SSH – informacje ogólne

Wstęp

Korzystanie z FTP ma wady:

1. Połączenie nie jest szyfrowane i można podsłuchać hasło.
2. Domyślnie korzystanie z konta roota jest wyłączone (opcja *RootLogin* w pliku *proftpd.conf* lub wpis w */etc/ftpusers*).
3. Nie obsługuje wielu poleceń systemowych.

SSH szyfruje połączenie, i to jeszcze przed autoryzacją hasłem (hasło nie jest nigdy przesyłane czystym tekstem). Zaleca się jednak używać autoryzacji w oparciu o klucze i tzw. „przepustki", a nie o hasła systemowe (co chroni przed atakiem *brute-force*). W pliku */etc/ssh/sshd_config* ustawiamy opcje:

PasswordAuthentication *no*
PubkeyAuthentication *yes*

Należy pamiętać, iż pliki konfiguracyjne użytkownika (*$HOME/.ssh/config*) nadpisują ustawienia zawarte w */etc/ssh/ssh_config*, ale jeszcze większy priorytet mają opcje w linii komend. Oczywiście nie wszystkie opcje można nadpisać.

Tworzenie klucza ogólnego

W czasie pierwszego uruchomienia serwera SSH (*/etc/init.d/sshd start*) generuje on następujące klucze (dla całego systemu):

Hostkey:
/etc/ssh/ssh_host_key (-rw-------) - identyfikacja serwera
/etc/ssh/ssh_host_key.pub (-rw-r--r--) - klucz publiczny serwera
DSA-Hostkey:
/etc/ssh/ssh_host_dsa_key (-rw-------) - identyfikacja serwera DSA
/etc/ssh/ssh_host_dsa_key.pub (-rw-r--r--) - klucz publiczny serwera DSA
RSA-Hostkey:
/etc/ssh/ssh_host_rsa_key (-rw-------) - identyfikacja serwera RSA
/etc/ssh/ssh_host_rsa_key.pub (-rw-r--r--) - klucz publiczny serwera RSA

Opis kluczy:

RSA1 - algorytm stworzony w 1978, nazwa pochodzi od pierwszych liter nazwisk twórców; dotychczas udało się złamać 500-bitowy klucz; wszystkie klucze 700-bitowe i większe uważane są za bezpieczne; używa protokołu SSH1.

RSA2 - używa protokołu SSH2.

DSA (*Digital Signature Algorithm*) - algorytm asymetryczny, amerykański standard narodowy uważany przez niektórych (*Schneier*) za bardziej bezpieczny, używa protokołu SSH2.

Tworzenie kluczy indywidualnych

Po skonfigurowaniu serwera ssh (*/etc/ssh/sshd_config*) oraz klienta ssh (*/etc/ssh/ssh_config*), należy utworzyć klucze indywidualne (każdy użytkownik tworzy je na swoim koncie):

ssh-keygen -t dsa

W czasie tworzenia kluczy pada pytanie o tzw. „przepustkę” (*passphrase*). Nie jest ona tożsama z hasłem systemowym i powinna różnić się od niego. Nie chodzi w tym przypadku o hasło do zdalnego konta, ale o dostęp do klucza. Akceptowane są spacje. Możliwe jest także ustanowienie “pustej przepustki” na potrzeby skryptów inicjowanych przez *crona* (zob. *BatchMode* w konfiguracji klienta *ssh*). Przepustki mogą być cache’owane (tzw. *Agent SSH*, polecenie *ssh-add*) i wtedy nie trzeba podawać ich przy każdej operacji kopiowania (nie zaleca się jednak korzystania z tej możliwości).

Klucze zostaną utworzone w:

$HOME/.ssh/id_dsa (rw-------) - klucz prywatny (nie udostępniamy nikomu!)

$HOME/.ssh/id_dsa.pub (rw-r--r--) - klucz publiczny (zamek publiczny)

Klucz publiczny należy przenieść (skopiować/wysłać pendrive) na komputer, z którym zamierzamy się łączyć, na konto o tej samej nazwie. Nazwę klucza zamieniamy przy tym na:

$HOME_Zdalne/.ssh/authorized_keys2 (r--------)

Jeśli zamierzamy dodać kilka kluczy-drzwi (wygenerowanych na różnych komputerach) wtedy dodajemy je po prostu za pomocą komendy:

cat id_dsa.pub >> *$HOME_Zdalne/.ssh/authorized_keys2* (r--------)

W pliku *$HOME/.ssh/authorized_keys* można umieścić linie z opcjami, np.:

command=”JakiśSkrypt” (wykonywany po połączeniu użytkownika)

from=”nazwa.dozwolonego.klienta”

Aby zmienić *passphrase*, należy:

ssh-gen -p

Ustanowienie połączenia

Wykaz możliwych komend (uwaga: nazwy komputerów muszą być zawarte w */etc/hosts*, a po każdej zmianie nazwy naszego hosta, np. z host na host.domena, należy zmienić opcję *AllowUsers* w pliku konfiguracyjnym serwera):

$ **ssh IP_komputera**

$ **ssh -l użytkownik nazwaKomputera** [lub:]

$ **ssh użytkownik@nazwaKomputera** (logowanie się jako inny użytkownik, przy autoryzacji kluczami, nie będzie możliwe, gdyż nie mamy dostępu do klucza umieszczonego na innym koncie)

ssh -f użytkownik@nazwaKomputera aplikacja (uruchamia zdalną aplikację w tle, co umożliwia zamknięcie terminala)

Prawidłowe połączenie za pomocą klucza wygląda tak:

```
krzysiu@nazwaKomputera:~> ssh krzychu@IP.zdalnego.komputera
The authenticity of host ‘IP.zdalnego.komputera’ can’t be established.
ECDSA key fingerprint is SHA256:jbg+vTFTFTb46ygvy7c99gVGvgVtvgVgvh45463.
Are you sure you want to continue connecting (yes/no)? Yes
Warning: Permanently added ‘IP.zdalnego.komputera’ (ECDSA) to the list of known hosts.
Enter passphrase for key ‘/home/krzysiu/.ssh/id_dsa’:
Last login: Sat Feb 17 14:40:28 2019 from 192.168.17.63
Have a lot of fun...
krzychu@nazwaZdalnegoKomputera:~>
```

Transmisja plików

scp /home/user/plik.txt user@zdalnyKomputer:/home/użytkownik/

Możliwe jest także wydanie komendy odwrotnej, powodującej skopiowanie pliku ze zdalnego komputera na nasz dysk lokalny.

Innym narzędziem jest *sftp* (posiada takie funkcje jak ftp, ale jest szyfrowane).

Zdalne uruchamianie programów:

ssh użytkownik@zdalny.komputer "echo testowyList | mail adresat@domena.pl"

Tunelowanie

Tunelowanie to łączenie naszego lokalnego portu z portem zdalnym. W efekcie użytkownicy obu sieci lokalnych mają wrażenie, że pracują w jednej sieci lokalnej bez pośrednictwa Internetu. Taki tunel może służyć do transportu mniej bezpiecznych protokołów, np. POP3. Najpierw łączymy się z naszym portem lokalnym (wybieramy pomiędzy 1024-65535):

ssh -L 10110:localhost:110 zdalnyKomputer *(lokalny port to 10110, a zdalny to 110, czyli POP3)*

Aby odebrać pocztę przez ten tunel, trzeba ją odebrać łącząc się po prostu na lokalny port 10110, a nie na zdalny 110. Tunel jest zamykany wraz z zamknięciem shella. Wykaz nasłuchujących portów można sprawdzić poleceniem:

netstat -l --tcp -p | grep ssh

Istnieje możliwość obejścia zabezpieczeń firewalla, np. tunelując usługę ftp przez port przeznaczony dla www.

Klient PuTTY

PuTTY to darmowy klient SSH dla systemu Windows. Należy zainstalować (a w zasadzie skopiować, gdyż są to wersje nieinstalacyjne, *standalone*):

- *PuTTY*;
- *PuTTYgen* (*putty-tools* w Linux)*;*

Ustawienia połączenia w PuTTY:

- *Session* → *numerIPserwera*, port *22*;
- *Connection / Data* → *nazwaZdalnegoUżytkownika*;
- *Session* / *Saved Sessions* → *naszaNazwaSesji* → *Save*;

Podczas pierwszego połączenia z serwerem, pojawi się „fałszywy komunikat", że serwer został zmieniony lub jego klucz (należy kliknąć „*Yes*"). Domyślnie *PuTTY* łączy się z serwerem przy użyciu hasła. Aby przejść na weryfikację za pomocą kluczy, musimy je najpierw utworzyć w programie *PuTTYgen* (SSH-2 DSA, 1024) lub użyć kluczy stworzonych w *openssh*. Szum generujemy poruszając dynamicznie myszką, określamy też „*passphrase*". Klucz publiczny kopiujemy tradycyjnym sposobem na serwer. Klucz prywatny wczytujemy w PuTTY:

- *Connection / SSH / Auth / Browse*...;
- *Session* → *Save*.

Przy logowaniu pojawi się wtedy komunikat:

```
login as: krzysiu
Authenticating with public key „dsa-key-20190217" from agent
Last login: Sat Feb 17 13:34:50 2019 from console
Have a lot of fun...
krzysiu@nazwaKomputera:~>
```

lub też tak (w przypadku importowanego klucza z *openssh*):

```
login as: krzysiu
Using username „krzysiu".
Authenticating with public key „imported-openssh-key"
```

```
Passphrase for key „imported-openssh-key”:
Last login: Sat Feb 17 13:34:50 2019 from IP.Zdalnego.Komputera
Have a lot of fun...
krzysiu@nazwaKomputera:~>
```

Konwersja kluczy prywatnych

Zdarza się, że klucze stworzone przez *PuTTYgen* są niekompatybilne z serwerem *openssh*.

Windows → Linux

Aby przekształcić klucz prywatny z formatu *PuTTY* (*.ppk) na format *OpenSSH* (*.pem) należy:

puttygen id_dsa.ppk -O private-openssh -o id_dsa *(w Linuksie)*

lub

Uruchom *PuTTYgen (w Windows)* / *Load private key* / menu *Conversions* / *Export OpenSSH key* → *id_dsa*

Linux → Windows

Aby przekształcić klucz prywatny z formatu *OpenSSH* (*.pem) na format *PuTTY* (*.ppk) należy:

Uruchom *PuTTYgen* (w Windows) / *File* / *Load private key: id_dsa (wybrać klucz prywatny OpenSSH)* / *Save the generated key* → *Save private key*

Konwersja kluczy publicznych

Windows → Linux

Uruchom *PuTTYgen (w Windows)* / *Load: Private Key* / Skopiować zawartość: *Public key for pasting into OpenSSH authorized_keys file* → *$HOME/.ssh/authorized_keys*

lub

puttygen kluczPublicznyPuTTY.pub -L

lub ręcznie usunąć nagłówek i stopkę klucza, a jego treść umieścić w jednej linii:

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "rsa-key-20171124"
ssh-dss
AAAAB3NzaC1yc2EAAAABJQAAAQEAvxNOXr/TEkxYzOABGOZ4pxIngxsIwRvizMBunaVcd+Bj6W
tGY1cdR8o2LJGoRP0nuXUKMV65+ncY+ZbHB+ngiCZ6SausF1PA5aeDkMEzyRSl+HJjb5ZB14lLd
qbvnK4NwuTmVflg60ORqE8OAFuZdOeH6eEBISbTo4rgeBJoaOjeo2Okr0s5+g10KTZjzhRZIJHe4F
cUS9FuC74Wvnr5rAhn2knvpOPPxH4FKjVMe4H+D1yPfCCeVnaHEpRt6LoSaD4CDSQu6riYjmIq0
U4cueYxYq/6M6+CsEqTaWnpZSC3eh3shnF0nku5EGQGgZEVIaYWtffkHFoCWb6bTSuw= dsa-
key-20190218
---- END SSH2 PUBLIC KEY ----
```

Linux → Windows

ssh-keygen -i -f *kluczPublicznyPuTTY.pub* **>** *nowyKluczPublicznyOpenSSH.pub*

Problemy

Jeśli pojawia się komunikat “*Permission denied (publickey,keyboard-interactive)*”, serwer SSHd nie rozpoznaje nazw NetBIOS we wpisie *AllowUsers*. Wydaje się, że nie korzysta on z pliku */etc/hosts*, ale z pliku */etc/nsswitch.conf* (na maszynie była kiedyś instalacja ypbind) i gdy go

brakuje lub gdy są nieodpowiednie wpisy, to nie rozpoznaje nazw hostów. (Niestety, nie wiem dlaczego tak jest i gdzie to zmienia się).

Inne darmowe klienty SSH dla Windows

- *SSH Secure Shell* (*http://osusls.osu.edu*)
- *Tera Term* (*http://www.zip.com.au/~roca/ttssh.html*)
- *Shell in a Box* (projekt *Google*, klient łączy się przez przeglądarkę internetową, serwer jest linuksowy: pakiet *„shellinabox”* w *Gentoo; strona projektu: https://code.google.com/archive/p/shellinabox/*)

Ostatnia aktualizacja: 20 września 2022.