FAQ SSH – Secure Shell

Antworten auf die häufigsten Fragen

Von Peter Siering

Das richtige Schlüsselformat

? Welche Schlüssellänge sollte ich für meinen SSH-Schüssel wählen?

! Solche Empfehlungen veralten zu schnell, als dass man sie guten Gewissens geben kann. Zum Jahresanfang 2018 gilt grob: DSA-Schlüssel gelten als unsicher. Sie sollten nicht mehr erzeugt und aus dem Verkehr gezogen werden. RSA-Schlüssel sollten mit einer Länge von mindestens 3072 Bits erzeugt werden. Dem Vortrag „Resilienced Kryptographie“ auf dem CCC-Congress von Rüdiger Weis und Christian Forler zufolge (siehe ct.de/yrp2) sind 2048-Bit-Schlüssel für rund 1000 US-Dollar per Coppersmith-Attacke aus dem öffentlichen Schlüssel errechenbar. Bei den elliptischen Verfahren wie ECDSA und Ed25519 spielt die Länge des Schlüssels keine entscheidende Rolle.

Passwort versus Schlüssel

? Welchen zusätzlichen Sicherheitsgewinn verspricht ein Schlüssel gegenüber einem Passwort, wenn eine SSH-Verbindung doch ohnehin verschlüsselt ist?

! Der Einsatz von Schlüsseln streut das Risiko: Der zur Authentifizierung nötige private Schlüssel verbleibt dabei auf dem Client nicht wie etwa ein Passwort auf dem Server.

Schlüssel ohne Passphrase sollte man tunlichst vermeiden: Gelangen sie in falsche Hände, sind sie ein Freifahrtschein – Ausnahmen für Dienste kann man machen, siehe „Schlüssel ohne Passphrase“.

Ein SSH-Server, der nur Zutritt via Schlüssel gewährt, begrenzt den Spielraum für potenzielle Eindringlinge: Ohne dass die einen gültigen Schlüssel vorweisen, prallen sie ab. Ist die Anmeldung per Passwort erlaubt, können diese hingegen munter Namen und Passwörter durchprobieren. Wer einen SSH-Server auf dem Standard-Port 22 im Internet betreibt, kennt das Treiben, das sich dann schnell in den Logs widerspiegelt (dem man mit fail2ban entgegenwirken kann).

Ein Schlüssel hat obendrein den charmanten Vorteil, dass man den öffentlichen Teil auf mehreren Servern hinterlegen kann. Dabei hilft zum Beispiel `ssh-copy-id`.

Noch mehr Sicherheit lässt sich erreichen, indem Sie SSH mit Multifaktorauthentifizierung kombinieren. Mit OATH-PAM lassen sich SSH und Einmal-Passwörter recht einfach kombinieren (siehe c’t 08/2015, S. 176). Man muss allerdings aufpassen, dass dabei nicht die Anmeldung per Benutzername und Passwort wieder aktiviert wird.

Beste Wahl für Windows

? Was benutze ich idealerweise als Windows-Nutzer, wo SSH (noch) nicht im Lieferumfang enthalten ist?

! Das hängt davon ab, was Sie brauchen: Als minimaler Client bieten sich die mit Git für Windows gelieferte Bash auf Basis von Mingw32 an, die auch einen SSH-Client enthält. Ähnliches, aber viel komplizierter einzurichten, liefert die Cygwin-Umgebung. Microsofts eigene OpenSSH-Implementierung empfiehlt sich in der auf GitHub beziehbaren Fassung. Wenn Sie mehr Komfort wünschen, ist Mobaxterm eine gute Wahl. Das altbekannte und eigentlich bewährte Putty fällt denen gegenüber ab, weil es ein eigenes Format für SSH-Schlüssel verwendet, das lästige Konvertiererei verlangt. Für einen SSH-Server bietet sich heute Microsofts OpenSSH-Implementierung an. Sie kostet nichts und harmoniert auch mit der PowerShell, was keineswegs selbstverständlich ist. Download-Links der genannten Programme finden Sie unter ct.de/yrp2.

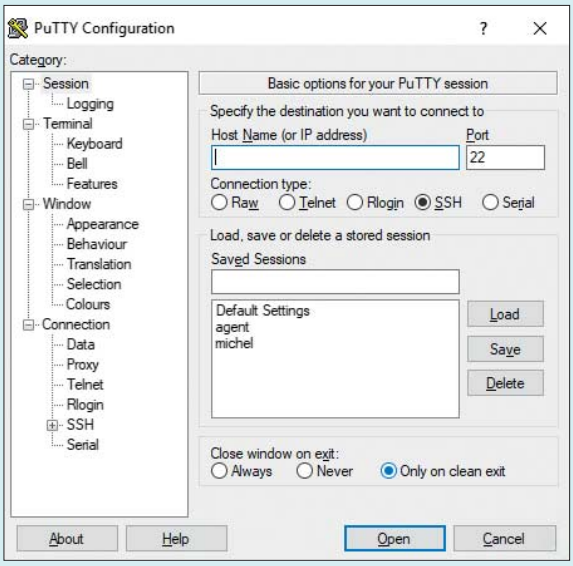
Schlüssel ohne Passphrase

? Für automatisierte Zugriffe auf entfernte Hosts möchte ich gern SSH benutzen. Das geht technisch mit einem Schlüssel ohne Passphrase. Nun habe ich gehört, dass das gefährlich sei. Was ist dran?

! In der Tat sollte man Passphrase-lose Schlüssel nur in Ausnahmefällen verwenden, etwa für nicht anders realisierbare Zugriffe auf Monitoring-Agents auf entfernten Servern via SSH. Es ist möglich, den Zugriff auf das aufzurufende Kommando einzuschränken, indem es dem Eintrag für den öffentlichen Teil des Schlüssels in der Datei authorized_hosts vorangestellt wird:

```
command="/usr/bin/check_mk_agent",↵
↳no-port-forwarding ssh-rsa AAAA ...
```

Somit kann ein Angreifer, dem der Schlüssel ohne Passphrase in die Hände geraten



Inzwischen eher zweite Wahl: der Windows-SSH-Client Putty. Das Schlüssel-Handling ist damit zu umständlich.

ist, nur dieses Kommando abrufen. Sollte trotzdem eine Shell nötig sein, können abgespeckte Shells wie rbash oder rssh und minimierte Rechte helfen.

Verbindung scheitert wortlos

? Mit manchem Host kann ich mich partout nicht verbinden. Wie kann ich Ursachenforschung betreiben?

! Rufen Sie den SSH-Client mit der Option `-v` auf. Dann sehen Sie viele Details der Verbindungsaushandlung, etwa welche Verfahren bei der Zertifikatsaushandlung verwendet oder verworfen werden. Wie geschwätzig die Ausgabe ist, entscheidet der Client anhand der Menge der Vs, ein `ssh -vvv meinserver` sollte in den meisten Fällen für Klarheit sorgen.

Warnung bei wechselndem Host-Key

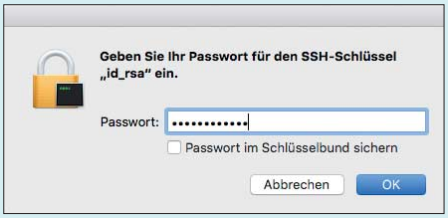
? Wenn sich der Host-Key ändert, spucken SSH-Clients eine Warnung aus und verweisen auf eine Zeile in der Datei known_hosts. Wie kann man mit dieser Angabe einfach eine Zeile löschen?

! Das geht mit `ssh-keygen -R meinserver`. Den Namen des Hosts, den Sie aus der Datei entfernen wollen, müssen Sie freilich anpassen. Aber: Der Host-Key ändert sich eigentlich nie. Prüfen Sie also erst, ob nicht etwas faul ist. Bei Neuinstallation eines Betriebssystems, etwa nach dem Neubespielen einer Raspi-Speicherkarte, ist es aber normal.

Passphrase nur einmal eingeben

? Mich nervt die ständige Eingabe der Passphrase, kann man sich das nicht schenken?

! Normalerweise bringen SSH-Clients auch einen SSH-Agent mit. Diesem Daemon/Dienst gibt man die Passphrase für seine(n) Schlüssel. SSH-Client-Software bittet dann den Agent um Mithilfe beim Verbindungsaufbau, er gibt den entsperrten privaten Schlüssel nicht an die Client-Software weiter. Per `ssh-add` an den Agent überantwortete Passphrases können mit einer Gültigkeitsdauer versehen werden. Praktisch ist ein solcher Agent auch dann, wenn Sie Werkzeuge zum gleichzeitigen Verbinden mit mehreren Hosts nutzen, etwa ClusterSSH. Achtung: Das Weiterleiten einer Agent-Sitzung auf einen entfernten Host sollte man nur nutzen, wenn man dem vollends vertrauen kann.



Ein SSH-Agent auf dem Client, wie er etwa in macOS eingebaut ist, erspart häufige Passphrase-Eingaben, sollte aber nicht allzu leichtfertig benutzt werden.

Tunnel für widerspenstige Dienste

? Mitunter genügen simple Portweiterleitungen mit SSH nicht (Option `-L`), die einen entfernten TCP-Port auf einen lokalen Port verbinden, etwa für die Konfiguration von Speedport-Routern der Telekom. Was kann man tun?

! SSH kann sich lokal wie ein SOCKS4/5-Proxy verhalten. Verbinden Sie sich einfach mit einem Host im entfernten Netzwerk, etwa mit `ssh -D 5454 myhost.example.org`. SSH lauscht dann am lokalen Port 5454, den Sie in einem beliebigen Browser als SOCKS-Proxy eintragen. Alle vom Browser veranlassten HTTP(S)-Zugriffe laufen dann über den entfernten Server.

Bequem und nützlich: Konfigurationsdatei

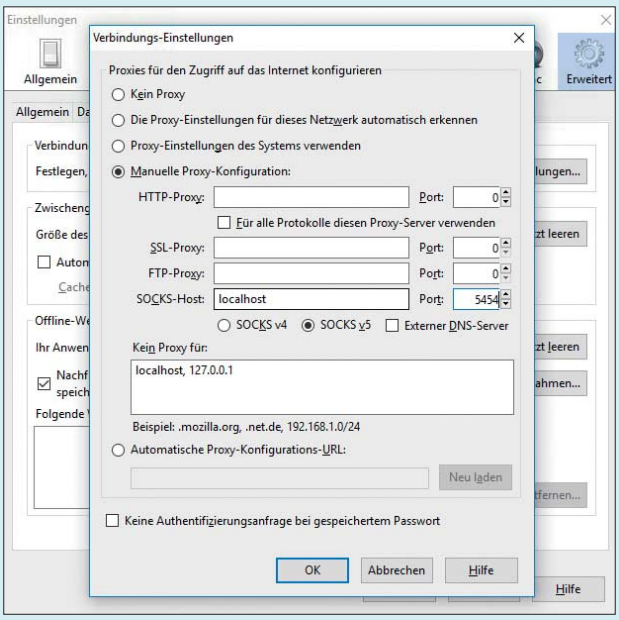
? Ich verwende Software, die Verbindungen per SSH aufbauen kann. Leider arbeitet die nur auf dem Standard-SSH-Port 22 und bietet keine Konfigurationsoption für einen anderen Port. Gibt es einen Trick?

! Sie können abweichende Ports für spezifische Hosts, alternative Ports, gezielt bestimmte Schlüssel, verkürzte Hostnamen und vieles weitere in der config-Datei im .ssh-Verzeichnis eintragen. Ein solcher Eintrag könnte so aussehen:

```
Host pve.example.org
  HostName pve.example.org
  Port 12345
  User root
  IdentityFile pve_key
```

Sie können mehrere solche Einträge vorsehen. Alle im Kontext dieses Nutzers geöffneten SSH-Verbindungen nutzen dann für den jeweiligen Host diesen Port und Nutzernamen. Das klappt aber nur, solange die Programme wie der SSH-Client selbst die libssh bemühen. *(ps@ct.de)*

34C3-Vortrag, Windows-SSH-Software:
ct.de/yrp2



SOCKS hilft, Web-Oberflächen von Routern und anderen Diensten und Geräten zu erreichen, die sich mit einem einfachen SSH-Port-Forwarding und in einem entfernten Netz nicht steuern lassen.