Biztonságosabb SSH kéttényezős hitelesítéssel

A következőkben ismertetem, hogyan kell létrehozni egy USB-minimeghajtó és az ssh-agent segítségével kéttényezős hitelesítést a rendszergazda bejelentkezéséhez.

Nagy rajongója vagyok a kéttényezős hitelesítésnek, és amikor csak lehet, azt használom, mert a statikus jelszavak nem épp a legbiztonságosabbak. A hagyományos jelszavak általában könnyen megfejthetők az emberek természetes, bizalomra való hajlamának kihasználásával, a kijelzők szélére ragasztott sárga cetlik tanulmányozásával, kulcsnaplózással és az egyre gyorsabb számítógépekkel való kulcsfeltöréssel. Mióta felváltottam a kéttényezős hitelesítéssel, sokkal nyugodtabban alszom.
A kereskedelmi forgalomban kapható, hálózati alapú, kéttényezős hitelesítési rendszerek általában túl drágák és bonyolultak ahhoz, hogy azokat otthon vagy kis hálózatokban használjuk. De nem kell aggódni, van megoldás. A ***Linux*** már eleve tartalmazza a kéttényezős hitelesítéshez szükséges elemeket. A közismert, biztonságos kommunikációs eszköz, az ***OpenSSH***, minden olyan összetevőt tartalmaz, amely az otthoni számítógépekre, a kis hálózatokra és esetenként a nagyobb hálózatokra is megfelelő, gépalapú, kéttényezős hitelesítéshez szükséges. Ebben a cikkben megmutatom, hogy a hordozható eszközök, az ***OpenSSH*** nyilvános és titkos kulcsa, valamint a lenyűgöző ***ssh-agent*** kombinálásával hogyan lehet kéttényezős hitelesítést kialakítani úgy az egyszerű, mint a privilégizált felhasználók számára.

Első példa – Kéttényezős hitelesítés USB meghajtóval

Kezdjük az egyszerű (adminisztrátori jogokkal nem rendelkező) felhasználókkal. Ekkor használhatjuk az ***SSH*** jól ismert nyilvános hitelesítését, egy kis trükkel kiegészítve. Ahelyett, hogy a titkos kulcsot a felhasználó ***.ssh*** alkönyvtárában tárolnánk, az ***USB***-meghajtóra mentjük.
A példa kedvéért vegyük a nem privilégizált bob felhasználót, aki egy machine1 nevű, ***Fedora Core*** számítógépre jelentkezik be. A `machine2` nevű, távoli ***Linux*** gépre is ezzel a felhasználónévvel jelentkezünk be. Hozzuk létre az ehhez szükséges nyilvános és titkos kulcsokat:

```
ssh-keygen -t rsa -f key-rsa-
➥bob@machine2 -C key-rsa-
➥bob@machine2
```

A (lehetőség szerint minél hosszabb és véletlenszerűbb) generátorszöveg begépelése után, az ***ssh-agent*** létrehozza a kulcspárt, alapértelmezés szerint a felhasználó ***.ssh*** alkönyvtárában, jelen esetben a ***/home/bob/.ssh***-ban. A fájl neve tetszőleges lehet, de a példa kedvéért most olyan beszédes nevet választottam, amelyből egyetlen pillantással megállapítható a felhasználó és a gép neve – ennek jelentősége majd az ezt követő, több kulcsos példákban lesz érezhető. (Feltételezzük, hogy az ***USB*** meghajtó valamilyen linuxos, például ***ext3*** vagy ***vfat*** fájlrendszerrel formázott, de a kulcs fájljogosultságát minden beillesztésnél 400-ra kell állítani.)
Végezzük el az ***USB*** meghajtó beillesztését a fájlrendszerbe, miután azt ***/media/usbdisk*** (a példában ezt használjuk), ***/media/usbdisk1***, ***/media/disk*** vagy ***/media/disk-1*** néven kell látnunk. Az imént készített titkos kulcsot mozgassuk egy erre a célra szentelt könyvtárba és korlátozzuk a hozzáférési jogot a tulajdonosra:

```
mv key-rsa-bob@machine2
➥/media/usbdisk
chmod 400 /media/usbdisk/
➥key-rsa-bob@machine2
```

Ezután másoljuk be a nyilvános kulcsot (`key-rsa-bob@machine2.pub`) a `machine2`-n lévő ***/home/bob/.ssh/authorized_keys*** fájlba. Gondoskodjunk róla, hogy az ***authorized_keys*** fájlt csak a tulajdonosa olvashassa:

```
chmod 400 authorized_keys
```

Végül bejelentkezhetünk a `machine2` távoli gépre bob néven a létrehozott kulcspárral (az `ssh` program a `-i` kapcsolóból tudja, hogy melyik kulcsot kell használnia):

```
ssh -i /media/usbdisk/key-rsa-
➥bob@machine2 bob@machine2
```

A generátorszöveg begépelése után a `machine2`-n futó ***SSH***-kiszolgáló belépteti bobot. Szüntessük meg az ***USB***-meghajtó (vagy más hordozható eszköz) beillesztését a machine1 gépen, és a titkos kulcs máris biztonságban van. Ezzel megvalósítottuk a kéttényezős hitelesítést: az egyik tényező az ***USB***-meghajtó, amely a titkos kulcsot, a másik tényező pedig a fejünk, amely a generátorszöveget tárolja. A nyilvános kulcsú ***SSH***-hitelesítés bizonyára sokak számára ismert és mindennapos, a titkos kulcs áthelyezése egy hordozható eszközre pedig egyszerű módja a tényezők fizikai szétválasztásának.

Második példa – Kéttényezős adminisztrátori hitelesítés ssh-agent segédprogrammal

Az első példában bemutattuk, hogyan lehet egy távoli gépre biztonságosan bejelentkezni úgy, hogy a hitelesítési tényezők szétválasztását ***USB***-minimeghajtóval oldottuk meg. Ez a megoldás jól működik nem privilégizált felhasználóknál, szemben az adminisztrátorokkal. Meg kell találnunk a módját, hogy root felhasználóként is bejelentkezhessünk.

Az egyik lehetséges és kézenfekvő megoldás az, hogy a távoli gép ***SSH***-kiszolgálójában engedélyezzük a root belépését közvetlenül a hálózatból. Sem kulcs, sem jelszó nem utazik a hálózaton, mégis megsértjük azt az ősi rendszeradminisztrátori szabályt, miszerint ez nem megengedhető. Nincs tehát más út, mint megtalálni annak a módját, hogyan jelentkezhetünk be először normál felhasználóként, majd eztán adminisztrátorként. Ismét az ***OpenSSH*** húz ki minket a bajból. Most is a nyilvános és titkos kulcsokat használjuk, egy kis beállítási trükkel kiegészítve. Először is, állítsuk be a távoli ***SSH***-kiszolgálót úgy, hogy a root felhasználó a belső ***loopback*** interfészen keresztül bejelentkezhessen, a külső hálózatból azonban ne. Másodszor, az `ssh-agent` segédprogramot úgy állítsuk be, hogy a távoli gép a helyi gépen tárolt kulcs lekérdezésével hitelesíthesse a root felhasználót. A leírtak a következő lépésekkel valósíthatók meg:

1. Készítsünk egy nyilvános és titkos kulcspárt a root felhasználónak.
2. Másoljuk a nyilvános kulcsot a ***root authorized_users*** állományába a távoli gépen.
3. A helyi gépen futtassuk az `ssh-add` segédprogramot, hogy a titkos kulcs a gyorsítótárba kerüljön.
4. Jelentkezzünk be egyszerű felhasználóként a távoli gépre `ssh`-val, az első példában leírtak szerint, de ez alkalommal használjunk az átirányítást.
5. A távoli gépen jelentkezzünk be rootként a ***localhost*** interfészen, minek hatására a távoli ***SSH***-kiszolgáló lekérdezi a helyi gépet és hitelesíti a root felhasználót.

Az `ssh-agent` pontosan a kívánt funkcionalitást biztosítja: lehetővé teszi, hogy távoli ***SSH***-kiszolgálók a helyi gép gyorsítótárában lévő, megfejtett titkos kulcsok lekérdezésével hitelesítsenek felhasználókat. A kulcsok soha nincsenek továbbítva a két gép között – a titkos kulcsok a helyi munkaállomáshoz csatlakoztatott hordozható eszközön maradnak.

Az ***ssh-agent*** nagyon sokoldalú eszköz, de a beállítása nem feltétlenül triviális. Először is, meg kell fejteni a titkos kulcsot az `ssh-add` eszközzel, és át kell adni az `ssh-agent` programnak. Másodszor, meg kell mondanunk, hogy az `ssh-add` és az `ssh-agent` hogyan tud szót érteni egymással. Ez utóbbi egy ***socket***, melynek helye az `SSH_AUTH_SOCK` környezeti változóban található. Az `ssh-agent` alapértelmezés szerint önkényesen választ nevet a socketeknek, így az `SSH_AUTH_SOCK` korrekt beállítása nehézkes lehet.

A legtöbb disztribúció, így a ***Fedora Core*** is, automatikusan létrehozza az ***ssh-add*** és ***ssh-agent*** közötti kapcsolatokat a grafikus bejelentkezéskor (például ***GNOME*** és ***KDE***). Jelentkezzünk be parancssorban, majd adjuk ki a következő utasítást:

```
ssh-add -l
```

Ha az ***ssh-add*** és az ***ssh-agent*** tudnak egymással kommunikálni, akkor ennek vagy a nyilvános kulcsok listája, vagy a `"The agent has no identities"` üzenet lesz az eredménye, ha egy kulcs sem létezik.

Ha az ***ssh-agent*** bármi okból nem fut, vagy az `SSH_AUTH_SOCK` környezeti változó nincs megfelelően beállítva, a `"Could not open a connection to your authentication agent"` üzenetet kapjuk. Az utóbbi esetben hajtsuk végre a következő parancsot:

```
eval `ssh-agent`
```

Ez elindít egy `ssh-agent` példányt, és a környezeti változókat is megfelelően beállítja az aktuális parancsértelmezőben. Ezután az 1. példában megismert módon kulcspárt generálunk a rootnak:

```
ssh-keygen -t rsa -f key-rsa-
➥root@machine2 -C "key-rsa-
➥root@machine2"
```

Helyezzük a titkos kulcsot a hordozható eszközre, és adjunk a tulajdonosnak olvasási jogot, rajta kívül azonban senkinek semmit:

```
mv key-rsa-root@machine2
➥/media/usbdisk
chmod 400 /media/usbdisk/
➥key-rsa-root@machine2
```

Másoljuk a nyilvános kulcsot a `machine2` távoli gép */**root**/.**ssh**/**authorized_keys*** állományába. A következő paranccsal adjuk meg a root titkos kulcsát az `ssh-agent` segédprogramnak:

```
ssh-add -t 300 /media/usbdisk/
➥key-rsa-root@machine2
```

A generátorszöveg begépelése után, az ssh-agent az „Identity added: key-rsa-root@machine2 (key-rsa-root@machine2)” üzenettel tér vissza, ha a kulcsot megadta. (A `-t` kapcsoló szabályozza a kulcsok gyorsítótárban való elérhetőségének idejét, ami a jelen esetben 300 másodperc, azaz 5 perc. Ennek hiányában a kulcsok örökre elérhetők lesznek.)

Jelentkezzünk be a távoli számítógépre egyszerű felhasználóként:

```
ssh -A -i /media/usbdisk/
➥key-rsa-bob@machine2
```

A generátorszöveg megadása után, már be is léptünk a `machine2` távoli gépre. (Ez a parancs ugyanaz, mint amit az első példában láthattunk, de a `-A` kapcsolót használjuk átirányításra.)

A távoli gépen hajtsuk végre az `ssh-add -l`

parancsot, mire a root kulcsát kell megpillantanunk, amit az imént adtunk meg az `ssh-agent` segédprogramnak, például:

```
   2048 fa:5c:4b:73:88:26:
➥..:... /media/usbdisk/
➥key-rsa-root@machine2 (rsa)
```

A `su` paranccsal váltsunk át a root felhasználóra (a `machine2` gépen), és állítsuk be az ***SSH***-kiszolgálót úgy, hogy a ***loopback*** interfészen a root bejelentkezhessen. Ehhez a */**etc**/**ssh**/**sshd_config*** fájlban a következő módosítások szükségesek:

```
PermitRootLogin yes
AllowUsers bob@*
AllowUsers root@localhost.*
```

(Előfordulhat, hogy a ***loopback*** interfész címét numerikus formában kell megadni:

```
AllowUsers root@127.0.0.1.)
```

Mentsük el a beállításokat, és indítsuk újra az ***SSH***-kiszolgálót:

```
service sshd restart
```

A root felhasználóval jelentkezzünk ki, majd az ***OpenSSH*** segítségével ismét jelentkezzünk be:

```
ssh root@localhost
```

A `machine2` távoli gépen futó ***OpenSSH***-kiszolgáló most már fogadja az adminisztrátori bejelentkezéseket a ***loopback*** interfészen, de a külső hálózatból továbbra sem. A root hitelesítéséhez szükséges információkat a machine1 géppel tárgyalja meg. Figyeljük meg, hogy a root titkos kulcsa nem hagyta el a machine1 számítógépet!
Az ***OpenSSH***-t ily módon használva, hatékonyan kiválthatjuk a `su` (***switch user – felhasználóváltás***) és `sudo` segédprogramokat.
De még nem végeztünk teljesen! A biztonság tovább növelhető, ha a `su` parancs használatát kizárólag a helyben csatlakoztatott eszközökre korlátozzuk. Módosítsuk a */etc/pam.d/su* fájlt, hogy megakadályozzuk a `su` használatát a hálózaton keresztül:

```
auth        required
➥pam_securetty.so
```

Mostantól a `su` csak a parancssorból és a virtuális terminálokból használható.
Szüntessük meg az ***USB***-eszköz beillesztését és fizikai csatlakozását a géphez. Ahhoz, hogy most valaki megszerezze a titkos kulcsot, el kell lopnia az ***USB*** meghajtót. Még ha ez sikerülne is, akkor vagy a generátorszöveg megszerzése bizonyulna körülményesnek, vagy elképesztően nagy számítási teljesítményre volna szükség a kulcs feltöréséhez.

Harmadik példa – Feszítsük tovább a húrt

Mielőtt újdonsült rendszerünket kitesszük a „vadon” megpróbáltatásainak, nem árt, ha betömünk még egy biztonsági rést.
Ha az ***ssh-agent*** segédprogramot átirányítással használjuk, az a távoli ***SSH***-kiszolgáló számára lehetővé teszi a helyi számítógépen tárolt, titkos kulcs lekérdezését. Ha azonban így több számítógépre szeretnénk belépni, egy rosszindulatú felhasználó megbabrálhatja a kulcsokat az egyik gépen úgy, hogy beléphessen egy másikra. Ebben az esetben rosszabb a helyzet, mintha statikus jelszót használnánk.
A probléma bemutatásához bővítsük ki a példahálózatot a `machine3` géppel. Elkészítjük a kulcsokat bobnak és rootnak a `machine3`-ra, az első és második példa szerint, majd a root titkos kulcsát megadjuk a `machine1`-en futó `ssh-agent` segédprogramnak.
Ekkor `bob` `ssh`-val belép a `machine3`-ra, a `-A` átirányító kapcsolót is használva. Ha most végrehajtjuk az

```
ssh-add -l
```

parancsot, a `machine2` és `machine3` gépek nyilvános kulcsait látjuk:

```
2048 fa:5c:4b:73:88:...: ...
➥/media/usbdisk/key-rsa-
➥root@machine2 (RSA)
2048 26:b6:e3:99:c1:...: ...
➥/media/usbdisk/key-rsa-
➥root@machine3 (RSA)
```

A példában a `machine1`-en futó `ssh-agent` a másik két gép magánkulcsait a gyorsítótárba tölti. Ez az az `ssh-agent`, amely lehetővé teszi, hogy adminisztrátorként lépjünk be az utóbbi két gép bármelyikére, és amely sajnos azt is megengedi, hogy a `machine2` gépről valaki rootként belépjen a `machine3`-ra, és fordítva. Ez nincs rendjén.
Szerencsére, ez a hiányosság kiküszöbölhető a `-c` kapcsolóval. A biztonság fokozható, ha minden egyes távoli számítógép adminisztrátori kulcsának tárolásához külön `ssh-agent` példányt futtatunk. A -c kapcsoló jelzi az `ssh-agent` számára, hogy a felhasználónak jóvá kell hagynia minden olyan kulcs felhasználását, amely a gyorsítótárban van. A távoli gépek számára dedikált `ssh-agent` példányok a még ismeretlen biztonsági hibák esetén is garantálják, hogy egy gép kulcsa teljesen független marad a többiétől.
Az `ssh-add` jóváhagyási funkciójának használata egyszerű: minden kulcs megadásánál használjuk a `-c` kapcsolót. Tegyünk egy próbát. Indítsunk egyszerre két `ssh-agent` segédprogramot a `machine1`-en, előre definiált socketek megadásával:

```
ssh-add -c /media/usbdisk/
➥key-rsa-root@machine2
ssh-add -c /media/usbdisk/
➥key-rsa-root@machine3
```

Mostantól kezdve jóvá kell hagyni a kulcs felhasználását, ha `ssh`-val be akarunk lépni a `machine2` vagy `machine3` gépre.
Arra is lehetőség van, hogy külön `ssh-agent` példányokkal tároljuk az egyes kulcsokat. Ehhez indítsunk két `ssh-agent` segédprogramot a machine1-en, előre definiált socketek megadásával:

```
ssh-agent -a /tmp/ssh-agent-
➥root@machine2
ssh-agent -a /tmp/ssh-agent-
➥root@machine3
```

Ismét hangsúlyozom, hogy az általam használt elnevezések önkényesek, ugyanakkor a példák szempontjából rendkívül beszédesek. Állítsuk be a környezeti változót és adjuk meg a kulcsot a `machine2`-nek:

```
export SSH_AUTH_SOCK=/tmp/
➥ssh-agent-root@machine2
ssh-add -c /media/usbdisk/
➥key-rsa-root@machine2
```

Ismételjük meg a fenti lépéseket a `machine3`-ra is, a `machine2`-re vonatkozó indexek értelemszerű cseréjével:

```
export SSH_AUTH_SOCK=/tmp/
➥ssh-agent-root@machine3
ssh-add -c /media/usbdisk/
➥key-rsa-root@machine3
```

Most jelentkezzünk be a `machine3`-ra (mivel az `SSH_AUTH_SOCK` környezeti változót legutóbb ennek a gépnek az `ssh-agent` programjára állítottuk):

ssh-add

Az `ssh-add` segédprogrammal tilthatjuk vagy jóváhagyathatjuk a titkos kulcsok használatát. A tiltás bekapcsolása a `-x`, kikapcsolása a `-X` kapcsolóval lehetséges. A kulcs tiltásánál megadott jelszót kell használnunk a tiltás kikapcsolásához. A `-c` kapcsoló esetén az `ssh-add` minden alkalommal figyelmeztet, amikor az `ssh-agent` segédprogramhoz kérés érkezik egy kulcs használatára. A figyelmeztető üzenet az `ssh-agent` segédprogramot futtató gépen jelenik meg, és hatékonyan akadályozza meg a jogosulatlan felhasználókat abban, hogy hozzáférjenek a kulcsainkhoz.

Kettő vagy 2.X tényező

A helyben tárolt ***SSH***-kulcsokat és a hozzájuk tartozó generátorszövegeket egyesek külön tényezőkként kezelik, nem minden alap nélkül. Mégis, sokkal megnyugtatóbbnak érzem, ha a kulcsok tárolása fizikailag is elkülönül a számítógéptől. A kulcsok hordozható eszközön tárolása csökkenti annak lehetőségét, hogy valaki megszerezze és feltörje azokat. Fontos annak megértése, hogy a kulcsok tárolása például ***USB***-minimeghajtón nem küszöböli ki teljesen annak lehetőségét, hogy egy rosszindulatú felhasználó megkaparintsa azokat. Amíg a hordozható eszköz beillesztve van, a kulcsok veszélyben lehetnek, így a helyi gépen is megfelelő óvintézkedéseket kell tenni annak érdekében, hogy a bejelentkezés a távoli gépekre biztonságos legyen. Használjunk erős jelszót a helyi (parancssori) belépéshez, legyenek telepítve a legfrissebb biztonsági javítócsomagok stb. Összességében tehát nyilvános kulcsú hitelesítés használatával jobban járunk, mint a statikus jelszavakkal, de csak addig, amíg a munkaállomáshoz kellően nehéz hozzáférni. Aztán hogy ki mennyire szereti magát biztonságban érezni, azt már az egyéni beállítottság és persze az üldözési mánia súlyosságának foka határozza meg.

```
ssh -A -i /media/usbdisk/key-
➥rsa-bob@machine2 bob@machine3
```

Hajtsuk végre az

```
ssh-add -l
```

parancsot, hogy lássuk, mely kulcsok elérhetők a machine1-en. Örömmel tapasztaljuk, hogy csak a `machine3` rootjának kulcsa jelenik meg. Jelentkezzünk ki a `machine3`-ról, állítsuk át a környezeti változót a `machine2` `ssh-agent` programjára, és jelentkezzünk be a `machine2`-re:

```
export SSH_AUTH_SOCK=/tmp/ssh-
➥agent-root@machine2
ssh -A -i /media/usbdisk/key-
➥rsa-bob@machine2 bob@machine2
```

Ismét ellenőrizzük az elérhető kulcsokat:

```
ssh-add -l
```

A listák csak és kizárólag az adott gép adminisztrátori kulcsát fedik fel előttünk. Az előző példában használt egyetlen `ssh-agent` esetén mind a `machine2`, mind a `machine3` gép kulcsát láttuk volna.
Ha minden olyan géphez, amelyre be akarunk jelentkezni, külön `ssh-agent` példányt futtatunk, az kicsit több munkával jár.
Az `SSH_AUTH_SOCK` környezeti változó átállítása, mikor másik gépre akarunk bejelentkezni, finoman szólva macerás. A folyamat egyszerűsítéséhez, írtam egy ***tfssh*** (***two-factor ssh – kéttényezős ssh***) névre keresztelt parancsfájlt, melynek szintaxisa a következő

```
tfssh [username@]host [keydir]
```

A parancsfájl (letölthető a ***LinuxJournal FTP***-helyéről: ***ftp.ssc.com/pub/lj/listings/issue152/8957.tgz***) szükség szerint elindítja az ssh-agent példányokat, beállítja a környezeti változót, a root kulcsot megadja az `ssh-agent` példánynak és a megadott felhasználónévvel (username) bejelentkezik a távoli gépre (host). Megadható továbbá, hogy a kulcsokat melyik (keydir) könyvtárban keresse, és mennyi ideig tartsa azokat a gyorsítótárban.

Összefoglalás

A statikus jelszavak több bajt okozhatnak, mint amennyit használnak. Gátat kell szabnunk ezek további használatának, és terjesztenünk kell a kéttényezős hitelesítést, melynek eszközkészletét a sokoldalú ***OpenSSH*** biztosíthatja. A nyilvános és titkos kulcsok, az átirányítási funkció és a hordozható eszközök használatával az ***OpenSSH*** gondoskodik a kulcsok biztonságáról, mely egyszerű, olcsó és hatékony eszközt jelent a gép alapú, kéttényezős hitelesítési rendszer készítéséhez.

A kulcsok tárolása

A kulcsok tulajdonképpen bármely hordozható eszközön tárolhatók. A példákban ***USB***-minimeghajtót használtunk, mert kicsi, könnyű és egyszerűen kezelhető. Bátran használjunk újraírható ***CD-ROM*** vagy ***DVD***-lemezt, de akár floppyt is, ha úgy tetszik.

Az ismertetett kéttényezős hitelesítési rendszer beállítása és használata némi munkát kíván, amelyet azonban bőségesen kompenzál az elért biztonsági szint. A `tfssh` parancsfájllal még ettől is megszabadulhatunk, vagyis a kéttényezős hitelesítést minden előnyével élvezhetjük, a járulékos teendők mellőzésével.

Linux Journal 2006., 152. szám

Paul Serry
Több mint 20 éve UNIX és Linux rendszeradminisztrátor. Számos Linux-könyv, pl. a Network Linux Toolkit és a Knoppix for Dummies szerzője. John „maddog" Hall társszerzőjeként több Red Hat Linux for Dummies és Fedora Core for Dummies könyvet is írt. Az új mexikói Albuquerque-ben él, elektronikus levélcíme pgsery@swcp.com.