Secure Shell - SSH unter Redhat / CentOS 7

SSH (oder ausgeschrieben **Secure SHell**) ist ein Protokoll, welches eine sichere Kommunikation zwischen zwei Systemen mittels einer Client/Server Architektur ermöglicht.

Die Benutzer können sich damit auch bei einem entfernten Host-System anmelden und dies anschliessend mit den Berechtigungen des lokal eingeloggten Users verwalten.

Im Gegensatz zu anderen Remote-Kommunikationsprotokollen wie FTP oder Telnet, verschlüsselt SSH die Anmeldung. Auf diese Weise können Eindringlinge auch keine unverschlüsselten Passwörter erkennen.

SSH wurde als Ersatz für ältere, weniger sichere Terminalanwendungen, wie **rsh** entwickelt. Das Programm **scp** ersetzt ältere Programme wie **rcp**, die zum Kopieren von Dateien zwischen Hosts verwendet wurden.

OpenSSH - Programmsuite

Die für die **Secure-Shell** benötigten Pakete werden im optimal Fall bereits bei der Erstinstallation erfolgreich ins System installiert. Bei Redhat teilen sich die einzelnen Programme der Programmsuite auf folgende Pakete auf:

- **openssh** : Die OpenSSH-Implementierung der SSH Protokoll-Versionen 2 (und 1)
- **openssh-clients** : Die OpenSSH-Client-Anwendungen
- **openssh-server** : Der OpenSSH-Server Daemon
- **openssh-askpass** : Passphrase-Dialog für OpenSSH und X

Paketinhalt von openssh

Mittels **rpm -qil** kann überprüft werden, welche Programme, Konfigurationsdateien und Dokumentationen beim Paket openssh installiert wurden.

```
# rpm -qil openssh
```

```
Name        : openssh
Version     : 6.6.1p1
Release     : 25.el7_2
Architecture: x86_64
Install Date: Mon 06 Feb 2017 07:14:52 PM CET
Group       : Applications/Internet
```

```
Size        : 1450050
/etc/ssh
/etc/ssh/moduli
/usr/bin/ssh-keygen
/usr/libexec/openssh
/usr/libexec/openssh/ctr-cavstest
/usr/libexec/openssh/ssh-keysign
...
```

Das gleiche könnten wir nun auch für das **openssh-clients** Paket, den **openssh-server** oder den **openssh-askpass** durchführen. So erkennen man, welche openssh Teile von welchen Paket zur Verfügung gestellt werden.

SSH für Key-Authentifizierung Konfigurieren und härten

Sobald man einen **Fernzugriff** auf seine **Linux-Systeme**, auf Konsolenebene nutzen möchte, kommt man um **SSH** nicht drum herum. Ist der SSH-Server-Dienst einmal im System aktiviert und für die Nutzer freigegeben, kann man sich von überall aus dem LAN mit seinem User via Putty einloggen. Gibt man nun noch den Port via Router frei und schaltet das Portforwarding für SSH ein, so ist der Server auch von unterwegs erreichbar.

Doch nun zum Problem; Passwörter sind heutzutage nicht mehr genug sicher, um auch wirklich einen Server mit sensiblen Daten zu schützen. Brute-Force-Attacken nehmen vermehrt zu und werden auch durch immer länger werdende Passwortlisten ständig effektiver. *Doch wie kann man sich nun dagegen schützen?* Ganz einfach! Anstatt wie üblich über ein Passwort zu authentifizieren, besteht die Möglichkeit, auf eine weitere Alternative zurückzugreifen: **SSH-Key Authentification**. Diese auf dem RSA-Verschlüsselungsprotokoll basierenden Schlüssel bestehen aus einem **Public-Key** und einem **Private-Key**.

Der Public-Key wird auf allen Linux Systemen hinterlegt, während der Private-Key NUR auf dem Client bleibt, welcher später eine Verbindung zu den Server aufbauen soll. Den Privat-Key kann man zusätzlich noch mit einer Passphrase schützen! *Wenn man sich also anschliessend mit einem der Systeme per Key-Authentifizierung verbindet, wird man nicht mehr nach dem Passwort des Systems gefragt, sondern nach der Passphrase des Private-Keys.*

ACHTUNG: Zum Schluss, wird das **SSH Login via Passwort** zum erhöhen der System-Sicherheit noch komplett **deaktiviert**. Dies sollte jedoch **erst nach erfolgreichem testen der Key-Authentifizierung gemacht werden**.

Erstellen des SSH-Key-Pairs (Public/Private Key)

Der erste Schritt ist auch gleich der kürzeste. Mit einem einzigen Befehl lässt sich das Pair erstellen.

```
# ssh-keygen -t rsa -b 4096
```

Anschliessend wird man noch gefragt, wo der Schlüssel gespeichert werden soll. Mit „Enter“ wird er am Standartort hinterlegt (/home/rebermi/.ssh/id_rsa). Gleich danach, wird man noch gefragt, ob man für deinen Privat-Key ein Passphrase (Passwort) erstellen möchte. Dies ist zwar *optional*, wird aber

dringend empfohlen! So werden die Systeme zusätzlich geschützt, falls jemals jemand an den Key kommen sollte. Nach Eingabe und Bestätigung des Passwortes werden die Keys generiert und das Random RSA Pic angezeigt.

```
[rebermi@vsat1t ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rebermi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rebermi/.ssh/id_rsa.
Your public key has been saved in /home/rebermi/.ssh/id_rsa.pub.
The key fingerprint is:
dc:81:69:d7:cb:7b:7b:c5:99:57:e1:96:62:53:89:23 rebermi@vsat1t.pnet.ch
The key's randomart image is:
+--[ RSA 4096]----+
|              . .|
|         o E o + |
|        + o o + o|
|       o o o = +.|
|        S . + +.+|
|             . o+|
|            . . o|
|             . ..|
|              .. |
+-----------------+
```

Der **Puplic-Key** wurde nun unter **/home/user/.ssh/id_rsa.pub** gespeichert. Dieser wird anschliessend wie im nächsten Kapitel beschrieben auf andere Server verteilt, damit mit dem gleichen Key auf mehreren Servern Authentifiziert werden kann. Der **Privat-Key**, liegt unter **/home/user/.shh/id_rsa**. **ACHTUNG: Denn Privat-Key sollte nur auf Clients liegen, auf die nur ich Zugriff habe!**

Server mit neu erstelltem Public-Key versorgen

Jetzt müssen, wie oben schon erwähnt nur noch die anderen Server bzw. Systeme, *(auf die zugreifen werden soll)*, von dem neu erstellten Public-Key erfahren. Dafür gibt es einen ganz einfachen Befehl:

```
# ssh-copy-id root@IP-oder-Name-des-Zielsystems
```

Nach ausführen des Befehls, wird nach dem Passwort gefragt. Dies ist das Passwort des Nutzers, in dem Falle Root, auf dem Zielsystem. Hintergrund des Ganzen ist folgender: *Dein Client öffnet eine SSH-Verbindung zum Zielsystem und fügt deinen Public-Key in die dortige ~/.ssh/authorized_keys – Datei. Falls diese noch nicht existiert, wird diese erstellt.* Man kann das Ganze natürlich auch manuell machen:

```
# cat ~/.ssh/id_rsa.pub | ssh root@IP-oder-Name-des-Zielsystems "mkdir -p
~/.ssh && cat >>  ~/.ssh/authorized_keys"
```

Ab jetzt sollte man ohne das Passwort des Servers bzw. des Zielsystems eine Verbindung über SSH herstellen können. Wenn ein Passwort für den Privat-Key erstellt wurde, wird nun stattdessen dieser verlangt.

Da man das normale Login mit Passwort jetzt nicht mehr benötigt, kann man wie schon erwähnt, den Zugriff über SSH via Passwort auch komplett verbieten. In dem Fall empfiehlt sich jedoch eine Sicherheitskopie des Privat-Keys anzulegen. Denn wenn dem Client mal etwas passiert, und man keinen physikalischen Zugriff mehr auf das Systeme hat, sperrt man sich so selber aus.

Zugriff per Passwort Login deaktivieren

Um den Zugriff via Passwort über SSH zu deaktivieren, verbindet man sich zuerst mit dem betreffenden System und bearbeite die sshd_config:

```
# vim /etc/ssh/sshd_config
```

In diesem Konfigurationsfile sucht man anschliessend den Eintrag “**PasswordAuthentication**” und ändert den Eintrag auf **PasswordAuthentication no**.

Wichtig: *Auch wenn dieses auskommentiert ist, muss explizit der Wert auf No gesetzt werden und die auskommentiert entfernt werden, damit es funktioniert!*

```
# Change to "no" to disable tunnelled clear text passwords

PasswordAuthentication no
..
```

Zum Abschluss, muss nun noch der SSH-Service neu gestartet werden:

```
# systemctl restart sshd
```

Schlusswort

`Die Nutzung eines SSH-Keys ist einem Passwort immer vorzuziehen und ist um einiges sicherer.` Allerdings ist es sehr schwer, an entfernte Systeme heranzukommen, sollte der Privat-Key einmal verloren gehen. Daher sollte dieser unbedingt gesichert werden.

Wird Windows als Client verwendet, kann automatisch beim booten den Private-Key mit dem Putty Tool “pageant.exe” welches sich nach der Installation auf einem 64 Bit System unter folgendem Pfad befindet:“C:\Program Files (x86)\SSHTOOLS\PuTTY\pageant.exe” verwendet werden.

Zum automatisieren, wird dann eine Verknüpfung von jenem Tool in den Autostart von Windows gemacht, mit Angabe des SSH Private-Keys under den Verknüpfungspunkt “Ziel:”

Beispiel: *“C:\Program Files (x86)\SSHTOOLS\PuTTY\pageant.exe” C:\id_rsa.ppk*

WICHTIG: Zur Verwendung und laden des Private-Keys unter Windows, muss der Key “id_rsa” noch mit puttygen.exe (Auch im Putty Verzeichnis) geladen werden und im Putty

eigenen id_rsa.ppk Format abgespeichert werden! Dieses id_rsa.ppk kann dann mit dem oberen Beispiel, als Verknüpfung im Autostart, beim Boot automatisch geladen werden.

Equip SSH Logins with MOTD or SSH-Banner Messages

Vor - und nach einem LOGIN auf unser System können wir dem Nutzer noch individuelle Informationen anzeigen und/oder mitgeben.

Hierfür gibt es zwei unterschiedliche Wege; also einmal das **issue.net** file oder das **motd** file.

- **motd** : Zeigt eine Banner Message gleich nach dem erfolgreichen Login an.
- **issue.net** : Zeigt ein Banner noch vor der Password Login-prompt.

motd - Display SSH Warning Message - After User Login

Das Akronym **motd** steht für message of the day. Diese Datei, welche vom Login-Programm benutzt wird, befindet sich im Konfigurationsverzeichnis unter `/etc/motd` und gibt nach einem erfolgreichem Login - aber noch vor dem Start der jeweiligen Login-Shell - eine Meldung aus.

Zur Information für etwaige Neugierige und Remoteuser, oder beides, je nach Betrachtungsweise, wollen wir hier nun noch ein paar Zusatzinformation ausgeben.

```
# vim /etc/motd
```

```
##########################################################################
##
 #
#
 #                 This is the admin server of Michael Reber.
#
 #
#
 #                                ATTENTION:
#
 #
#
 #             Unauthorized access to this system is prohibited !
#
 #
#
 #    This system is actively monitored and all connections may be logged.
#
 #         By accessing this system, you consent to this monitoring.
#
```

```
 #
#
##########################################################################
##
```

War die Datei zuvor noch nicht vorhanden, so passen wir die Berechtigungen noch wie folgt an:

```
# chown root:root /etc/motd
# chmod 644 /etc/motd
```

issue.net - Display SSH Warning Message - Before User Login

Mit Hilfe dieser Datei `/etc/issue.net` können ebenfalls Meldungen am Bildschirm ausgegeben werden - diese Informationen erscheinen dann jedoch vor dem Login-Prompt - sobald man sich via ssh anmelden möchte. Hierzu legen wir einfach die Datei `/etc/issue.net` mit dem editor unserer Wahl an.

```
# vim /etc/issue.net
```

```
##########################################################################
##
 #
#
 #                      This is a private home server.
#
 #
#
 #             Unauthorized access to this system is prohibited !
#
 #
#
 #    This system is actively monitored and all connections may be logged.
#
 #         By accessing this system, you consent to this monitoring.
#
 #
#
##########################################################################
##
```

Die Berechtigungen passen wir dann wie folgt an:

```
# chown root:root /etc/issue.net
# chmod 644 /etc/issue.net
```

Abschliessend ergänzen wir noch in der `/etc/ssh/sshd_config` die BANNER Konfiguration.

```
# vim /etc/ssh/sshd_config
```

Add the Banner-Statement to sshd_config, with the correct path to: **/etc/issue.net** file and save it!

```
..
Banner /etc/issue.net   #(you can use any path you want)
```

Nun fehlt nur noch der Restart des SSH Daemon:

```
# systemctl restart sshd
```

Last update: **2019/02/20 15:47**