A1_ssh

ssh - Secure Shell zur sicheren remote-Administration

Eine der bekanntesten Anwendungen für asymmetrische Schlüsselverfahren ist die Secure Shell, z.B. openssh.

Abb1. Zur Erinnerung: Bei symmetrischen Verfahren (oben) dient ein Key sowohl zum Ver- als auch zum Entschlüsseln. Die asymmetrische Varianten (unten) benutzen dafür zwei verschiedene Schlüssel, einen privaten und einen öffentlichen.

Jeder Host benötigt ein eigenes Schlüsselpaar aus privatem und öffentlichem Schlüssel, das meist schon bei der Installation der Software generiert wird. Bei dem einleitenden Handshake authentifiziert sich der Zielhost (ssh-Server) mit seinem privaten Schlüssel.

Der Client generiert anschließend einen Sitzungsschlüssel, verschlüsselt ihn mit dem öffentlichen Key des Servers und sendet ihn an den Server. Damit öffnet er einen sicheren Tunnel (siehe Abb.2), über den anschließend die Datenkommunikation abläuft. Die Benutzer-Authentifizierung erfolgt danach über einen Passwortdialog oder ebenfalls mittels asymmetrischer Kryptographie. Beide Versionen laufen über die gesicherte Verbindung des ssh-Tunnels und sind somit für Außenstehende nicht entzifferbar. Nach der Authentifizierung stellt der Server dem Client eine Shell zur Verfügung. Das ssh-Protokoll hat noch viele weitere nützliche Funktionen, z.b. scp zum sicheren Datentransport (siehe ganz unten).

Abb.2 Die Secure Shell verwendet asymmetrische Kryptographie, um den Server gegen den Client und den Benutzer gegen den Server zu authentifizieren. Beide Partner brauchen dazu ein eigenes Schlüsselpaar.

ssh - Secure Shell zur sicheren remote-Administration

Anmelden mit Benutzernamen und Passwort auf srvYZ

userXY@pcXY:~$ ssh [userYZ@]srvYZ

- Beachte ! Benutzerangabe [userYZ@] nur bei abweichendem
- Benutzer auf dem Server.
- Bei der erstmaligen Anmeldung wird ein 'fingerprint' des öffentlichen Schlüssels des Remote-PC/Servers errechnet und bei Bestätigung* (Authentizität) wird der Public-Key des Servers in die .ssh/known_hosts geschrieben.

Anmelden mittels Public-Key-Verfahren

1. Schlüssel erzeugen:

- Syntax: ssh-keygen [-f KEY_FILE] [-C COMMENT] [-t TYP (rsa|dsa)] [-b KEY_LENGTH (1024|2048)]

userXY@pcXY:~$ ssh-keygen

- Standard ist ein RSA-Schlüsselpaar mit 2048 bit Schlüssellänge
- Eine Passphrase (Passwortsatz) dient der Sicherheit, oft im Intranet aus Komfortgründen weggelassen (für automatisierte
- Jobs)
- Das Schlüsselpaar liegt in ~/.ssh.
- Der private Schlüssel in der Datei id_rsa/id_dsa (default) Der public-key in der Datei id_rsa.pub/id_dsa.pub (default)

2. öffentlichen Schlüssel auf remote-Server ablegen

userXY@pcXY:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub [userYZ]@srvYZ

- Kopieren des public-keys automatisch auf den remote-Server in die Datei 'authorized_keys' mit dem Befehl 'ssh-copy-id'

zwei kompliziertere Alternativen dazu:

userXY@pcXY :~$ scp .ssh/id_rsa.pub [userYZ@]srvYZ:.ssh/authorized_keys

- Kopieren des ersten public-keys auf den remote-Server und umbenennen in die Datei authorized_keys

userYZ@srvYZ:~/.ssh$ cat .../usbstick/id_rsa.pub >> authorized_keys

- public-key über einen sicheren Weg (usbstick) direkt auf dem Server an die Datei authorized_keys angehängen (neben anderen public-Keys)

3. Anmelden nun ohne Passwort möglich

userXY@pcXY:~$ ssh [userYZ@]srvYZ

Tip: bei Fehlermeldung 'Agent admitted failure to sign using the key' neuen privatkey mit dem Befehl 'ssh-add' hinzufügen!

ssh - Secure Shell zur sicheren remote-Administration

weitere Beispiel für die Secure Shell:

Kopieren eines Verzeichnisses: z.B. ~/linux nach srvYZ:

userXY@pcXY:~$ scp -rv ~/linux [userYZ@]srvYZ:

Synchronisieren von Daten mit rsync über eine ssh-Verbindung:

userXY@pcXY:~$ rsync -av ~~/linux/ [userYZ@]srvYZ:~~/linux/

Absetzen eines remote-Kommandos:

userXY@pcXY:~$ ssh user@pcXY 'ls -l'

Port-Forwarding:

Port-Weiterleitung eines lokalen Ports an einen entfernten

Zielport durch eine verschlüsselte ssh-Verbindung (Tunnel)

ssh user@ssh-srv -L 1025:localhost:80

oder (Verbindung dauerhaft):

ssh -f -N -L 1025:localhost:80 user@ssh-srv

ftp via ssh → sftp:

userXY@pcXY:~$ sftp user@pcXY

Connected to pcXY

sftp>

für Konsolenpuristen!

userXY@pcXY:~$ cat .ssh/id_rsa.pub | ssh [userYZ@]srvYZ 'cat >> .ssh/authorized_keys'

Der Inhalt des public-keys ' id_rsa.pub ' wird in einen Puffer|Pipe gesteckt

und nach dem Anmelden auf dem Server ' ssh userYZ@srvYZ ' an das Schlüsselbund angehängt, mit dem übergebenen Befehl ' 'cat >> .ssh/authorized_keys''

- Fingerprint des ECDSA-pubkey auf dem Server anzeigen und zur Bestätigung vergleichen:

```
userYZ@srvYZ:~$ ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
```

Unterlagen:

- wget ftp://ftp.itm.bfw/pub/unterricht/linux/netzwerk/09_sicherheit/03_openssh/

Internet:

- http://wiki.ubuntuusers.de/SSH
- http://www.youtube.com/watch?v=Hxsl-jj2Bq0

- http://www.lug-ottobrunn.de/wiki/Remote_Desktop_mit_X2GO

Quellen

- Dokument: A1_ssh
- ID: 138

Revision #2
Created 29 May 2026 08:01:10 by Admin
Updated 29 May 2026 08:03:26 by Admin