Zehn SSH-Tricks

Julius Plenz

Interaktive Kommandos

- Steuerung von *ssh* direkt, nicht dem darin laufenden Programm
- Escape-Sequenzen beginnen mit einer Tilde (~), sind aber nur nach CR wirksam
- Mögliche Kommandos: Terminate Connection, Open Command Line, Suspend SSH, List Forwarded Connections, ...

Dateien kopieren

- Mit *scp* (secure copy) kann man Dateien über eine verschlüsselte SSH-Verbindung übertragen.
- Dateisysteme auf anderen Rechnern können ganz oder teilweise eingebunden werden, der Zugriff findet immer über die SSH-Verbindung statt. (sshfs, shfs)

Kompression

- Option *-C* schaltet gzip-Kompression ein
- Dateien werden je nach Typ schneller übertragen, bei interaktiven Verbindungen über eine sehr schnelle Leitung lässt sich allerdings ein Performance-Verlust bemerken
- Bei langsamen Leitungen zu empfehlen

Verbindungsteilung I

- Das Protokoll SSH sieht vor, dass über eine verschlüsselte Verbindung mehrere *Kanäle* übertragen werden (Multiplexing)
- So können über *eine* verschlüsselte TCP-Verbindung (Master Connection) mehrere Shells geöffnet, Dateien übertragen und andere TCP-Verbindungen weitergeleitet werden

Verbindungsteilung II

- Vorteile:
  - Passworteingabe entfällt
  - TCP-Handshakes entfallen
  - Geschwindigkeitsvorteil
- Nachteile:
  - Socket-basiert: Potentieller Angreifer, der auf den Socket Zugriff hat, kann ohne Passworteingabe auch auf den entfernten Rechner zugreifen!

X11-Forwarding

- X11-Anwendungen werden auf einem entfernten Rechner ausgeführt, die grafische Ausgabe wird aber über die SSH-Verbindung auf den lokalen Rechner weitergeleitet.
- Sehr langsam, Kompression einschalten
- Auf Grund des X11-Protokoll-Designs hat man mit Round-Trip-Zeiten zu kämpfen (-> FreeNX!)

Local-Port-Forwarding

- *Ssh* leitet TCP-Verbindungen, die auf einem bestimmten lokalen Port eingehen über die SSH-Verbindung an den entfernten Rechner weiter.
- Nützlich für:
  - POP3/SMTP verschlüsseln
  - In unsicheren Netzen (Sniffer) über einen sicheren HTTP-Proxy surfen
  - Einzelne Verbindungen durch eine Firewall schleusen

Remote-Port-Forwarding

- Pedant zum Local-Port-Forwarding: Verbindungen, die auf dem entfernten Computer eintreffen werden über die SSH-Verbindung auf den lokalen Computer weitergeleitet
- Anwendung:
  - Zugriff auf Computer in einem NAT, ohne dass der Router Port-Forwarding unterstützt
  - Umgehung von Firewalls für einzelne Verbindungen

Dynamic Forward I

- *ssh* startet einen SOCKS4/5-Server auf dem lokalen Computer
- TCP-Verbindungen, die auf diesem SOCKS-Server eingehen, werden über die verschlüsselte Verbindung an den entfernten Rechner weitergeleitet
- Dieser leitet die Verbindung wiederum an die Empfängeradresse weiter

Dynamic Forward II

- Besonders auf Kongressen sehr praktisch :-)
- Programme, die das SOCKS-Protokoll nicht unterstützen (Opera, lynx, ...), können per *tsocks* umgeleitet werden
- Jede Verbindung kann getunnelt werden
- Gefahr: DNS-Anfragen

Encrypted Tunnel

- OpenSSH erstellt auf dem entfernten und lokalen Rechner je ein *tun*-Interface
  - Root-Rechte benötigt
  - *tun*-Kernelmodul muss geladen werden
- Über das tun-Interface geschickte Daten kommen beim jeweils anderen Rechner an
- So lassen sich *beliebige* Daten tunneln

Poor Man's VPN I

- Durch Setzen von Routen und eventuell IPTables-Regeln kann nun ein vollwertiges VPN aufgebaut werden
- Vorteile
  - Sehr schnelle und flexible Einrichtung (< 1 min)
  - Kein neues Protokoll, keine Zertifikate
  - Keine Config-Dateien

Poor Man's VPN II

- Nachteile
  - TCP over TCP (Retransmission-Problem)
  - Betriebssystemspezifisch (*tun*-Interface)
  - VPN aus mehreren Subnetzen wirft schon Schwierigkeiten auf
- Kein Ersatz für professionelle VPN-Lösungen
- Aber: Für ein VPN mit dem heimischen LAN reicht's

Danke! — Fragen?

<julius@plenz.com>