjuk később.

FTP megoldások

A weboldalak adminisztrálására számtalan megoldás létezik, egyszerűbb esetben a tartalomkezelő felület kínál erre lehetőséget valamilyen bővítmény segítségével, vagy eleve elégséges az a funkcionalitás (cikkek és képek közzététele stb.) amely adott, plusz a phpmyadmin. Ha ennél többet szeretnénk, például hozzáférni a PHP és egyéb fájlokhoz közvetlen módon, akkor kerülnek képbe a különböző hozzáférést biztosító szolgáltatások. A legtöbb esetben még mindig az FTP protokollt[32] használjuk erre, de sokan az OpenSSH beépített megoldását az SFTP-t, vagy az SCP-t alkalmazzák. Elterjedt megoldás volt régebben az scp-only shell használata is, de szerencsére ezt az SFTP már jobbára kiszorította. Manapság még inkább fontosabb alapkövetelmény, hogy akármilyen megoldást is használunk, a lehetőség a szeparációra meglegyen. Azaz mindenki csak a saját könyvtárát láthassa, amikor becsatlakozik a kliensével, illetve mi adhassunk meg kivételeket. Ugyanígy alapkövetelmény ma már az is, hogy ha FTP-t használunk, akkor képes legyen SSL vagy TLS alatt működni, és ne csak az azonosítás menjen az SSL réteg alatt, hanem az adatkapcsolat többi része is. Ezért most olyan klienseket fogunk bemutatni, ahol ezek a dolgok megvalósíthatóak.

Pure-FTPd[33]

Ahogyan a weboldal első kiemelt része is említi „Security first”, azaz a megalkotóknak a legfontosabb a biztonság volt. Támogatja a chroot-ot[34], a virtuális szervereket, az SSL/TLS kapcsolatot vegyes és kizárt üzemben is. Azaz beállítható, hogy csak az azonosítás vagy az azonosítás és az adat is titkosított réteg alatt menjen. Könnyen konfigurálható és virtuális felhasználói adatbázisával együtt egészen nagy rendszerek kialakítására is kényelmesen használható. Segítségével akár a www-data felhasználó nevében felvett virtuális felhasználókat tudunk létrehozni, így biztosítva, hogy a safe mode-ban futó PHP szkriptek se álljanak le bizonyos esetekben a futtató környezet (Apache esetén a www-data) és a fájljogosultságok eltérése miatt (safe mode gid). Szabályozható vele az anony-

[30] http://en.wikipedia.org/wiki/Htaccess
[31] http://www.openssl.org/
[32] http://hu.wikipedia.org/wiki/File_Transfer_Protocol
[33] http://www.pureftpd.org/project/pure-ftpd
[34] http://hu.wikipedia.org/wiki/Chroot

mous és a tényleges felhasználók sávszélessége is, ezzel biztosítva, hogy egy-egy nagyobb feltöltési sávszélességgel rendelkező felhasználó ne tudja a végletekig leterhelni a szervert, valamint megmondható az is, hogy a Pure-FTPd mekkora terhelés[35] mellett szolgáljon még ki. Támogatja az FXP (server-to-server) protokollt, amely segítségével 2 szerver között tudunk nagyobb mennyiségű adatot úgy mozgatni, hogy az otthoni kis sávszélességű vonalunkon nem megy át az adat, ezzel is nagyban gyorsítva az adatátvitelt. Nagyon hasznos funkció továbbá, hogy a Passive Port Range opcióval meghatározható azon portok tartománya, ahol a passzív mód esetén kiszolgál. Így rögzíthetjük ezen portokat egy egyszerű Iptables szabály segítségével, ezzel is megkönnyítve a másik oldalról jövő NAT mögüli kapcsolatok dolgát.

Telepíteni a már megszokott apt-get install pure-ftpd paranccsal lehet, ez telepíteni fogja a pure-ftpd-common csomagot is mint függőséget. A telepítés után a konfigurációs fájlokat az /etc/pure-ftpd/conf könyvtárban találjuk. A Pure-FTPd-t praktikusan ebbe a conf könyvtárba elhelyezett szöveges fájlok segítségével (és a fájlokba beleírt opció állapotával[36]) tudjuk finomhangolni, ez a gyakorlatban így néz ki (a conf könyvtár tartalma):

```
/etc/pure-ftpd/conf# ls -f
..              AnonymousCantUpload       MaxLoad
MaxClientsNumber   PureDB                 FSCharset
Daemonize          BrokenClientsCompatibility
MaxIdleTime        DisplayDotFiles         MaxClientsPerIP
AllowAnonymousFXP  AnonymousCanCreateDirs    AnonymousBandwidth
PAMAuthentication  AltLog                  UserBandwidth
TLS                NoAnonymous             AllowUserFXP
UnixAuthentication ChrootEveryone          MinUID
```

Nézzük egyenként az opciók jelentését és a fájlok tartalmát:

NoAnonymous [yes/no]

Ezzel engedhetjük vagy tilthatjuk az anonymous (névtelen) felhasználó bejelentkezését.

AnonymousCantUpload [yes/no]

Ezzel engedhetjük vagy tilthatjuk az anonymous felhasználónak a fájlfeltöltést. Ha megtiltjuk, akkor csak letöltheti az anonymous módban elérhető fájlokat és könyvtárakat. Alapesetben állítsuk no-ra a fájl tartalmát.

AnonymousCanCreateDirs [yes/no]

Ezzel engedhetjük vagy tilthatjuk szintén a vendég felhasználó számára a könyvtárak létrehozását, alapesetben állítsuk no-ra.

AnonymousBandwidth [KB/Sec]

Ezzel beállíthatjuk, hogy a vendég felhasználó mekkora sávszélességgel forgalmazhat. Ha például egy népszerű szabad szoftvert teszünk ki anonymous FTP-re, akkor mindenképpen érdemes ezt az értéket egy olyan tapasztalati úton kipróbált maximumra hangolni, amely még nem veszélyezteti a lemez I/O műveletekből fakadóan a gép stabil üzemét, még akkor sem ha nagyon nagy az érdeklődés az adott fájl iránt.

AllowAnonymousFXP [yes/no]

Ezzel engedhetjük vagy tilthatjuk, hogy a vendég felhasználó az FXP[37] protokoll szerint, a kliens kihagyásával 2 szerver között mozgathasson direktben adatot.

AllowUserFXP [yes/no]

Ez az előző opció normál, azaz beazonosított felhasználókra szánt változata.

AltLog [clf:/var/log/pure-ftpd/transfer.log]

[35] http://hu.wikipedia.org/wiki/Load_(sz%C3%A1m%C3%ADt%C3%A1stechnika)
[36] http://download.pureftpd.org/pure-ftpd/doc/README
[37] http://en.wikipedia.org/wiki/File_eXchange_Protocol

Ezzel beállíthatjuk az átviteli napló pontos helyét. Alapesetben a /var/log/pure-ftpd/transfer.log fájlban érdemes tárolni a felhasználók aktivitását.

ChrootEveryone [yes/no]

Ezen opció segítségével beállíthatjuk, hogy minden felhasználó csak a számára létrehozott home környezetet láthassa, gyakorlatilag / (gyökér) környezetként. Ez különösen hasznos, ha több esetleg „idegen” felhasználót vagyunk kénytelenek beengedni a szerverre. A szerver biztonsága érdekében ez alapbeállítás kellene, hogy legyen, ezért állítsuk Yes-re.

Daemonize [yes/no]

Ezzel adhatjuk meg, hogy a Pure-FTPd külön démonként vagy az inetd[38] részeként fusson. Javasolt démon módban futtatni.

DisplayDotFiles [yes/no]

Mutassa-e a ponttal kezdődő nevű, rejtett fájlokat, mint például a .htaccess-t. Javasolt ezt kikapcsolni, azaz no-ra állítani, és csak akkor engedni a klienseknek, hogy lássák például a .htaccess-t, ha erre ténylegesen szükségük van.

FSCharset [UTF-8]

Ezzel beállítjuk az alapértelmezett karakterkódolást, amely ma már praktikusan az UTF-8.

MaxClientsNumber [szám]

Ezzel meghatározhatjuk, maximum mennyi felhasználót enged be egy időben a rendszer. Nagy terheltségű rendszer esetén érdemes alulról felfele haladva tesztelve beállítani, hogy a túlterheltség ne akadályozza a szerver működését.

MaxClientsPerIP [szám]

Ezzel meghatározhatjuk, hány klienst engedjen egy adott IP-címről. Tekintve, hogy manapság már az otthoni hálózatok is NAT mögött vannak, távolról 1 IP-nek látszanak, ami azt jelenti, hogy akár egy egész alhálózat lehet 1 db IP-cím mögött (amely azért az otthoni felhasználókra nem jellemző), praktikusan érdemes 1-5 közé tenni ezt a számot. Ha sok kérést kapunk Proxy vagy nagyvállalati tűzfal gépek mögül, akkor érdemes növelni ezt a számot.

MaxIdleTime [perc]

Ezzel percben megadhatjuk, mennyi inaktivitás után szakítsa meg a kapcsolatot a szerver a klienssel. Mivel ezt az opciót a legtöbb klienssel felül lehet bírálni, ezért nem érdemes túl nagy számot választani, praktikusan 5-15 perc közötti értéket adjunk meg.

MaxLoad [szám]

A túlterhelés megakadályozására alkották meg ezt a remek opciót, amely nem engedi a Pure-FTPd-nek, hogy egy bizonyos rendszerterheltség (load) felett további erőforrásokat emésszen fel. Általánosságban elmondható, hogy a load[39] egy olyan iránymutató szám a Unix/Linux rendszerekben, amely esetében az 1-es alatti érték jelképezi az üzemszerű erőforrás-felhasználást. Ez nem azt jelenti, hogy a MaxLoad értékének 1-et kellene beírnunk, de érdemes a top vagy a htop program segítségével monitorozni az FTP szerver és a szerver terheltségét, és egy olyan értéket meghatároznunk, amely a felhasználók viselkedése és a hardver tűréshatárán belül egy ésszerű érték. Ez jelenti azt is, hogy ekkor a lemez I/O várakozás még nem okoz nagyobb loadot és a memóriafelhasználás sem terelődik át a swap tartományba. A példa kedvéért, ez egy nagyobb terheltségű FTP szerveren jelenleg 12-es érték, de ez teljesen szubjektív és egyedi mérésekre alapozott.

TLS [0-3]

Ezzel engedhetjük vagy tilthatjuk, és meghatározhatjuk az SSL használatát a következőképpen:

0: az SSL/TLS réteg tiltva van.

1: engedjük az SSL/TLS-t és a sima titkosításmentes kapcsolatot is

[38] http://manpages.ubuntu.com/manpages/precise/man8/inetd.8.html

[39] http://en.wikipedia.org/wiki/Load_(computing)

2: visszautasítja a nem SSL/TLS mechanizmussal kezdődő azonosítási kísérleteket, anonymous kapcsolat esetében is.

3: visszautasítja a titkosításmentes kapcsolatot és SSL/TLS alapú adatkapcsolatot épít ki.

Ugyanakkor az adatkapcsolati réteg esetében is kényszeríti az SSL/TLS kapcsolatot.

A 2-es és 3-as opció esetében első lépésben létre kell hoznunk az SSL kulcsot és bemásolnunk a következőképpen:

```
mkdir -p /etc/ssl/private
openssl req -x509 -nodes -newkey rsa:4096 -keyout \
 /etc/ssl/private/pure-ftpd.pem \
 -out /etc/ssl/private/pure-ftpd.pem
chmod 600 /etc/ssl/private/*.pem
```

Az Ubuntu LTS-ben lévő Pure-FTPd TLS/SSL támogatással kerül csomagolásra, így az SSL kulcs elkészítése, valamint az opció bekapcsolása után további teendőre nincs szükség.

UserBandwidth [KB/s]

Akárcsak az anonymous esetében, itt is KB/Sec-ben adhatjuk meg az azonosított felhasználó maximális átviteli képességét. Vigyázzunk: az esetlegesen ugyanabban a hálózatban lévő felhasználók akár gigabites forgalmat is generálhatnak.

MinUID [uid]

Ezzel az opcióval adhatjuk meg az azonosítás során, hogy mely felhasználók csatlakozhatnak, hiszen jellemzően a sima felhasználói szint 1000-es UID-nál kezdődik, így a root és a többi 1000 alatti felhasználó még a jó jelszó megadása után sem tud csatlakozni, amely kifejezetten kívánatos óvintézkedés. Így tartalma legyen 1000.

PassivePortRange [szám szám]

Ez egy rendkívül fontos opció, mivel jelen esetben kényszeríthetjük, hogy az FTP forgalom passzív üzemmód[40] esetén milyen tartományban üzemeljen, így a tűzfalszabályokat is könnyedén mögé igazíthatjuk a következő módon:

A fájl tartalma legyen „42000 42100”, sima szóközzel elválasztva, majd pedig helyezzünk el a tűzfalunkban egy ehhez hasonló szabályt:

```
#FTP Passive Port + SSL
$IPTABLES -A INPUT    -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 20 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A tcp_packets -i eth0 -p tcp -m tcp --dport 42000:42100 -j ACCEPT
```

Ezek után a kliensen az SSL/TLS és a passzív mód kapcsolókat kell bekapcsoltatni a felhasználóval.

Természetesen számtalan egyéb[41] opciója is létezik még a Pure-Ftpd-nek, azonban a legfontosabb és legnépszerűbb opciókat felsoroltuk.

Vsftpd[42]

A projekt weboldala szerint a Vsftpd valószínűleg a legbiztonságosabb és leggyorsabb FTP szerver a Unix rendszerű gépek esetében. Az biztos, hogy akárcsak a Pure-FTPd esetében számtalan biztonsági és működést segítő opcióval rendelkezik[43]. Ugyancsak támogatja a virtuális felhasználókat és az SSL/TLS-t is. Érdekessége, hogy a legnagyobb Linux terjesztések majdnem kivétel nélkül Vsftpd-t használnak a nagy terheltségű ISO és egyéb kiszolgálásaikra.

[40] http://en.wikipedia.org/wiki/File_Transfer_Protocol
[41] http://download.pureftpd.org/pure-ftpd/doc/README
[42] https://security.appspot.com/vsftpd.html
[43] https://security.appspot.com/vsftpd.html#features

ProFTPD[44]

Talán az egyik legtöbb platformon futó, rendkívül széles körben konfigurálható FTP démon, amelynek hosszú múltja van. Sajnos azonban a hosszú múltban adódtak kellemetlen biztonsági incidensek is, talán ezért a projekt főoldalán csak a magas konfigurálhatóságot említik első sorban. Ennek ellenére a ProFTPD igen megbízható és sokoldalú FTP szerver megoldás.

Számtalan más szabad szoftveres FTP szerver megoldás létezik még, de ma ezek a leginkább elterjedtek. Amennyiben tájékozódni akarunk a többi megoldás tudását és elérhetőségét illetően, akkor a Wikipedia[45] egy remek táblázatban ezt összefoglalja számunkra.

SFTP[46]

Az SFTP alrendszer részben az SCP utódja, valójában az SSHD egy jól elkülönített alrendszere. Az SCP egy remek eljárás az SSH titkosított rétegein belül fájlátvitelre. Ugyanúgy támogatja az SSH összes biztonsági megoldását, mint a parancssori mód. Azaz érvényes rá az Allow Users direktíva, és a kényszerített kulcshasználat is, amikor a jelszavas azonosítást kikapcsoljuk, és RSA vagy DSA kulcsok segítségével végezzük el az azonosítást. De használható vele az OTP (one time password) azonosítás is. Gyengesége, hogy egy standard SSH szerver felépítése esetén az SCP jogot nem lehet jól elkülöníteni az SSH parancssori felület jogaitól, valamint nehéz chroot-ba kényszeríteni. Van ugyan rá egy megoldás az scponly shell[47], amikor is egy speciális chroot-ba kényszerített, nagyon limitált shell környezetet kap a felhasználó, amely segítségével a sima SSH-t már nem tudja használni, csak egy adott könyvtár fölötti részhez fér hozzá. Ezzel jellemzően az automatizált mentéseket lehet biztonságosabbá tenni. Az SFTP esetében azonban már az SSH konfigurációjában megmondhatjuk, hogy ki milyen séma szerint férhet és mihez. Teljesen biztonságos, mivel minden adat az SSH BINARY csatornáján keresztül megy, így továbbra is lehetőség van az összes biztonsági azonosítás és korlátozás használatára. Gyakorlatilag egy minimális odafigyeléssel ma talán ez a legmegfelelőbb megoldás a sima FTP kiváltására. Szerencsére a kliensek is számos platformon támogatják, így a fejlesztőknek is sok eszköz áll rendelkezésükre, sőt manapság már a kereskedelmi weblapfejlesztő és verziókövető rendszerek is támogatják ezt a módot. Részletesebb beállítási útmutatót a Távoli adminisztráció fejezet SFTP részében találunk.

Webstatisztika-készítő szoftverek

AWStats[48]

Ez egy alapvetően Perl nyelven írt, igen részletes statisztikakészítő program. Szükséges hozzá CGI támogatás az Apache szerveren belül, éppen ezért egy kicsit komplikáltabb az üzemeltetése is egy sima HTML-t kezelő programnál. Viszont sokkal több és részletesebb lehetőséget kapunk. Tudása gyakorlatilag a kereskedelmi statisztikakészítő programok tudásával vetekszik. A pontos funkciók részletezése a honlapján[49] megtalálható. Üzemeltetésénél érdemes számításba venni, hogy CGI

[44] http://proftpd.open-source-solution.org/
[45] http://en.wikipedia.org/wiki/List_of_FTP_server_software
[46] http://en.wikibooks.org/wiki/OpenSSH/Cookbook/SFTP
[47] https://github.com/scponly/scponly/wiki
[48] http://awstats.sourceforge.net/
[49] http://awstats.sourceforge.net/#FEATURES

módban fut, ezért érdemes korlátozni és védeni esetleg https és htaccess használatával az elérhetőségét. Rendelkezik magyar nyelv támogatással is.

Webalizer[50]

Az egyik legrégebbi és leggyorsabb (hiszen C nyelven írták) naplóelemző szoftver. Beállítása és üzemeltetése is igen egyszerű. Az Awstats-szal ellentétben a Webalizer futtatását vagy emberi beavatkozással parancssorból (vagy tömegesen szkriptelve) vagy pedig cron-ból időzítve végezhetjük. Nincsen tehát gyakorlatilag támadási felülete, ugyanis sima HTML kódot + képeket publikál egy előre meghatározott könyvtárba. Honosított számtalan nyelvre, köztük a magyarra is. Tudása nagyon hasonló az Awstats-hoz, azonban a statikus tartalom kezelése miatt kevesebb extrát tud. Egyszerű felhasználású eszköznek igen kiváló. Bár a publikált területen a HTML oldalak támadási felületet nem kínálnak, azonban maga a statisztika adhat lehetőséget visszaélésre (például: egy azonosítás nélküli felület, amelyet sokszor használunk, benne lesz a publikus adatok között) ezért érdemes szintén https és htaccess védelemmel ellátni.

Piwik[51]

Egy PHP és MySQL alapokon nyugvó nyílt forráskódú, teljesen valós időben működő webforgalom-analizáló szoftver. A weboldal tanúsága szerint több mint 320.000 weboldal használja és alternatívái akarnak lenni a Google Analytics-nek. Jelenleg 46 nyelvet támogat, köztük a magyart is. Tudása összetettebb és szélesebb körű mint társaié, valamint nagy előnye, hogy az adatainkat nem szolgáltatjuk ki egy 3. félnek, azaz saját rendszerünkön belül tároljuk az adatokat.

BBclone[52]

Akár csak a Piwik, PHP alapokon nyugvó valós idejű statisztikaszoftver, ahol szintén nem kell kiszolgáltatni az adatainkat egy kereskedelmi cégnek. Nyílt forrású és számos nyelven támogatott, köztük magyarul is.

[50] http://www.webalizer.org/
[51] http://piwik.org/
[52] http://bbclone.de/