SSH HOWTO

Den Secure Shell Dämon sicher einrichten

Patrick Fey

<debianhowto (at) nachtarbeiter (dot) net>

2004-08-06

Versionsgeschichte

Version 1.0	2004-08-06	Patrick
Initial release		
Version 1.0.1	2004-09-02	Patrick
Minor fixes, scp example in 4-2		

Der Dienst **ssh** kann ein hohes Sicherheitsrisiko für ein System darstellen, wenn er nicht richtig eingerichtet ist. Das Verbieten des Logins durch den Benutzer **root** und Authentifizierung über öffentliche Schlüssel kann die Sicherheit eines ferngewarteten Systems exponentiell erhöhen.

Inhaltsverzeichnis

Allgemeine Hinweise

Copyright and License

Disclaimer

Feedback

Feedback is most certainly welcome for this document. Send your additions, comments and criticisms to the following email address : `<debianhowto (at) nachtarbeiter (dot) net>`.

Translations

There are no translations of this document availlable at present.

Einleitung

SSH ermöglicht es, sich von einem entfernten Computer aus an einer Shell auf dem Zielcomputer anzumelden und daran so zu arbeiten, als sitze man direkt vor dem Zielsystem. Dabei werden im Gegensatz zu anderen ähnlichen Diensten wie Telnet alle Daten verschlüsselt übertragen.

SSH wurde 1995 von Tatu Ylönen an der Technischen Universität Helsinki als Reaktion auf ein Sicherheitsproblem mit dem Telnet-Protokoll entwickelt. Seine Software war zunächst frei verfügbar. Später schränkte er die Lizenzbedingungen zunehmend ein und gründete die Firma SSH Communication Security Ltd. Basierend auf den freien Quellen der ursprünglichen Software, jedoch unter Verwendung der Krypto-Bibliothek der freien SSL-Implementierung OpenSSL, wurde von der OpenBSD Gruppe später die heute am weitesten verbreitete freie Version von SSH (OpenSSH [http://www.openssh.org/]) entwickelt.

Obwohl an sich relativ sicher, kann der Dienst SSH doch ein hohes Sicherheitsrisiko in einem System darstellen, wenn er nicht richtig eingerichtet ist - bietet er doch dem potentiellen Angreifer ein bequemes Werkzeug zur Fernsteuerung des Servers.

So ist zum Beispiel die Version 1 des SSH Protokolls auf Grund des schwachen kryptographischen Verfahrens, das es zur Integritätssicherung der versendeten Pakete verwendet, sehr anfällig für einen sog. Mann in der Mitte (engl. "Man in the middle") Angriff. Ein Patch, der gegen diese Sicherheitslücke entwickelt wurde, erwies sich ebenfalls als so anfällig, dass es Angreifern ermöglicht wurde, beliebige Befehle als **root** auf dem Zielsystem auszuführen. Übrigens verwendet [http://slashdot.org/article.pl?sid=03/05/18/1416213&mode=nested&threshold=4&commentsort=3] Filmheldin Trinity diesen sog. SSH1 CRC32 Exploit im Film "The Matrix Reloaded" erfolgreich, um einen Computer zu knacken.

Es sollte also nur noch Version 2 des SSH Protokolls verwendet werden, insbesondere weil auch heute noch verstärkt nach Servern gescannt wird, die diese Sicherheitslücke aufweisen.

In beiden Protokollversionen kennt SSH verschiedenste Wege, um Benutzer zu authentifizieren. Leider sind die Verfahren nicht alle gleich sicher. Generell gilt es, so viele Möglichkeiten wie möglich zu deaktivieren. Am besten beschränkt man sich auf ein oder zwei Verfahren.

Das Rhost-Verfahren ist extrem unsicher und in der Grundeinstellung auch ausgeschaltet. Es handelt sich hier um eine sog. Trusted-Host-Authentifizierung, bei der sich in einer speziellen Datei eingetragene Clients nicht mehr zu authentifizieren brauchen. Das ganze Modell vertraut darauf, dass IP-Adressen und Ports korrekt sind. Diese Annahme ist heutzutage aber leider recht leichtsinnig.

Bei einem ähnlichen Verfahren, das aber standardmäßig auch abgeschaltet ist, authentifizieren sich Client und Server gegenseitig, indem Sie Ihre öffentlichen Hostschlüssel austauschen. Wer unbedingt Trusted- Host-Authentifizierung benötigt, sollte diese Methode verwenden.

Standardmäßig wird die sog. kennwortbasierte Methode verwendet. Hierbei wird das Kennwort des Benutzers am System zur Authentifizierung verwendet und verschlüsselt zum Server übertragen. Obwohl das Kennwort verschlüsselt übertragen wird, ist es aber prinzipiell möglich, durch einen sog. SSH Keystroke Timing Angriff von den verschlüsselten Daten indirekt auf das Kennwort zu schließen, ohne die übertragenen Daten selbst zu entschlüsseln.

Die ausgefeilteste und sicherste Variante der Authentifizierung von Benutzern ist das sog. PublicKey-Verfahren. Hierbei legt ein Benutzer seinen öffentlichen Schlüssel in seinem Home-Verzeichnis ab und authentifiziert sich mit dem passenden privaten Schlüssel auf seinem eigenen Rechner. Der private Schlüssel kann und sollte zusätzlich mit einem Kennwort vor Missbrauch geschützt werden. Dadurch wird verhindert, dass bei einem erfolgreichen Angriff auf den Rechner des Benutzers Nutzen aus dem privaten Schlüssel gezogen werden und so auch in das Zielsystem eingedrungen werden kann.

Um eine möglichst hohe Sicherheit zu gewährleisten, werden wir also die Verwendung der Version 1 des SSH Protokolls verbieten und die kennwortbasierte Authentifizierung abschalten. Stattdessen werden wir das PublicKey-Verfahren zur Authentifizierung verwenden.

Login von root verbieten

Der erste Schritt zu einem gut konfigurierten SSH-Server ist jedoch, den direkten Login von **root** zu verbieten. Dazu legen wir zunächst einen neuen lokalen Benutzer an, mit dem wir uns in Zukunft über SSH einloggen werden. Von diesem Benutzerzugang aus können wir ggf. mittels des Befehls **su root** werden . Ein potentieller Angreifer muss so zunächst zwei Kennworte herausfinden, bevor er vollen Zugriff zum Server erlangen kann. Außerdem wird das "Austesten" des **root**-Kennworts erschwert.

Wir legen also zunächst einen neuen Benutzer mit einem Benutzernamen und einem Kennwort unserer Wahl an.

```
adduser benutzername
Enter new UNIX password: geheimeskennwort
Retype new UNIX password: geheimeskennwort
Enter the new value or press return for the default Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [y/n] y
```

Nun sollten wir uns einmal mit dem neuen Benutzer einloggen und testen, ob es auch geht, bevor wir uns selbst aussperren. Haben wir uns erfolgreich eingeloggt, werden wir **root** und bearbeiten die Konfigurationsdatei von **sshd**.

```
su root
Password: supergeheimesrootkennwort
vi /etc/ssh/sshd_config
```

In der Datei suchen wir die Zeile

```
PermitRootLogin yes
```

und ändern die Zeile in

```
#PermitRootLogin yes
PermitRootLogin no
```

ab. Dann speichern und schließen wir die Datei. Jetzt müssen wir noch den **ssh**-Dämon neu starten und dann sollten wir uns nicht mehr als **root** in das System einloggen können.

Warnung

Der **ssh** Dämon erhält bestehende Verbindungen auch dann aufrecht, wenn er neu gestartet wurde. Nachdem Sie die Konfigurationsdatei geändert haben, sollten Sie sicherheitshalber Ihr "altes" Terminalfenster geöffnet lassen, während Sie sich nach dem Neustart von **sshd** wieder einloggen. Auf diese Weise können Sie Loginfehler auf Grund von z.B. Tippfehlern in der Konfigurationsdatei schneller beheben. Wenn der Login nach der Konfigurationsänderung erfolgreich war, können Sie Ihr "altes" Terminalfenster schließen.

```
/etc/init.d/sshd restart
```

Nun machen wir noch mal den Test. Nachdem wir festgestellt haben, dass wir uns wirklich nicht mehr als **root** einloggen können, loggen wir uns in Zukunft als normaler Benutzer ein und werden dann per **su** zum Superuser.

Authentifizierung über öffentliche Schlüssel

Bevor wir zur Einrichtung des Servers für dieses Authentifizierungs-Verfahren kommen, müssen wir zunächst auf dem jew. von uns genutzten lokalen Rechner ein eigenes Schlüsselpaar erstellen. Im Folgenden habe ich für Sie eine Anleitung für Windows und für Linux verfasst. Ich gehe dabei davon aus, dass Sie unter Windows Putty [http://www.chiark.greenend.org.uk/~sgtatham/putty/] und unter Linux OpenSSH [http://openssh.org/] verwenden. Sollten Sie ein anderes Programm verwenden, konsultieren Sie bitte die Dokumentation zu Ihrem jew. Programm.

Erstellung eines Schlüsselpaares unter Windows

Wir werden nun ein Schlüsselpaar auf unserem lokalen Rechner erstellen. Öffnen Sie zunächst aus dem Startmenü das Programm **PuttyGen**. Im aufspringenden Fenster wählen Sie unten unter "Parameters" als Schlüsseltyp "SSH2 DSA" aus. Klicken Sie anschließend auf die Schaltfläche "Generate" um einen solches Schlüsselpaar zu erzeugen.

Bewegen Sie nun Ihre Maus über die leere graue Fläche, bis der blaue Balken 100% erreicht hat. **PuttyGen** nutzt Ihre Mausbewegungen dazu, einen Zufälligkeitsfaktor für Ihren Schlüssel zu erstellen.

Geben Sie als "Key Comment" nun einen passenden Kommentar für den Schlüssel (zum Beispiel "username@servername") und als "Passphrase" ein sicheres Kennwort für Ihren privaten Schlüssel an. Speichern Sie anschließend den privaten und öffentlichen Schlüssel an einem bestimmten Ort in Ihrem Dateisystem, schließen Sie jedoch das eigentliche Fenster noch nicht.

Loggen Sie sich nun auf Ihrem System ein ohne **root** zu werden. Erstellen Sie nun im Home-Verzeichnis Ihres Benutzers ein neues Verzeichnis `.ssh` und dort eine neue Datei mit dem Namen `authorized_keys`.

```
mkdir $HOME/.ssh
vi $HOME/.ssh/authorized_keys
```

Anschließend kopieren Sie den Inhalt aus dem Fenster im Bereich "Public key for pasting ..." in die eben neu erstellte Datei. Achten Sie dabei darauf, dass der gesamte Inhalt in der ersten Zeile der Datei steht. Setzen Sie nun die entsprechenden Rechte für die Datei.

```
chmod g-w $HOME $HOME/.ssh $HOME/.ssh/authorized_keys
```

Öffnen Sie nun den Putty-Terminal und laden Sie wie gewohnt die gespeicherte Sitzung für Ihren Server in das Konfigurations-Fenster. Wechseln Sie nun im rechten Optionen-Baum zu "Connection" > "SSH" und wählen Sie unter "Preferred SSH Version" den Eintrag "2 only" aus. Wechseln Sie dann im Options-Baum zu "Connection" > "SSH" > "Auth" und wählen Sie unter "Private key file for authentication" die entsprechende private Schlüsseldatei aus. Wechseln Sie im Options-Baum nun zurück zu "Session" und speichern Sie die neuen Werte in Ihrer vorhandenen Sitzung ab. Klicken Sie anschließend wie gewohnt "Open" um einen Testlauf zu starten.

Sie werden nun von Putty gefragt, mit welchem Benutzer Sie sich einloggen wollen. Geben Sie hier den Namen des Benutzers an, in dessen Home-Verzeichnis Sie den öffentlichen Schlüssel gespeichert haben. Sie werden nun nach dem Kennwort gefragt, das Sie für den privaten Schlüssel vergeben haben. Dieses Kennwort wird benötigt, damit Putty den privaten Schlüssel zum Aufbau der Verbindung nutzen kann. Es wird nicht über das Internet an den Server versandt. Dieses Kennwort schüzt Ihren privaten Schlüssel vor Missbrauch, falls Ihr lokales System kompromittiert werden sollte.

Wichtig

Achten Sie bitte genau auf die Hinweise beim Login. Klappt die Authentifizierung mit dem öffentlichen Schlüssel nicht, gibt Putty lediglich die Fehlermeldung "Server refused our private key!" aus und versucht anschließend direkt einen normalen Login durchzuführen. Weil das normaler Weise klappt, ist der Fehler zunächst nicht offensichtlich.

Nachdem wir erfolgreich eine Verbindung mit dem Server aufbauen konnten, können wir nun die kennwortbasierte Authentifizierung deaktivieren (Dies erledigen wir gleich im nächsten Schritt!). Somit ist nur noch eine Authentifizierung mit unserem privaten Schlüssel möglich.

Tipp

In diesem Zusammenhang sei auch erwähnt, dass es evt. gar nicht so falsch ist, ein Backup des privaten Schlüssels an einem sicheren Ort aufzubewahren.

Erstellung eines Schlüsselpaares unter Linux

Öffnen Sie eine Kommandozeile und erstellen Sie ein neues DSA-Schlüsselpaar.

```
ssh-keygen -d
```

Sie werden aufgefordert einen Verzeichnisnamen anzugeben, in dem das Schlüsselpaar erstellt werden soll. Drücken Sie hier einfach die Eingabetaste um den Standardwert zu übernehmen. Anschließend werden Sie aufgefordert, ein Kennwort einzugeben und zu bestätigen. Dieses wird verwendet, um Ihren privaten Schlüssel vor Missbrauch durch Fremde zu schützen.

In dem von Ihnen angegebenen Verzeichnis finden Sie nun Ihr neues Schlüsselpaar. Der private Schlüssel heißt standardmäßig `id_dsa` und der öffentliche Schlüssel liegt in der Datei `id_dsa.pub`. Übertragen Sie die Datei mit dem öffentlichen Schlüssel in Ihr Home-Verzeichnis auf dem Server. Nutzen Sie dazu nach Möglichkeit eine verschlüsselte Übertragungstechnik wie SCP oder SFTP. Denken Sie daran, die IP-Adresse und den Benutzernamen anzupassen, wenn Sie das folgende Beispiel verwenden.

```
scp $HOME/.ssh/id_dsa.pub serverbenutzername@999.999.999.211:id_dsa.pub
```

Jetzt können wir den Schlüssel an die Datei `authorized_keys` im Verzeichnis mit den SSH-Einstellungen anhängen und dem Server so unseren öffentlichen Schlüssel als autorisierten Schlüssel übergeben. Öffnen Sie eine Komandozeile und loggen Sie sich auf Ihrem Server ein, ohne **root** zu werden.

```
mkdir $HOME/.ssh
touch $HOME/.ssh/authorized_keys
cat $HOME/id_dsa.pub >> $HOME/.ssh/authorized_keys
```

Vergeben Sie nun noch die benötigten Dateirechte.

```
chmod g-w $HOME $HOME/.ssh $HOME/.ssh/authorized_keys
```

Nun sind die Vorbereitungen auf dem Server abgeschlossen und Sie können sich theoretisch verbinden. Legen Sie in Ihrem Home-Verzeichnis auf dem Client im Ordner mit den SSH-Einstellungen eine Datei mit den notwendigen Einstellungen an und machen Sie dann einen Verbindungsversuch. Denken Sie daran die IP-Adresse und den Benutzernamen anzupassen.

```
mv $HOME/.ssh/id_dsa $HOME/.ssh/999.999.999.211-dsa
chmod g-r $HOME/.ssh/999.999.999.211-dsa
ssh 999.999.999.211 -p 22 -l server-benutzername -i $HOME/.ssh/999.999.999.211-dsa
```

Wenn Ihre Verbindung geklappt hat, können Sie diese Einstellungen in Ihre eigene angepasste Client-Konfigurationsdatei (`~/.ssh/config`) eintragen.

```
vi $HOME/.ssh/config

# Einstellungen für 999.999.999.211
Host meinserver
Hostname 999.999.999.211
IdentityFile2 /home/client-benutzername/.ssh/999.999.999.211-dsa
Port 22
PreferredAuthentications publickey
Protocol 2
User server-benutzername
```

Jetzt können Sie sich von der Kommandozeile ganz einfach mit Ihrem Server verbinden.

```
ssh meinserver
```

Nachdem wir erfolgreich eine Verbindung mit dem Server aufbauen konnten, können wir nun die kennwortbasierte Authentifizierung deaktivieren (Dies erledigen wir gleich im nächsten Schritt!). Somit ist nur noch eine Authentifizierung mit unserem privaten Schlüssel möglich.

Tipp

In diesem Zusammenhang sei auch erwähnt, dass es evt. gar nicht so falsch ist, ein Backup des privaten Schlüssels an einem sicheren Ort aufzubewahren.

Laufzeitkonfiguration von ssh

Wir gehen nun Zeile für Zeile die Konfigurationsdatei des **ssh** Servers durch und setzen dabei verschiedene sicherheitsrelevante Einstellungen.

```
vi /etc/ssh/sshd_config

# sshd configuration file
# see sshd_config(5) for details
```

Wir verwenden einen andern Port als den Standard-Port, um Standardscans von sog. "Skript-Kiddies" auf Port 22 ins Leere laufen zu lassen. Welchen Port Sie letztendlich verwenden ist im Grunde egal. Sie sollten jedoch einen Port oberhalb der Ports für sehr bekannte Dienste (oberhalb von 1024) verwenden. Vergessen Sie nicht, später die Konfiguration Ihres Termial-Clients (z.B. Putty) entsprechend anzupassen.

```
# listen on port 2222
Port 2200
```

Wenn auf Ihren Server mehrere IP-Adressen konnektiert sind, sollten Sie den SSH-Dienst auf eine der IP-Adressen beschränken. Evt. verwenden Sie eine IP ausschließlich zu Administrationszwecken. Dadurch wird das Risiko eines Angriffs auf den Server über SSH drastisch reduziert. Wenn auf Ihren Server lediglich eine IP-Adresse konnektiert ist, können Sie diese Option überspringen.

```
# listen on primary ip only
ListenAddress 999.999.999.211
```

Auf Grund der Sicherheitslücken in Version 1 des SSH-Protokolls verwenden wir ausschließlich Version 2.

```
# use ssh2 only
Protocol 2

# private host keys for ssh2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

# use privilege separation
UsePrivilegeSeparation yes
```

Wir deaktivieren die kennwortbasierte Authentifizierung und verwenden nur noch die auf dem Austausch des öffentlichen Schlüssels basierende Authentifizierung, die wir im letzten Abschnitt eingerichtet haben.

```
# disable password authentication
PasswordAuthentication no
PAMAuthenticationViaKbdInt no
```

Wir geben dem Benutzer 30 Sekunden Zeit, um sich einzuloggen. Erfolgt in dieser Zeit kein erfolgreicher Login, so wird die Verbindung abgebrochen. Es kann sich immer nur ein Benutzer gleichzeitig einloggen.

```
# allow one login at a time
# authentication may last 30 secs
LoginGraceTime 30
```

```
MaxStartups 1
```

Wir verbieten direkte Logins des **root** Benutzers und erlauben nur Logins des Benutzers, den wir zuvor erstellt haben.

```
# root logins are not permitted
PermitRootLogin no
AllowUsers yourusernamehere
StrictModes yes
```

Abgesehen von dem "PublicKey" Verfahren deaktivieren wir alle Möglichkeiten der Authentifizierung, da wir nur dieses Verfahren nutzen. Sicherheitshalber verbieten wir die Benutzung leerer Kennwörter.

```
# enable pubkey authentication
PubkeyAuthentication yes

# disable all other authentications
RSAAuthentication no
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreUserKnownHosts yes
ChallengeResponseAuthentication no
KerberosAuthentication no

# disable empty passwords
PermitEmptyPasswords no

PrintMotd no
KeepAlive yes

# enable sftp subsystem Subsystem
sftp /usr/lib/sftp-server

# logging
SyslogFacility AUTH
LogLevel INFO
```

Der **ssh** Server muss nun noch neu gestartet werden, bevor die Einstellungen übernommen werden. Anschließend können wir uns über einen sicher eingerichteten **ssh** Dienst freuen.

Warnung

Der **ssh** Dämon erhält bestehende Verbindungen auch dann aufrecht, wenn er neu gestartet wurde. Nachdem Sie die Konfigurationsdatei geändert haben, sollten Sie sicherheitshalber Ihr "altes" Terminalfenster geöffnet lassen während Sie sich nach dem Neustart von **sshd** wieder einloggen. Auf diese Weise können Sie Loginfehler auf Grund von z.B. Tippfehlern in der Konfigurationsdatei schneller beheben. Wenn der Login nach der Konfigurationsänderung erfolgreich war, können Sie Ihr "altes" Terminalfenster schließen.

```
/etc/init.d/ssh restart
```

Die Konfiguration von **sshd** ist nun abgeschlossen.

SSH Begrüßungsnachricht ändern

Wer nicht daran interessiert ist, bei jedem Login darüber aufgeklärt zu werden, dass die Debian-Entwickler für keinerlei Schaden haften, der kann in der Datei `/etc/motd` eine eigene Begrüßungsnachricht angeben oder den Inhalt einfach löschen. Dort steht nämlich, was dem Systemnutzer nach dem Login via **ssh** angezeigt wird.

```
vi /etc/motd

Willkommen auf www.debianhowto.de
/root sein verpflichtet!
```

Damit sollte unser **ssh** Dämon nun nicht nur sicher(er) eingerichtet sein, sondern auch eher den eigenen Wünschen entsprechen.