Az OpenSSH száz meg egy előnye (1. rész)

Mick az ssh felszínét kapargatja.

Eljött a Paranoid Pingvin mai mesedélutánjának ideje. De ne aggódjanak – ez a mese pont ugyanolyan mazsoláznivalókról szól, mint amilyeneket már megszokhattak a Linuxvilágtól. Sőt, az az igazság, hogy az OpenSSH-ban olyan sok mazsoláznivaló található, hogy ez a cikk a következő számokba is átnyúlik majd!

Ebben a hónapban az ssh hátterével és szerkezetével foglalkozunk. Megnézzük, miképpen kell lefordítani, illetve telepíteni az OpenSSH-t, hogyan használhatunk ssh-t a telnet titkosított kiváltására, hogyan állítsunk be néhány alapvető változót, illetve miképpen használjuk az scp-t titkosított fájlátvitelhez. A következő hónapban az RSA/DSA bejelentkezésről fogok írni, a helyi kapu-átirányításról, a távoli parancsvégrehajtásról, illetve néhány más fejlett és hatékony ssh/OpenSSH képességről.

A rend kedvéért először arról szeretnék beszélni, miképpen került hozzánk ez a program, illetve azokról az emberekről ejtenék pár szót, akik elérhetővé tették számunkra.

Az SSH története

Amióta csak megszületett, a Unixban az volt az egyik legnagyszerűbb dolog, hogy a rendszer-karbantartási feladatokat többféleképpen is el lehetett végezni távoli konzolokról. Sajnos, a legtöbb ilyen módszer (telnet, rsh és az X, csak hogy néhányat említsünk) mindent egyszerű szövegként küld át a hálón, beleértve a jelszavakat is. Ahogy az Internetet egyre többen használták, és ahogy egyre több lelkes önjelölt kalózocska és unatkozó csomagszaglászó kisiparos jelent meg, a szövegalapú hálózati felügyelet túlhaladottá vált.

Néhány évvel ezelőtt azonban egy finn betyárszaki, *Tatu Ylonen* létrehozott egy lélegzetelállítóan jó dolgot, amit Secure Shellnek, azaz ssh-nak neveznek. Az ssh olyan eszközök csoportja, ami nagyjából a Sun rsh, rcp és rlogin parancsainak felel meg, egyetlen igen fontos eltéréssel: az üldözési kényszerrel. Az ssh-val mindent meg lehet valósítani, ami az rsh, rcp és a rlogin segítségével elérhető, kihasználva a választott szabad kódkönyvtárakat és azonosító módszereket. Az ssh egy változata erősen függ az RSA-tól, a kitűnő, de szabadalmak által védett módszertől, ami megköveteli, hogy minden őt használó program engedélyek birtokosa legyen (azaz fizessenek érte), kivéve azt az esetet, ha „nem kereskedelmi” módon használják. Gyakran azonban a nem kereskedelmi felhasználás is nehézkes lehet. Várjunk csak, az RSA US szabadalom 2000 szeptemberében lejárt – gond megoldva, nem? Majdnem: Tatunak is meg kell élnie valamiből, így aztán mire az RSA végre kötetlenné válna, az ssh vált kötötté, mivel Tatu cége, a F-Secure megszigorította az engedélykövetelményeket. Lényegében az ssh 2.0 változatától kezdve az ingyenesség és a szabad felhasználás többé már nem engedélyezett (az RSA-fejleményektől függetlenül). Akárhogy is nézzük, az ssh csak akkor válhat internetszabvánnyá (ahogy azt Tatu is szeretné), ha legalább egy ingyenes megvalósítása létezik.

Ezzel be is léptünk *Theo de Raadt* és az OpenBSD birodalmába. Az OpenBSD, a BSD ingyenes változatának, a NetBSD-nek egy biztonságra sokat adó hajtása. Theo és a nyílt forráskód közösségbeli hittestvéreink az OpenBSD fejlesztőgárdájában szerették volna beil-

1. táblázat Az ssh_config néhány alapvető értéke

Érték	*Lehetséges érték*	*Leírás*
CheckHostIP	*Yes No*	(Alapértelmezett=yes) Figyeljen-e az ügyfélgép IP-címének változására ismert számítógépkulcsok esetén, vagy sem. Figyelmezteti a felhasználót, ha ellentmondást talál.
Chiper	*3des blowfish*	(Alapértelmezett=3des) Melyik blokk-kódolást használja a ssh v.1 kapcsolatokhoz.
Chipers	*3des-cbc, blowfish-cbc, arcfour, cast128-cbc*	Milyen sorrendben próbálja végig blokk-kódolásokat a ssh v.2 kapcsolatokhoz.
Compression	*Yes No*	Használja-e a gzip programot a titkosított adatok tömörítésére vagy sem. Korlátozott sávszélesség esetén hasznos lehet, de valamelyest lassít.
ForwardX11	*Yes No*	(Alapértelmezett=yes) Átirányítja az X kapcsolatokat a titkosított csatornára, és emellett megfelelően beállítja a DISPLAY változót. Nagyon hasznos lehetőség!
Password-Authentication	*Yes No*	(Alapértelmezett=yes) Megpróbáljon-e (titkosított) Unix jelszóazonosítást végezni, az RSA/DSA azonosításon felül, vagy helyett.

leszteni az ssh-t az OpenBSD 2.6-os kiadásába. Csakhogy beleütköztek az ssh különböző megkötéseibe. Amint tudomásukra jutott, hogy egy svéd programozó, *Bjoern Groenvall* az ssh 1.2.12 (Ylonen legutolsó ingyenes – kivéve az RSA részt – változata) egy fejlettebb változatát mutatta be, az OpenBSD-s fickók egy percet sem vártak tovább, azonnal a nyilvánosság elé vitték. Az OpenSSH azóta az OpenBSD része, és mára már jóformán minden Unix-változatra átültették.
A Groenvall munkájára épülő OpenSSH (az ő OSSH nevű változata szintén elérhető) az ssh protokoll korábbi változatait egészíti ki modulrendszerrel és titkosítási eljárásokkal oly módon, hogy az OpenSSH-t egyetlen szabadalmaztatott eljárás vagy hasonló dolog használata nélkül is le lehet fordítani (például az ssh v.1 protokollok nélkül, ezek az RSA-tól függenek). Az OpenBSD csapat másik újítása, hogy az OpenSSH kódot szétválasztotta egy tiszta (clean) változatra, ami olyan egyszerű és felületfüggetlen amennyire csak lehetséges, illetve egy hordozható (portable) változatra, amit többféle Unix-változat alá is le lehet fordítani az OpenBSD mellett.
Ez az utolsó újítás a legérdekesebb a linuxosok számára: a tiszta változat megtartása teszi lehetővé a kód ellenőrizhetőségét, ami így alapvetően állandó és biztonságos. Csak miután Theo (aki egy igazi paranoid) rábólintott erre a kódra, készülhetnek el a hordozható fejlesztések. Így aztán mi egy olyan programcsomagból húzhatunk hasznot, amely nagyon biztonságos, emellett százszázalékosan Linux-megfelelő.
Mellesleg az OSSH felfedezésétől számítva, kevesebb mint két hónap kellett ahhoz, hogy az OpenBSD csapat kiadja az OpenSSH 1.2.2-t, és mindössze hat és fél hónap, hogy a teljes mértékben hordozható, és ssh v.2-megfelelő OpenSSH 2.0 megjelenjen. Ez még akkor is figyelemre méltó teljesítmény, ha tekintetbe vesszük, hogy Ylonen és Groenvall munkájára építettek, különösen, ha beszámítjuk a végeredmény kitűnő minőségét, illetve azt, hogy senki nem fizetett nekik ezért egy petákot sem!
Nos, ennyi lenne az ssh és az OpenSSH története. Remélem, egyetértünk abban, hogy elég lenyűgöző, akárcsak az OpenSSH maga, amely minden valószínűség szerint hamarosan a nyílt forráskódú Unixok elsődleges ssh változatává válik.
Egyébként az „ssh v.1.x” és a „ssh protocoll v.1” az ssh program kiadásnak, illetve a protokollnak felel meg, és nem igazán jelentik ugyanazt. De mivel a csomag és a protokoll változatszámai nagyjából megfeleltethetők egymásnak, itt most az „ssh v.1.x” kifejezést fogom használni, ha az RSA-alapú ssh/OpenSSH változatra gondolok, illetve az „ssh v.2.x” kifejezést, amennyiben a RSA/DSA-t támogató változatokra szeretnék utalni. Ha esetleg nem tudnánk, mi a különbség az RSA és a DSA között, legyen elég annyi, hogy mindkettő ugyanazt teszi, de a DSA-t nem köti semmiféle szabadalom vagy engedély.

Hogyan működik az SSH?

A Secure Shell működése erősen hasonlít a webes SSL-kapcsolatokra (nem véletlen, hogy az OpenSSH által használt titkosító függvényeket az OpenSSL nyújtja, amely a Netscape Secure Socket Layer forráskód könyvtárainak szabadon felhasználható változata). Mindkettő titkosított csatornákat épít fel, amelyek általános gépkulcsokra (host key) épülnek, vagy nyilvánossá tett bizonyítványokat (digitális aláírás – digital certificate) ellenőriznek egy megbízható szolgáltatónál (certificate authority, például a VeriSign). Lássuk, tulajdonképpen hogyan is épül fel ez a kapcsolat.
Először is, az ügyfél és a kiszolgáló (nyilvános) gépkulcsokat cserélnek. Ha az ügyfélgép még soha nem találkozott az adott nyilvános kulccsal, az ssh és a legtöbb böngésző rákérdez, hogy elfogadjuk-e a bizonytalan eredetű kulcsot. Ezután a gépek a kulcsokat használják arra, hogy létrehozzanak egy kapcsolatkulcsot (session key). Minden további adattitkosítás a kapcsolatkulcs alapján történik valamelyik blokk-kódolási módszer szerint, mint amilyen például a Triple-DES (3DES), a blowfish vagy az idea. Ezután a kiszolgáló megpróbálja azonosítani az ügyfélgépet RSA vagy DSA bizonyítványok segítségével. Amennyiben ez nem lehetséges, az ügyféltől hagyományos felhasználónév/jelszó párt vár (lehetőségünk van rhosts-típusú, tehát az ügyfél IP-címe alapján működő azonosításra RSA-kulcsokkal vagy anélkül, de az OpenSSH támogatja a Kerberos IV és az skey rendszereket is). Végül a sikeres azonosítás után kezdetét veheti a tulajdonképpeni kapcsolat: egy távoli héjprogram, biztonságos fájlátvitel, távoli parancsfuttatás stb.

Amint azt korábban már említettem, az ssh tulajdonképpen egy eszközgyűjtemény:

- sshd – kiszolgáló démonként működik minden más programhoz.
- ssh – az elsődleges eszköz: távoli héjprogram, távoli parancsvégrehajtás, illetve kaputovábbítás.
- scp – fájlátvitelhez használható eszköz.
- sftp – interaktív fájlátvitelre használható eszköz. Általában csak kereskedelmi ssh-csomagokban található meg.
- ssh-keygen – titkos, illetve nyílt kulcspárokat készít RSA vagy DSA azonosításhoz (ide értve a gépkulcsokat is).
- ssh-agent – az RSA/DSA azonosítás automatizálására szolgáló démon.
- ssh-add – titkos kulcsot tölt be az ssh-agent folyamatba.
- ssh-askpass – az ssh-add X felülete.

Ezen eszközök közül a legtöbb felhasználót csak az ssh érdekli, hiszen a „titkosított telnet” az ssh legegyszerűbb felhasználása.
Az scp, az ssh-agent, és az ssh-add, valamint az ssh a maga erős azonosítási és TCP-kapuátirányítási képességével együtt rendkívül hatékony csomagot jelent. Mivel paranoidok vagyunk, és minél több hálózaton átküldött anyagot szeretnénk titkosítani, ezzel a rugalmassággal tényleg sokat nyerhetünk.

Az OpenSSH letöltése és telepítése

Ha az OpenSSH legutolsó változatát szeretnénk letölteni akár forráskód, akár RPM formátumban, először is látogassuk meg az OpenSSH honlapját (lásd a Kapcsolódó címeket). Ugyanitt található az OpenSSL, ez elengedhetetlen az OpenSSH-hoz. Szükségünk lehet még a zlib-re, ez a freesoftware.com honlapján (lásd a Kapcsolódó címeket) érhető el.

Megjegyzem, nem biztos, hogy az RPM-ekkel könnyen boldogulunk. Amikor RPM-eket akartam telepíteni a OpenSSH.com-ról a SuSE Linuxot futtató laptopomra, minden kiválóan működött kivéve az sshd-t, ami nem települt fel a SuSE „chkconfig” csomag hiánya miatt. Az egyes Linux-változatok vagy szenvednek ebben a hibában vagy nem (már ha van rajtuk egyáltalán chkconfig). Nos, az is lehet, hogy a SuSE-nek saját RPM csomagja van az OpenSSH-hoz. Amennyiben az RPM nem működik, az OpenSSH-t (és valószínűleg az OpenSSL-t, illetve a zlibet is,) forrásból kell lefordítani. A Linux öreg motorosainak a „készítsd el saját telepítésed” stílus megszokott dolog, de ha nem tartozunk közéjük, akkor se keseredjünk el. Mindhárom csomag egy .configure parancsfájlt használ, ez pedig szükségtelenné teszi a legtöbb felhasználónak, hogy szerkesztenie kelljen bármilyen makefájlt. Ha rendszerünkben megvan a gcc fordító és az általában használatos programkönyvtárak, illetve ezek viszonylag naprakészek, a fordítás általában gyors és zökkenőmentes. Az én esetemben az OpenSSL 0.9.5a és a lib 1.1.3 változat telepítése után a következő lépéseket kellett megtenni az OpenSSH 2.1.1p4 telepítéséhez:

```
tar -xzvf openssh-2.1.1p4.tar.gz
cd openssh-2.1.1p4
./configure --sysconfdir=/etc/ssh
make
make install
```

Az INSTALL fájlban található utasításoknak megfelelően a configure parancsfájlt testre szabott értékekkel egészítettem ki: ahelyett, hogy minden beállítófájlt egyszerűen a /etc könyvtárba helyeznék, arra utasítottam, hogy hozzon létre és használjon egy alkönyvtárat /etc/sshd néven. Mivel az OpenSSH e változata az RSH és a DSA kulcsokat is támogatja, nem rossz gondolat mérsékelni azt a zűrzavart, amit az ssh a /etc könyvtárban kelt. Egy csak ügyfélalapú telepítés esetén mindössze ennyit kell tenni. Megjegyezzük, hogy a fent említett változatszámok már valószínűleg túlhaladottá váltak, mire ezt olvassuk: ne felejtsük el megnézni az OpenSSH honlapját a legfrissebb változatokért.

Ha a sshd Secure Shell démont is futtatni szeretnénk (azaz ssh kapcsolatokat kívánunk fogadni távoli gépekről), létre kell hoznunk indítófájlokat, valamint a SuSE esetében át kell szerkeszteni a /etc/rc.config fájlt.

A Red Hat könyvtár tartalmaz egy sshd.init fájlt, amit a /etc/rc.d könyvtárba lehet másolni, és a megfelelő futásszintű könyvtárból hivatkozni rá (/etc/rc.d/rc2.d stb.). Ugyanitt található az „sshd.pam”, ezt a /etc/pam könyvtárba kell másoljuk, amennyiben Pluggable Authentication Modules (PAM) rendszert és „openssh.spec”-et használunk (ezzel saját OpenSSH RPM csomagokat lehet létrehozni). Ezeket a fájlokat nyilvánvalóan a RedHat alatti használatra tervezték, de elképzelhető, hogy működnek más RedHat-alapú rendszeren is (Mandrake, Yellow Dog stb.).

A SuSE könyvtár szintén tartalmaz egy „openssh.spec” állományt a SuSE-hoz használható prpm OpenSSH csomagok létrehozásához, és egy „rc.sshd” fájlt, amit az /etc/rc.d könyvtárba lehet másolni (jelenleg /sbin/init.d a SuSE-ban). Továbbá tartalmaz egy „rc.config.ssd”-t is, ennek tartalmát a /etc/rc.config fájlhoz kell adni, hogy az rc.sshd parancsfájl helyesen működjön. Ezt egyszerűen meg lehet tenni a következő paranccsal:

```
cat ./rc.config.ssd > /etc/rc.config
```

Készítsünk egy közvetett hivatkozást (`ln -s`) az rc2.d illetve az rc3.d könyvtárakba, és a SuSE már készen is áll a titkosított héjprogramok

2. táblázat Az sshd_config értékbeállításai

Érték	*Lehetséges értékek*	*Leírás*
Port	*number*	(Alapértelmezett=22) A démon által figyelt TCP-kapuszám. A megváltoztathatóság lehetősége különösen akkor válik hasznossá, amikor kapucímátírást (Port Adress Translation) használunk, hogy több gépet rejthessünk el egyetlen IP-cím mögé.
PermitRootLogin	*yes no*	Elfogadjon-e rendszergazdai bejelentkezést, vagy sem. A leghelyesebb ezt az értéket „No”-ra állítani. A rendszergazdák előnyben nem részesített azonosítóval jelentkezhetnek be, majd su-val léphetnek be rendszergazdaként.
PasswordAuthentication	*yes no*	(Alapértelmezett=yes) engedélyezzen (titkosított) felhasználónév/jelszó belépést, vagy tartson ki az RSA/DSA kulcsalapú azonosítás mellett.
PermitEmptyPasswords	*yes no*	(Alapértelmezett=no) engedélyezzen-e üres jelszóval történő bejelentkezéseket a rendszerbe. Figyelmen kívül hagyható, ha a PasswordAuthentication=no. Ugyanígy nem vonatkozik az RSA/DSA kulcsok jelszavaira sem. (Az üres jelszó a kulcsok esetén elfogadható).
X11Forwarding	*yes no*	(Alapértelmezett=no) engedélyezze-e, hogy az ügyfelek X-windows alkalmazásokat futtassanak ssh csatornákon keresztül. Valójában semmit sem nyerhetünk azzal, hogy no-ra állítjuk, mivel az sshd_config nem képes kikapcsolni az általános TCP továbbítást is egyúttal (ami éppúgy használható X11-továbbításra is).

fogadására! A démon életre kel minden rendszerindításkor, vagy kézzel is indíthatjuk a `/etc/rc.d/rc.sshd start` paranccsal.

SSH mindenkinek, avagy mi ez a „titkosított telnet" dolog

Mi a helyzet, ha csupán interaktív héjprogramokat szeretnénk futtatni távoli rendszereken la Telnet? Nagy valószínűséggel még a beállítófájlokba sem kell belenézni, elég, ha ennyit begépelünk:

```
ssh távoli.gép.neve
```

Ezután megadjuk a jelszót (az ssh feltételezi, hogy ugyanazt a felhasználónevet szeretnénk használni a távoli gépen is), és ha nem gépeltük el, már használhatjuk is távoli héjprogramunkat! Ez vitathatatlanul egyszerű, és összehasonlíthatatlanul biztonságosabb, mint a telnet. Ha a távoli rendszeren más névvel szeretnénk dolgozni, mint helyi gépen, a -l kapcsolóval adhatjuk meg a kívánt felhasználónevet. Például ha a laptopomon „mick"-ként vagyok bejelentkezve, és szeretnék bejelentkezni ssh-val a kong-fu.mutantmonkeys.org-ra mint „mbauer", a következő parancsot használom:

```
ssh -l mbauer kong-fu.mutantmonkeys.org
```

Mit csinál ez a parancs? Egyáltalán nem látszik másnak, mint a telnet. Rákérdez, használhatja-e a kiszolgáló nyilvános kulcsát vagy sem, esetleg valamivel tovább tart a kapcsolat létrehozása, illetve a rendszer foglaltságától, a hálózattól stb. függően a kapcsolat némileg lassabb lehet, mint a telnet, de a legtöbb esetben nem sok különbséget lehet észrevenni. Viszont úgy jelentkeztem be a rendszerbe, hogy nem küldtem keresztül a hálózaton egyszerű szövegként a jelszavamat és a felhasználónevemet, és az összes adat szintén titkosítva van! Bármit megtehetek, amit csak akarok, akár egy `su` parancsot is kiadhatok, anélkül, hogy leselkedőktől kellene tartanom. Ráadásul mindez csupán egy kevéske lassulásba került!

Turkáljunk a beállítófájlokban!

Az OpenSSH beállítása szintén nem túl bonyolult dolog. A ssh-ügyfél és kiszolgáló viselkedésének irányításához mindössze két állományt kell módosítani: az ssh_config és sshd_config fájlokat. A csomag telepítésétől függően ezek a fájlok a vagy a /etc alatt, vagy a .configure --sysconfdir értékben megadott könyvtárban találhatók.
Az ssh_config a helyi gépről kezdeményezett ssh kapcsolatok általános beállítóállománya. Ezeket a beállításokat felülbírálhatjuk a parancssori kapcsolók és a felhasználók saját beállítóállományai segítségével (ezek helye a $HOME/.ssh/config). Például ha a /etc/ssh/ssh_config a következő sort tartalmazza:

```
Compression no
```

de a /home/bobo/.ssh/config-ban

```
Compression yes
```

szerepel, akkor valahányszor Bobó ssh-kapcsolatot kezdeményez, a tömörítés alapértelmezés szerint engedélyezett lesz, annak ellenére, hogy minden más felhasználó számára, akinek a $HOME/.ssh/config állománya nem tartalmazza ezt a beállítást, a tömörítés ki lesz kapcsolva. Másrészről viszont, ha Bobó a következő paranccsal indítja az ssh-t:

```
ssh -o Compression=no távoli.gép.neve
```

akkor ebben a kapcsolatban nem alkalmaz tömörítést.

Az ssh_config egy értéklistát tartalmaz, soronként egy értékkel, a következő formátumban: `kapcsoló érték(ek)`
Néhány kapcsoló kétértékű, azaz értéke vagy „yes" vagy „no". Mások viszont értékek vesszővel elválasztott listáját is tartalmazhatják. A legtöbb érték magától értetődő, de mindegyiket részletesen ismerteti az ssh(1) leírása. Az *1. táblázat* bemutat néhányat a legérdekesebbek és legfontosabbak közül (a dőlt betűk lehetséges értékeket jelölnek).
Jó néhány érték használható még ezeken kívül. Néhányat ezek közül a cikk második részében ismertetünk. A teljes lista pedig megtalálható a leírásban.

A Secure Shell Démon, az sshd beállítása és futtatása

Mindez kitűnően működik mindaddig, amíg mások által karbantartott rendszerekhez kapcsolódunk. De még nem beszéltünk arról, miképpen kell beállítanunk saját gépünket, hogy ssh-kapcsolatokat is fogadni tudjunk. Ahogy az már lenni szokott, ez is nagyon egyszerű. Akárcsak az ssh-ügyfél esetében, az sshd alapértelmezett viselkedését is egyetlen fájl, az sshd_config szabályozza, amely szintén vagy a /etc alatt, vagy a ssh beállításakor megadott könyvtárban található. Amint azt már az ssh-ügyfél esetében láttuk, a parancssori kapcsolók felülbírálhatják a beállítófájlban található értékeket. Az ügyféllel ellentétben azonban, a démonnak nincs külön beállítófájlja az egyes felhasználók könyvtáraiban, hiszen az egyszerű felhasználók nem befolyásolhatják a démon viselkedését.
A *2. táblázat* bemutat néhány dolgot azok közül, amelyeket az sshd_config-ban lehet beállítani. Természetesen rengeteg további érték is módosítható itt. Néhányat közülük a következő hónapban bemutatok majd, az eddigiek megértése bőven elegendő az induláshoz (feltételezve, hogy a közvetlen cél a telnet és az ftp kiváltása).

Az scp használata titkosított fájlátvitelhez

Még egy témát tárgyalunk az e havi részben. Az scp parancs, legtöbb működésében azonos a jó öreg rcp eszközzel, amely könyvtárak és fájlok gépek közötti másolására szolgált. (Az igazság az, hogy az scp a rcp forráskódján alapul.) Ha esetleg nem ismerjük egyiket sem, ezek nem interaktív programok: mindegyiket parancssorból kell indítani, amelyben meg kell adni a másolandó anyag és a cél pontos elérési útját.
Emiatt a nem interaktív jelleg miatt az scp kevésbé felhasználóbarát, mint az ftp: akár tetszik, akár nem, az scp használatához bele kell nézzünk a leírásba (vagy egy olyan cikkbe, mint ez), és meg kell jegyezni néhány kapcsolót. Ezzel szemben, mint általában a többi parancssoros eszköz, az scp is sokkal inkább használható parancs-

Kapcsolódó címek

OpenSSH: ➲ http://www.openssh.com

zlib: ➲ ftp://ftp.freesoftware.com/pub/infozip/zlib

SSH információk
➲ http://www.inf.bme.hu/~mulzs/sshfaq/ssh-faq.html
➲ http://www.boran.com/security/ssh_stuff.html

SSH hivatkozások (linkek)
➲ http://www.db.toronto.edu/~djast/ssh.html
➲ http://linuxmafia.com/pub/linux/security/ssh-clients
➲ http://ssh.gatordog.com

SSH FTP tükrök
➲ ftp://ftp.wiretapped.net/pub/security/cryptography/ssh/
➲ ftp://ftp.zedz.net/pub/crypto/ssh/

fájlokban, mint amilyen egy interaktív eszköz lehetne. Az scp „gyorstalpaló” használatát valójában könnyű elsajátítani. Az alapvető formátum a következő:

```
scp [ kapcsolók ] forrásmeghatározás \
       célmeghatározás
```

ahol a forrás és a cél meghatározásai lehetnek egyszerű Unix-típusú fájl- és útvonalnevek (pl.: `./docs/hello.txt`, `/home/me/mydoc.txt` stb.), vagy egy gép és felhasználó azonosítására is alkalmas karaktersorozat:

```
felhasználó@távoli.gép.név:útvonal/fájlnév
```

Például: tegyük fel, hogy a „crueller” gépen dolgozunk, és a „recipe” fájlt szeretnénk átküldeni a „kolach” nevű gépen található saját könyvtárunkba. Továbbá feltételezzük, hogy mindkét gépen azonos felhasználónevet használunk. A folyamat valahogy a következőképpen néz ki:

```
crueller: > scp ./recipe kolach:~
mick@kolach's password: *******
recipe        100%   |****************>|    13226
00:01
crueller: >
```

A parancs begépelése után a jelszót kellett megadnunk, a rendszer feltételezte, hogy azonos nevet használunk a távoli gépen is. Az scp ezután átmásolta a fájlt, miközben egy hasznos kis folyamatjelző sáv mutatja az események előrehaladtát. Ennyi az egész!
Ha a cruelleren „mick”-ként dolgozom, a kolachra azonban mbauer-ként szeretném felmásolni a fájlt, a data/pastries könyvtárba, akkor parancssor a következőképpen alakul:

```
crueller: > scp ./recipe \
       mbauer@kolach:~/data/pastries/
```

Most változtassunk egy kicsit. Tételezzük fel, hogy a /etc/oven.conf állományt szeretnénk letölteni a kolachról:

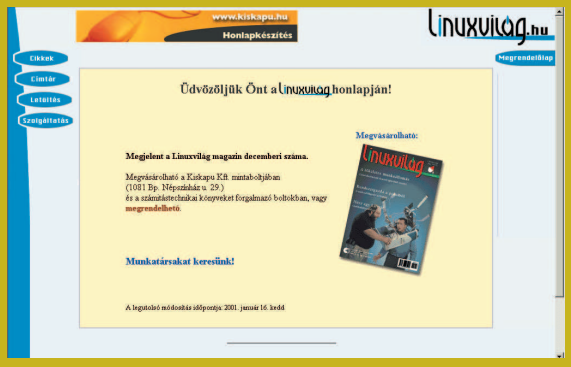
```
crueller: > scp mbauer@kolach:/etc/oven.conf
```

Mindenki érti a lényeget? Az egyetlen fontos dolog, amire emlékezni kell, hogy a forrás megelőzi a célt.
Habár a legtöbb felhasználó az ssh-t és az scp-t mindössze egyszerű bejelentkezésre és fájlátvitelre használja, ez csak a felszíne annak, amire az ssh képes. A következő hónapban azzal foglalkozunk, hogyan lehet az ssh-átvitelt még biztonságosabbá tenni RSA és DSA kulcsok segítségével, miképpen teszik lehetővé a „null-passparse” kulcsok azt, hogy az ssh parancsok parancsfájlokban is használhatók legyenek, hogyan lehet a bizonyítványokat a memóriában tárolni, hogy elkerüljük a nemkívánatos azonosító kéréseket, és azzal, mi módon lehet keresztülvezetni egyéb TCP-szolgáltatásokat egy titkosított ssh kapcsolaton.

Mick Bauer (mick@visi.com) alkalmazott biztonsági vezető az ENRGI hálózatmérnöki és tanácsadó cég minneapolisi részlegénél. 1995 óta Linux-rajongó, és 1997 óta vakbuzgó OpenBSD-s. Különös élvezetét leli abban, hogy ezeket az élvonalbeli operációs rendszereket rávegye arra, hogy elavult roncsokon fussanak. Mick szívesen vesz minden kérdést és hozzászólást.

Szakmai tanácsok

Az ssh-keygen programmal hozz létre egy kulcspárt, ezzel megoldhatod, hogy távoli rendszerre bejelentkezéskor ne kelljen jelszót megadni. A program futtatása után másold a .ssh/identity.pub fájlt a távoli gépre .ssh/authorized_keys néven.
Nem tudsz bejelentkezni ssh-val? Az ssh jelszót kér, amikor szerinted nem kéne ezt tennie? Használd a `-v` kapcsolót, így böngészheted a bőbeszédű tájékoztató- és hibaüzeneteket.
Állítsd be az RSYNC_RSH változót „ssh”-ra, így minden rsync forgalom titkosítva lesz.
Titkosíthatsz minden POP-forgalmat a géped és a levelező-kiszolgáló között, ha a fetchmailt ssh-n keresztül futtatod. Lásd a Linuxvilág honlapján (➲ www.linuxvilag.hu).



Ha a bash parancssorában meg akarsz találni egy korábban begépelt utasítást, nyomd le a CTRL-R billentyűkombinációt, majd írj be a keresett parancsból néhány betűt.
Ha le akarod másolni a webhelyed teljes tartalmát, de nincs parancssori hozzáférésed a kiszolgálóhoz, használd a `wget` programot a `--mirror` kapcsolóval.
Debian rendszereken futtathatod a következő cron feladatot éjszakánként: `apt-get update && apt-get -y --download-only dist-upgrade && apt-get autoclean`.
Ez tárolja a frissített csomagokat, így a telepítésük gyorsabb lesz.
A /usr fájlrendszert csak programok telepítése során kell írni. Mivel az fsck indításkor csak az írható fájlrendszereket ellenőrzi (befűzések száma stb.), ha a /usr-t csak olvashatóra állítod, meggyorsíthatod a rendszerindítást. Emellett, ha a rendszer összeomlik, a /usr-t nem kell majd fsck-zni.
A levelező-kiszolgálód nyílt levéltovábbító (open relay)? Használd a rlytest programot és győződj meg róla, hogy nem az.
Az inetd.conf fájlban tegyél megjegyzésbe mindent, amit nem használsz.
A .Xdefaults fájlban kikapcsolhatod a Netscape-ben a <blink> formázás megjelenítését a következő hozzáadásával:
`blinkingEnabled False`
A .netscape/preferences.js fájlból kikapcsolhatod a Netscape „Shop” gombját:
`user_pref("browser.chrome.disableMyShopping", true)`