Kilenc SSH-trükk

Titkosítsuk minden kapcsolatunkat SSH segítségével!

Az SSH az `rsh` és az `rlogin` távoli bejelentkezésre használható, de nem titkosított programok utódja. Az `rsh` és az `rlogin`, akárcsak a `telnet`, régi származással büszkélkedhet, mostanra azonban túlhaladottá és biztonsági szempontból elégtelenné vált. Ugyanakkor e programok meglepően nagyszámú, igen hasznos szolgáltatást gyűjtöttek össze a Unix-fejlesztések két évtizede alatt, így a legjobbak közülük az SSH-ba is bekerültek. Következzék az a 9 trükk, amit a leghasznosabbnak találtam. Hozzuk ki a legtöbbet az SSH programból! Ez a cikk is az OpenSSH-n alapul, ezért ha esetleg valamilyen más változatot használnak, a trükkök kipróbálása előtt ellenőrizze a leírást.

X11-átirányítás

Az SSH-n keresztül X-es kapcsolatainkat titkosíthatjuk. Ráadásul nemcsak a forgalom lesz titkosított, hanem a távoli gép `DISPLAY` környezeti változója is a megfelelő értéket veszi fel. Ha tehát a helyi gépünkön X-et futtatunk, távoli X-alkalmazásaink varázslatos módon a mi képernyőnkön jelennek majd meg.
Az X11-átirányítást az `ssh -X` *gépnév* paranccsal kapcsolhatjuk be. Az X11-átirányítást csak olyan gépekről használjuk, amelyeknek a rendszergazdájában megbízunk, ellenkező esetben az X11 alapú támadásokkal szemben sebezhetőkké válunk. Egy, az X11-átirányításon alapuló ügyes trükkel képeket jeleníthetünk meg az `xterm` ablakokban. Futtassuk a beépített képnézővel ellátott `w3m` böngészőt a távoli gépen, figyeljük meg a `w3m-img` Debian csomagot vagy a `w3m-imgdisplay` RPM-et. Ezek az X11-átirányítás segítségével `xterm` ablakunk felett egy keret nélküli ablakot nyitnak. Ha levelünket szöveges ügyfél segítségével, távolról olvassuk SSH-n keresztül, ezzel a módszerrel ugyanabban az `xterm` ablakban megnézhetjük a benne rejlő képességeket.

Beállításfájl

Az SSH a ***~/.ssh/config*** helyen keresi a felhasználói beállításfájlt. Ez például a következőképpen nézhet ki:

```
ForwardX11 yes
Protocol 2,1
```

A `ForwardX11 yes` ugyanazt jelenti, mintha a `-X` kapcsolót adtuk volna meg a parancssorban. A `Protocol` sor adja meg az SSH-nak, hogy előbb az SSH2 rendszert használja, és ha nem megy, csak akkor lépjen vissza a SSH1-hez. Amennyiben kizárólag SSH2-t használunk, töröljük a `,1` részt.
A beállításfájlunkban a `Host` utasítás alkalmazásával szakaszokat is kialakíthatunk, amelyek csak bizonyos távoli gépekhez való kapcsolódás során lépnek életbe. Hasznos beállításfájl-utasítás a `User`, amely a távoli gépen adja meg felhasználónevünket. Ha `ssh -l távolifelhasználó távoligép` vagy `ssh távolifelhasználó@távoligép` alakban gyakran jelentkezünk be valamelyik gépre, érdemes a beállításfájlunkba illeszteni a következő sorokat:

```
Host távoligép
ForwardX11 yes
User távolifelhasználó

Host *
ForwardX11 no
```

Mostantól csak a távoli gépet kell begépelnünk, ha a ***ForwardX11*** képesség bekapcsolásával távoli felhasználóként akarunk bejelentkezni. Minden más esetben a ***ForwardX11*** legyen kikapcsolva, ahogy azt fentebb ajánlottuk. A csillag minden gépnévvel egyezőnek tekintendő, még azokkal a nevekkel is, amelyeket korábban a `Host` szakaszokban felsoroltunk, de mindig csak az első találat utasítását használja a rendszer. Az egyedi `Host` szakaszokat ezért beállításfájlunkban az általános `Host` szakasz elé tegyük.
Az SSH rendszerbeállításfájllal is rendelkezik, amelynek neve ***/etc/ssh/ssh_config***. Az SSH a következő sorrendben gyűjti a beállításadatokat: parancssori kapcsolók, felhasználói beállításfájl, végül rendszer-beállításfájl. Az összes kapcsolót megtalálhatjuk a man `ssh_config` szövegében.

Gyorsítsunk: tömörítés és kódolások

Az SSH bármelyik kapcsolaton képes használni a `gzip` tömörítést. Az alapértelmezett tömörítés körülbelül 4×-es szövegtömörítésnek felel meg. A tömörítés különösen jól jöhet, ha például X-alkalmazást szeretnénk modemes vagy lassú hálózati kapcsolaton átirányítani. A tömörítést az `ssh -C` kapcsolóval vagy a ***Compression yes*** beállításfájlba illesztésével kapcsolhatjuk be.
Egy másik sebességgyorsító módosítás lehet titkosító kódolásunk megváltoztatása. Számos régi rendszer alapértelmezett kódolása a háromszoros DES (3DES), amely lassabb a Blowfish vagy AES kódolásnál. Az OpenSSH új változatai alapértelmezés szerint a Blowfisht használják. A kódolást az `ssh -c blowfish` kapcsolóval állíthatjuk át.
Beállításfájlunkban attól függően kell beállítanunk a kódolást, hogy SSH1 vagy SSH2 alapú rendszert használunk-e. Az SSH1 esetében a `Cipher blowfish`, míg SSH2 alatt a `Ciphers blowfish-cbc,aes128-cbc,3des-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc` alakot használjuk.

Kapuátirányítás

A kapuk a kiszolgáló különféle szolgáltatásait jelképező számok; például a 80-as kapu a HTTP-szolgáltatásnak, míg a 110-es kapu a POP3-nak felel meg. A szabványos kapuszámokhoz tartozó szolgáltatások listáját a ***/etc/services*** állományban találjuk. Az SSH gépünk bármelyik kapujáról képes adatokat átküldeni egy SSH-t futtató távoli gépre, s a forgalmat aztán a távoli kiszolgáló a másik gép tetszőleges kapujára irányíthatja. De miért akarnánk mi ilyesmit? Két okból: a titkosítás kedvéért és a csövezett (tunelling) kapcsolatért.

Titkosítás

Egy csomó alkalmazás használ olyan protokollt, amelyben a jelszavak és az adatok egyszerű szövegként utaznak a hálózaton. Ilyen a POP3, IMAP, SMTP és az NNTP. Az SSH átlátszó módon képes őket titkosítani. Tegyük fel, hogy a levelezőprogramunk normál esetben a ***mail.example.net*** POP3-kapujára szokott kapcsolódni (110-es kapu). Továbbá tegyük fel, hogy a ***mail.example.net*** gépre ugyan az SSH-val nem tudunk közvetlenül belépni, viszont van azonosítónk a ***shell.example.net***-en. Ilyenkor megmondhatjuk az SSH-nak, hogy titkosítsa a saját gépünk 9110-es kapuját (a számot tetszőlegesen választhatjuk meg), és a ***shell.example.net*** SSH-kiszolgálóját felhasználva küldje a ***mail.example.net*** 110-es kapujára:

```
ssh -L 9110:mail.example.net:
➥110 shell.example.net
```

Azaz a helyi 9110-es kaput továbbítsd a ***mail.example.net*** 110-es kapujára a ***shell.example.net***-re nyitott SSH-kapcsolaton keresztül. Ezután nincs más dolgunk, mint a levelezőprogramunknak megadni, hogy a helyi gépünk 9110-es kapujához kapcsolódjon. Itt az adat titkosítódik, az SSH-kapun keresztül átkerül a ***shell.example.net***-re, ahol visszakódolódik, végül a ***mail.example.net*** 110-es kapujára továbbítódik. Hasznos mellékhatásként a ***mail.example.net*** POP3-démonja azt fogja hinni, hogy minden forgalmat a ***shell.example.net***-től kap.

Csövezett kapcsolatok

Az SSH képes a tűzfalon átívelő hídként működni – védje az a tűzfal akár a saját gépünket, akár a távoli gépet, netalán mindkettőt. Az egyetlen dolog, amire szükségünk van, egy ismert SSH-kiszolgáló a tűzfal másik oldalán. Például igen sok DSL és kábelmodemes cég tiltja a levelek küldését saját gépünk 25-ös (SMTP) kapuján keresztül.
Következő példánkban kábelmodemes kapcsolatunkon keresztül levelet küldünk cégünk SMTP-kiszolgálójáról. A példában felhasználtuk a ***mail.example.net*** nevű SMTP-kiszolgálón meglévő azonosítónkat. Az SSH-parancs a következő lesz:

```
ssh -L 9025:mail.example.net:
➥25 mail.example.net
```

Ezt követően levélküldő programunknak mondjuk meg, hogy a helyi gép 9025-ös kapuját használja levélküldésre. Ez a gyakorlat igen hasonlónak tűnik az előzőhöz, csak most a ***mail.example.net*** 25-ös kapujára a ***mail.example.net***-en keresztül a helyi 9025-ös kapuról nyitottunk csatornát. A tűzfal szemszögéből mindössze szabályos SSH-adatok utaznak a szabványos SSH-kapun (22) keresztül, köztünk és a ***mail.example.net*** között.

Bináris adatok csövezése távoli héjprogramba

Az SSH-n keresztül használt csövezés távoli héjprogramokba teljesen átlátszó módon működik. Figyeljük meg a következőket:

```
cat myfile | ssh user@desktop lpr
```

```
tar -cf - source_dir | ssh user@desktop
➥'cat > dest.tar'
```

Az első példa a ***myfile*** állományt csövezi az a ***desktop*** nevű gépen futó `lpr` programba. A második példa egy tarfájlt hoz létre, és a tartalmát a terminálra írja (mivel a tarfájl nevének helyére egy kötőjelet írtunk), amit aztán átcsövezünk a ***desktop*** nevű gépre, és átirányítjuk egy fájlba.

Távoli héjparancsok futtatása

Az SSH segítségével nem feltétlenül kell felhasználói beavatkozással (interactive) héjprogramot nyitnunk, ha mindössze csak a távoli program kimenetére vagyunk kíváncsiak. Például a következőt is megtehetjük:

```
ssh felhasználó@gép w
```

A fenti parancs felhasználóként lefuttatja a `w` parancsot a ***gép*** nevű gépen, majd megjeleníti az eredményt. Használhatjuk a parancsok önműködővé tételére is:

```
perl -e 'foreach $i (1 .. 12) \
{print `ssh server$i "w"`}'
```

Ügyeljünk az SSH-parancs körüli fordított egyszeres idézőjelekre (aposztróf). A fenti sor Perl segítségével 12 alkalommal hívja meg az SSH-t, mindannyiszor más-más távoli gépen futtatva le a `w` parancsot, kezdve a ***server1***-től egészen a ***server12***-ig. De minden egyes SSH-kapcsolat létrejöttekor be kell majd gépelnünk a jelszavunkat. Ha viszont továbbolvasunk egy kicsit, megtudhatjuk, hogyan kerülhetjük el a biztonság feláldozása nélkül a jelszógépelést. (Ugyanezt sokkal egyszerűbben és erőforrás-takarékosabban az alábbi héjprogaramocska is tudja:

```
for i in `seq 1 12` ; do ssh server$i "w"; done
```

– ***Mészáros Gergely***, a fordító.

Utazzunk SSH Java Applettel

Seregnyi ember szaladgál lemezen magával hordozva a PuTTY-ot vagy valamilyen hasonló windowsos SSH-programot, hátha utazásai során nem biztonságos gépet kell használnia. Ez a módszer azonban csak akkor működik, ha lemezről lehet futtatni a programokat. Igaz, a PuTTY programot a honlapjáról is letölthetjük és elindíthatjuk.
Egy másik lehetséges megoldás, ha az előzőleg valamely honlapra felrakott SSH Java appletet böngészőből használjuk. Például kitűnő Java SSH-ügyfél a Mindterm, ennek nem üzleti célú felhasználása ingyenes. A programot a
➲ http://www.appgate.com/mindterm oldalon találjuk meg.

Összegzés

Az SSH beállítása a trükkök alkalmazása során néhány helyen könnyen félresiklik. Számos hibát felderíthetünk az `ssh -v` kapcsolóval és a kimenet figyelésével. Kétségtelen, hogy a trükkök egyike sem nélkülözhetetlen az SSH használatához, viszont könnyen kerülhetünk olyan helyzetbe, amikor örülünk, hogy mégis ismerjük őket. Nem árthat meg, ha kipróbálunk közülük néhányat!

Linux Journal 2003. augusztus, 112. szám

Daniel R. Allen (da@coder.com)
1995 óta elfogult Linux-rajongó. Egy kitcheneri programtanácsadó cég, a Prescient Code Solutions elnöke Ontario és Ithaca, területén.