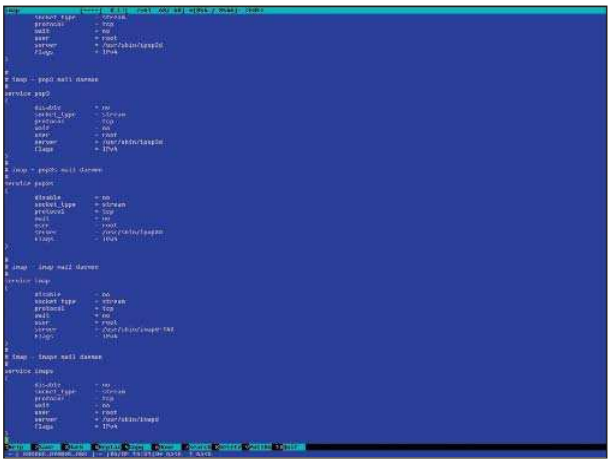
Linuxos kiszolgálót mindenkinek! (5. rész)

A SuSE Linux mint kiszolgáló – kisvállalati és otthoni környezetben.

Cikksorozatom ötödik része elsősorban a különböző levelezési protokollokkal foglalkozik, kiemelt figyelmet szentelve a POP3 és IMAP4 protokolloknak és különböző változataiknak. Cikkem előző részében Postfix SMTP-kiszolgálónkat telepítettük és állítottuk be; most azon leszünk, hogy megfelelő felületet teremtsünk a kiszolgálón a különböző levelezőprogramok csatlakozásához.

Az elmélet

Először nézzük meg a szóban forgó két protokollt! A POP3 (Post Office Protocol version 3) igen népszerű protokoll, a felhasználók jelentős része ezt használja az elektronikus üzeneteknek az internetes kiszolgálókról való letöltésére. A POP3 protokoll segítségével a felhasználó levelezőprogramja letölti a leveleket a kiszolgálóról, majd az esetek nagy többségében törli is onnan őket. A POP3 feladata, hogy a kiszolgálóval kapcsolatot létesítsen, elvégezze a felhasználó hitelesítését, majd a felhasználó alapértelmezett bejövő postafiókjából kiolvassa a leveleket. Ugyanakkor semmilyen módosításra nem képes a kiszolgálón, a levelek rendezése ebben az esetben csak az ügyfél oldalán történhet meg. A működés pontos leírása az RFC 1939 számú leírásban található meg, ami az interneten lelhető fel. (➲ http://www.faqs.org/rfcs/rfc1939.html)
Az IMAP4 (Internet Message Access Protocol version 4) (➲ http://www.faqs.org/rfcs/rfc1730.html) protokoll lehetőséget biztosít arra, hogy a postafiókunkat ne a saját gépünkön, helyben tároljuk, hanem a tartalmát a kiszolgáló őrizze meg számunkra. Ebben az esetben különböző műveletek végrehajtására nyílik lehetőségünk a kiszolgálón, így könyvtárakat hozhatunk létre, leveleket másolhatunk közöttük, újakat hozhatunk létre vagy éppen törölhetjük őket. Ez nagyon hasznos szolgáltatás lehet akkor, ha a felhasználónak arra van szüksége, hogy a világ bármely részéről azonos módon férjen hozzá a postafiókjaihoz. Mindkét protokoll adatfolyam alapú kapcsolatot használ a TCP protokoll felett. A POP3 alapbeállításként a 110-es, míg az IMAP4 a 143-es kaput használja a kapcsolattartásra, így a telepítés után ezeket a kapukat kell engedélyezni a tűzfalon. Mindkét protokoll alapvető hiányossága, hogy a felhasználóhitelesítést és az adatkapcsolatot egyaránt nyílt formában végzi, így a folyamat lehallgatható. Mivel a postafiókhoz tartozó felhasználóknak gyakran egyéb hozzáférésük is van a kiszolgálóhoz, a nevüket és jelszavukat megszerezve a kiszolgáló nagymértékben sebezhetővé válik. Erre kínál megoldást mindkét protokoll esetén az SSL-es (Secure Sockets Layer) változat. A protokolloknak ezek a változatai mind a felhasználói hitelesítés, mind a kapcsolattartás folyamatát titkosított formában végzik, így a támadónak



Az IMAP csomag beállítása a xinetd bejegyzések között

nincs esélye rá, hogy felhasználóneveket és jelszavakat hallgasson le. Ezek a protokollok úgynevezett nyilvános kulcsú titkosítást használnak, aminek nagy előnye, hogy a felépítéséből kifolyólag jelenlegi matematikai tudásunk szerint gyakorlatilag lehetetlen visszafejteni. Ahhoz, hogy ezeket a protokollokat használni tudjuk, úgynevezett tanúsítványok létrehozására van szükségünk, amelyek segítségével a felhasználó és a kiszolgáló közötti adatfolyam titkosításra kerül. Ezeket a tanúsítványokat nekünk kell a telepítés után létrehoznunk, méghozzá eszményi esetben minden kiszolgált tartományhoz külön tanúsítvánnyal. Az utóbbi azért fontos, mert a tanúsítvány a kiszolgáló nevét is tartalmazza, és amennyiben az nem egyezik a megszólított kiszolgáló címével, erről minden esetben figyelmeztetést kapunk. Ez akkor fordulhat elő, ha kiszolgálónknak több neve is van, például ***mail.tartomany.hu*** és ***kiszolgalo.tartomany.hu***. Amennyiben a tanúsítvány a ***mail.tartomany.hu*** címre lett kiállítva, mi viszont a ***kiszolgalo.tartomany.hu*** címen próbálunk csatlakozni hozzá, ügyfélprogramunk erről nagy valószínűséggel figyelmeztetést fog küldeni, hogy felhívja rá a figyelmünket: a kiszolgáló esetleg egy hamis kiszolgáló vagy rossz a tanúsítvá-

nyunk. Mindez azért fontos, mivel a nyilvános kulcsú titkosítás csak addig használható biztonságosan, amíg a tanúsítványok érvényesek és nem sérültek meg. Amint a támadó hozzáfér a tárolt titkos kulcsokhoz és azokkal készít tanúsítványokat, a kapcsolat többé már nem megbízható.
A POP3 SSL-es változata a POP3S, amely a 995-ös kaput használja a kapcsolattartásra, míg az IMAP4 SSL-es változata az IMAPS, amely a 993-as kapun tartja a kapcsolatot az ügyfelekkel.

A gyakorlat

Most, hogy az elméleti alapokkal megismerkedtünk, nézzük a gyakorlatot és telepítsük, valamint állítsuk be a kiszolgálókat! Mind a POP3, mind az IMAP protokoll feladatainak ellátására több kiszolgálócsomag (úgynevezett démon) is létezik, így a POP3-ra az `imap` csomag, a `solidpop3` csomag, vagy akár a `qpopper` csomag. IMAP esetén pedig a legnépszerűbb az `uw-imapd`, amely a SuSE 8.2-es és 9.0-s kiadások alatt az `imap` csomag neve alatt található, valamint itt van a Cyrus IMAPD, amely egy olyan IMAP-kiszolgáló, ami képes a kiszolgálón található felhasználóktól függetlenül is postafiókokat kezelni. Mi most a POP3 és az IMAP4-es protokoll feladatait az `uw-imapd` csomagra fogjuk bízni, így a YaST-tal az `imap` csomag telepítésére lesz szükségünk. Az `imap` csomag futtatásához szükségünk lesz vagy az `inetd`, vagy a `xinetd` Internet Service Daemon (internetszolgáltatás démon) telepítésére, ugyanis ezeket a démonokat csak az adott kapura beérkező kérés esetén indítja el a rendszer. A kérések érkezését pedig az `inetd`, illetve a `xinetd` hivatott figyelni.
A `xinetd` az `inetd` egy továbbfejlesztett, feladatai bővített változata, így ennek a használatát javaslom, amennyiben ez nem jelent lényeges átszervezést a rendszerben. A beállítások leírása során én a `xinetd` használatára támaszkodom.
A csomagok telepítése után a */usr/sbin* könyvtárban megjelenik az `ipop2d`, `ipop3d` és `imapd` futtatható állomány. Ezek lesznek a különböző protokollok kiszolgálódémonjai, amelyeket a `xinetd` internetszolgáltatás-démon indít a megfelelő kapura érkező kérés esetén.
Ezeknek a démonoknak alapvetően nincsenek beállításokra szolgáló állományai, így a csomag telepítése után akár használhatjuk is őket.
Beállítás gyanánt mindössze engedélyezni kell a tűzfalon a megfelelő kapuk elérését, valamint a `xinetd` kiszolgálót szükséges beállítani a kérések megfelelő helyre való továbbítása érdekében.

A nyilvános kulcsú titkosítás

A nyilvános kulcsú titkosítás (Public Key Encoding) alapgondolata, hogy gyakorlati titkosságot tud nyújtani anélkül, hogy a kapcsolattartó partnerek a kapcsolat megkezdése előtt kulcsot cseréltek volna, mint az a hagyományos szimmetrikus titkosítások esetén szükséges. A nyilvános kulcsú titkosító két kulccsal dolgozik: egy nyilvános kp és egy titkos ks kulccsal. A kódoláshoz a nyilvános kulcsot használjuk, míg a dekódolás a titkos kulcs segítségével történik. Ezért is nagyon fontos, hogy a titkos kulcsunkhoz senki ne férhessen hozzá.
Az algoritmus alapját az a matematikai feladvány adja, amely szerint egy szám prímtényezős felbontása algoritmikus értelemben véve nehéz feladat.
Most matekozzunk egy kicsit és nézzük meg, hogy miként is működik az algoritmus! Vegyünk két nagy, akár több száz számjegyű prímszámot, p-t és q-t, továbbá nézzük meg a szorzatukat. Ez nevezzük n-nek. A kódolt szöveget úgy értelmezzük, mint y=xe (mod n), ahol x a kódolandó szöveg, y a kódolt szöveg, e a kódoló függvény, (mod n) az n-nel való maradékosztás jele. Az e kódoló függvényt úgy válasszuk meg, hogy az relatív prím legyen φ(n)-hez, azaz az n szám osztóinak számához. A φ(n) abban az esetben, ha n prímszám, akkor (n-1). Mivel p és q, a két kiinduló számunk prímszám, a φ(n)= φ(p*q)=(p-1)*(q-1). Tehát e-t úgy válasszuk meg, hogy e és (p-1)*(q-1) szám relatív prímek legyenek, tehát (e, φ(n))=1.
Kódolni ezek után már tudunk, most nézzük a dekódolást! Az elv itt is hasonló, egy kongruenciát kell megoldanunk, de most az x=yd (mod n) műveletről van szó, ahol x a kiinduló, tehát immár dekódolt szöveg, y a dekódolandó szöveg, d a dekódoló függvény, valamint (mod n) itt is az n-nel való maradékosztás jele. A d dekódoló függvény és az e kódoló függvény között jelen esetben is összefüggésnek szükséges lennie, mint bármely kódoló–dekódoló felépítés esetén, ez az összefüggés pedig a d*e=1 (mod φ(n)) (tehát a d és e függvény egymás inverze (mod φ(n))). Amennyiben ezt a kongruenciát is megoldottuk, megkapjuk a d dekódoló függvényt, ennek ismeretében pedig már egyszerűen dekódolható az y szöveg.
Az üzenetek kódolásához az n számot és az e kódoló függvényt nyilvánosságra hozzuk, a többit pedig szigorúan titokban tartjuk. Ezek alapján remekül látszik, hogy amíg n-ből nehéz vagy a gyakorlatban lehetetlen előállítani p-t és q-t, tehát nem tudjuk a számot prímtényezőire bontani, addig a titkosítás megfelelő méretű p és q választása esetén gyakorlatilag feltörhetetlen. Hiszen amíg n-t nem tudjuk felbontani, nem bírjuk meghatározni a φ(n)-t, így nem vagyunk képesek meghatározni a d dekódoló függvényt sem.
Ezt a titkosítást mára rengeteg helyen használják, így az összes számítógépes protokoll SSL-es kiterjesztése is, ilyen például az IMAP SSL, POP3 SSL, HTTPS vagy akár az SSH is.
A kódolófolyamat sajátossága, hogy az algoritmus remekül használható digitális aláírás készítésére, hiszen ha megnézzük, hogy a feladó a saját titkos kulcsával írja alá a levelét, akkor az aláíró nyilvános kulcsa birtokában bárki ellenőrizheti a feladó kilétét.
Az eddigieket összefoglalva elmondhatjuk, hogy a kapcsolattartáshoz az egyik fél egy nyilvános kulcsot és egy titkos kulcsot készít, majd a nyilvános kulcsot elérhetővé teszi, például egy weblapon. Amennyiben egy feladó titkosított üzenetet szeretne küldeni nekünk, letölti a nyilvános kulcsunkat, és ezzel kódolja az üzenetet. Ezt a kódolt üzenetet csak a nyilvános kulcshoz tartozó titkos kulccsal fogjuk tudni dekódolni, ezért csak a címzett teheti meg.

A xinetd beállítása

A `xinetd` démon beállítását a */etc/xinetd.conf* állományon keresztül végezhetjük el. Az állomány tartalmazza a démon alapvető beállításait, mint például a naplózás helyét, a kérést indítható ügyfelek címét, címtartományát, valamint a kéréseket fogadó hálózati csatoló címét. A SuSE Linux csomagjából telepített `xinetd` csomag a különböző démonokra vonatkozó beállításokat a */etc/xinetd.d* könyvtárban tárolja. A */etc/xinetd.conf* állományban erre való utalásként az alábbi sor található meg:

```
includedir /etc/xinetd.d
```

Ennek a sornak a jelentése annyi, hogy a –*/etc/xinetd.d* könyvtár alatt található állományok tartalmát a rendszer a beállításállomány futtatásakor fűzze a beállításfájlokhoz. Ennek a megoldásnak annyi értelme van, hogy a beállítások áttekinthetők, mert egyesével, külön állományban vannak elhelyezve. Ennek ellenére mi akár közvetlenül a */etc/xinetd.conf* állományba is készíthetünk bejegyzéseket, bár a rendszer által alkalmazott megoldás sokkal áttekinthetőbb.
A különböző beállítások tehát a */etc/xinetd.d* könyvtárban lévő állományokban találhatóak; bennünket ezek közül most az *imap* állomány érdekel.
Az *imap* állományban találhatóak az `imap` csomag által telepített démonok beállításai, így itt tudjuk a POP2, POP3, POP3 SSL, IMAP4 és IMAP4 SSL démonokat beállítani.
Az összes bejegyzés a következő sorral kezdődik:

```
service <szolgáltatás neve>
```

Itt a szolgáltatás neve a */etc/services* állományban megtalálható szolgáltatásokhoz tartozó kapu számát határozza meg, például a POP3 a 110-zel is helyettesíthető lenne.
A kapcsos zárójelen belül található kapcsolók közül az első a `disable`. Ennek *yes* értéke esetén a démon a kapura érkező hívás esetén sem indul el, ugyanis le van tiltva. Amennyiben egy-egy szolgáltatást szeretnénk használni, ezt az értéket állítsuk *no*-ra. A következő kapcsoló a `socket_type`, ez a kapcsolat típusát adja meg; ne nyúljunk hozzá, hagyjuk `stream` (folyam, adatfolyam) értéken. A `protocol` kapcsoló értelemszerűen a protokoll típusát adja meg, ez `tcp`, ne állítsuk át!
A `wait` és `user` kapcsolókat se piszkáljuk, kivéve, ha például a démont nem rendszergazdaként akarjuk futtatni. Ehhez viszont a különböző futtatási és tárolási állományoknál is gondoskodni kell a megfelelő biztonsági beállításokról, hozzáférési jogosultságokról. A `server` kapcsoló adja a kiszolgáló futtatási állományát, erre még szükségünk lesz, ezért jegyezzük meg. A `flags` kapcsoló az IP-hálózat változatát adja meg, ezt se piszkáljuk, hacsak nem IPv6-os hálózattal rendelkezünk.
Alapesetben számunkra az IMAP SSL és a POP3 SSL csatorna lesz érdekes, mivel a legtöbb levelezőprogram ezeket támogatja, valamint titkosított felhasználóazonosítást és adatátvitelt tesznek lehetővé. Így tehát el kell készítenünk egy `service imaps` és `service pop3s` bejegyzést, ahol a `disable` kapcsolót *no* értékre kell állítani.
Amennyiben az SSL nélküli protokollváltozatokat is használni szeretnénk, az előbb említett `disable` kapcsolót az IMAP és POP3 szolgáltatások beállításánál is állítsuk *no* értékre.
A beállítások végeztével a beállításokat a `/etc/init.d/xinetd restart` paranccsal tölthetjük be. Amennyiben a tűzfalunkon a megfelelő kapuk nyitva állnak, a szolgáltatásokat akár ki is próbálhatjuk.

Nem ilyen egyszerű az élet…

Mielőtt a jól végzett munka édes gyümölcseit élvezvén hátradőlnénk, a `telnet` paranccsal próbáljuk meg megnyitni a beállított szolgáltatásokat. A kiszolgálón egy konzolt indítva a `telnet localhost 110` (ez a POP3), a `telnet localhost 143` (ez az IMAP), a `telnet localhost 993` (ez az IMAP-SSL), illetve a `telnet localhost 995` (ez pedig a POP3-SSL) paranccsal próbálhatunk meg csatlakozni a kiszolgálóhoz. A POP3- és IMAP-kiszolgálókhoz való csatlakozás valószínűleg könnyűszerrel menni fog, és POP3 esetén a következő üzenetet fogjuk kapni:

```
kackac:/etc/xinetd.d # telnet localhost 110
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK POP3 localhost v2001.80 server ready
```

Ekkor bejelentkeztünk a kiszolgálóra – úgy tűnik, minden működik. A CTRL+] billentyűkombinációval ki is léphetünk. IMAP esetében:

```
kackac:/etc/xinetd.d # telnet localhost 143
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS
STARTTLS LOGINDISABLED] localhost IMAP4rev1
2002.332 at Sun, 15 Feb 2004 19:00:21 +0100 (CET)
```

Úgy tűnik, ide is sikerült bejelentkezünk, de ne örüljünk annyira. Éles szemű olvasók észrevehették a `LOGINDISABLED` szócskát a kiszolgáló kimenetén. Nos, biztonsági megfontolásokból a SuSE 8.2-es és 9.0-s kiadásokba épített IMAP-kiszolgálókra csak az SSL-es protokollt használva lehet bejelentkezni. Ugyanakkor felmerül a kérdés, hogy mit tegyünk, ha ez nekünk nem elegendő, mert például a *SquirrelMail* webes levelezőrendszert szeretnénk használni, ami nem ismeri az IMAP protokoll SSL-es kiterjesztését. Ez ugye elvileg nem is jelent biztonsági kockázatot, mert ugyan a csatorna nem használ titkosítást, ellenben nem is jut ki a kiszolgálóról, hiszen mind a webes levelezőrendszer, mind pedig az IMAP-kiszolgáló ugyanazon a gépen található.
Mi a megoldás? Mint fent írtam, ez a 8.2-es és 9.0-s rendszerekben bevezetett korlátozás. Keressünk tehát egy 8.1-es csomaggyűjteményt, például az ➲ ftp.suselinux.hu kiszolgálón, töltsük le az *ftp.suselinux.hu/8.1/suse/i586* könyvtárból az *imap-2001a-155.i586.rpm* csomagot, majd egy Midnight Commander (mc) segítségével lépjünk be az *rpm* állományba. Ezen belül a */usr/sbin* könyvtárban egy `imapd` futtatható állományt találunk, ő lesz a mi barátunk. Másoljuk be ezt az állományt, mondjuk *imapd-143* néven a

/usr/sbin könyvtárba, módosítsuk a */etc/xinetd.d/imap* állományban az IMAP szolgáltatáshoz tartozó beállítások között a `server` kapcsolót */usr/sbin/imapd-143* névre. Ezek után mentsük az állományt és a már megismert módon indítsuk újra a xinetd kiszolgálót.
Láss csodát, ezek után eltűnik a `LOGINDISABLED` felirat a kiszolgáló címsorából. Nyertünk.
Figyelem, azért nem a teljes `rpm` csomag telepítését javasoltam, mert az felülírja a mostani csomagunkat és a 8.1-es változathoz tartozó IMAP-démon pedig az SSL-es IMAP protokollt nem ismerte.
Figyeljünk oda rá, hogy a titkosítás nélküli protokollokat lehetőleg csak a legutolsó esetben használjuk, illetve akkor, ha meg vagyunk győződve róla, hogy biztonságos környezetben tesszük (például a `localhost`-on vagy saját vállalati, illetve otthoni hálózatunkon belül).
Na, akkor most vegyük sorra a POP3 SSL és IMAP SSL protokollokat!
Mind a 993-as, mind a 995-ös kapura való bejelentkezéskor az alábbi megy végbe:

```
kackac:/etc/xinetd.d # telnet localhost 995
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Connection closed by foreign host.
kackac:/etc/xinetd.d #
```

Azonnal visszakapjuk a parancssort. „Hmmm…”, gondolhatjuk, itt valami nem felel meg. Gyorsan utánanézünk a */var/log/mail* naplóállományban, hogy mi lehet a baj.
Itt a következőt találjuk:

```
Feb 15 19:18:32 kackac ipop3d[4535]: Unable to
➥load certificate from /etc/ssl/certs/ipop3d.pem,
➥host=localhost [127.0.0.1]
```

Na igen, megvan a hiba. Korábban beszéltünk már a tanúsítványokról, amikre az SSL protokollok működéséhez szükség van. Nos, pont ezt a tanúsítványt hiányolja a kiszolgáló. De semmi vész, mindjárt létrehozzuk a megfelelő tanúsítványokat.

Hitelesítésszolgáltatás röviden

Linux-rendszerek alatt az `openssl` csomag segítségével hozhatunk létre Certification Authority-t, azaz hitelesítésszolgáltatót. Az ilyen hitelesítésszolgáltató rendszer feladata a fő tanúsítvány létrahozása, illetve új tanúsítványok kibocsátása, kezelése és adott esetben visszavonása.
A hitelesítésszolgáltató rendszerek egy állományrendszerhez hasonlóan faszerkezetben épülnek fel. Legfelül a fő tanúsítvány áll, ami különleges a rendszerben, mivel ez az egyetlen olyan tanúsítvány, amely saját magával van aláírva. Az összes alsóbbrendű tanúsítvány úgy készül, hogy a hierarchiában egy feljebb álló tanúsítvánnyal kerül aláírásra, így igazolva annak hitelességét. A szerkezet felépítéséből látszik, hogy akárcsak a nyilvános kulcsú titkosításnál a két prímszámunkra, úgy itt a fő tanúsítványra kell nagyon vigyázni. Ha a fő tanúsítvány illetéktelenek számára hozzáférhetővé válik, az összes általunk kibocsátott tanúsítvány hitelessége elveszett.
Nézzük, miként tehetjük elérhetővé rendszerünk IMAP és POP3 SSL szolgáltatásait. Pillanatnyilag nem foglalkozunk a hitelesítésszolgáltató rendszerrel, létrehozunk egy tanúsítványt, amit azok a felhasználók fognak használni, akik megbíznak a mi rendszerünkben. Ez a tanúsítvány a későbbiekben majd lecserélhető lesz a hitelesítésszolgáltató rendszerből kibocsátott tanúsítvánnyal.
Először is keressük meg, hogy a rendszerünk melyik könyvtárban tárolja az elkészített tanúsítványokat – SuSE alatt ez általában a */etc/ssl/certs* könyvtár. Váltsunk tehát ebbe a könyvtárba, majd adjuk ki a következő parancsot:

```
openssl req -new -x509 -nodes -out imapd.pem
➥-keyout imapd.pem -days 365
```

Fontos, hogy a készítendő tanúsítvány adatainak kitöltésekor a *Common Name* (CN) mindenképpen az a név vagy IP-cím legyen, ami alapján hivatkozni szeretnénk a kiszolgálóra, ellenkező esetben a tanúsítvány hibát jelezhet.
A `days` kapcsoló állításával tudjuk állítani, hogy a tanúsítvány milyen hosszú időtartamra legyen érvényes. A 365 természetesen pontosan egy évet jelent.
Amennyiben több tartományt is ki szeretnénk szolgálni, ezek megkülönböztetésére készítsünk különböző tanúsítványokat. Ezt úgy tehetjük meg, ha a fenti parancsot a következőképpen módosítjuk:

```
openssl req -new -x509 -nodes -out
➥imapd-sajat.tartomany.hu.pem -keyout
➥imapd-sajat.tartomany.hu.pem -days 365
```

vagy

```
openssl req -new -x509 -nodes
➥-out imapd-XXX.XXX.XXX.XXX.pem
➥-keyout imapd-XXX.XXX.XXX.XXX.pem -days 365
```

Ebben az XXX.XXX.XXX.XXX a kiszolgáló IP-címe.
Ha elkészültünk a tanúsítvánnyal, a következő paranccsal olyan formába alakíthatjuk át, amelyet a legelterjedtebb ügyfelek fel tudnak dolgozni:

```
openssl x509 -in tanusitvany-neve.pem
➥-out tanusitvany-neve.crt
```

Mostani cikkemben rendszerünket felkészítettük a különböző levelezőügyfelek fogadására és elkészítettük a szükséges tanúsítványokat. Mindenki próbálgassa a rendszert, de tartsuk szem előtt az alapvető biztonsági szempontokat. Lehetőleg ne nyissunk felesleges réseket a tűzfalon és ne forgalmazzunk kódolatlan formában jelszavakat!
Sok sikert mindenkinek!

Illés Viktor (viktor@ei.hu)
23 éves, a BME műszaki informatikus szakának hallgatója, mellette weblapokkal, linuxos és windowsos rendszerekkel foglalkozik. Szabadidejét legszívesebben a szabadban tölti, teniszezik és kerékpározik.