SSH Secure Shell

- Kurzakte –

Server:

Installation	
• Server-Binary standardmäßig installiert in /usr/sbin/sshd (Download: http://www.ssh.fi) • Achtung: Versionen SSH 1.x und SSH 2.x sind nicht kompatibel (Versionsanzeige mit ssh –V)	
Besondere Merkmale	
• verwendet die Protokolle SSL (Secure Socket Layer) und SSH (Secure Shell) am TCP-Port 22 • Verbindung wird symmetrisch verschlüsselt mit DES oder IDEA • Schlüsselübergabe erfolgt asymmetrisch verschlüsselt über RSA oder DSA (DSA patentrechtlich unbedenklich)	
Konfiguration Server	
/etc/ssh/sshd_config	• Hauptkonfigurationsdatei von sshd
/etc/ssh/ssh_host_key /etc/ssh/ssh_host_key.pub	• das hostbezogene RSA1-Schlüsselpaar des Servers (SSH-Version 1) • werden meist bei der SSH-Installation durch ein Skript erzeugt • manuell mit: ssh-keygen –f /etc/ssh/ssh_host_key –N " " –t rsa1
/etc/ssh/ssh_host_rsa_key /etc/ssh/ssh_host_rsa_key.pub /etc/ssh/ssh_host_dsa_key /etc/ssh/ssh_host_dsa_key.pub	• Host-Schlüsselpaare für SSH-Version 2, (privater und öffentlicher Schlüssel) • wird bei jeder Verbindungsaufnahme vom Server zum Client geschickt • je nach dem verwendeten asymmetrischen Verschlüsselungsverfahren RSA oder DSA
~/.ssh/authorized_keys	• enthält **benutzerbezogene** öffentliche Schlüssel von Clients, um sich ohne Passwortabfrage auf diesem Konto einloggen dürfen (den Usern wird vertraut) • diese liegen in Datei ~/.ssh/ id_dsa.pub des Client-Rechners kopiert mit: cat ~/.ssh/id_dsa.pub | ssh Server-IP "cat >> .ssh/authorized_keys" • muss vom Admin des SSH-Servers gepflegt werden
/etc/environment ~/.ssh/environment	• neben den globalen Systemvariablen werden auch die in diesen Dateien enthaltenen Umgebungsvariablen geladen
etc/ssh/sshrc ~/.ssh/rc	• systemweite und benutzerbezogene Startdateien der SSH-Shell, die alternativ abgearbeitet werden
/etc/hosts.equiv ~/.rhosts	• Dateien enthalten hostweite Äquivalenzen bzw. Benutzeräquivalenzen • primitive passwortfreie Authentifizierung der "r-Utilities" auch für SSH • nur wenn auf Server HostbasedAuthentication u. IgnoreRhosts gesetzt (V. 2)
/etc/ssh/shosts.equiv ~/.shosts	• Dateien enthalten hostweite Äquivalenzen bzw. Benutzeräquivalenzen • primitive passwortfreie Authentifizierung nur für SSH gültig (nicht r-utilities)
/etc/issue.net	• Nachricht für Remote-Benutzer als Login-Meldung (Banner /etc/issue.net)
/etc/motd	• Nachricht für Benutzer nach erfolgreichem Login (PrintMotd yes)
/etc/nologin	• Loginverbot für alle User außer root, falls Datei existiert (UseLogin yes)
Konfiguration Client	
ssh –v –l jmeese SSH-Server	• Einloggen als Benutzer jmeese im Debug-Modus zur Problemsuche
~/.ssh/config	• benutzerspezifische Konfigurationsdatei (vor /etc/ssh/ssh_config gelesen)
/etc/ssh/ssh_config	• systemweite Konfigurationsdatei des Clients ssh • wird nach den Optionen der Kommandozeile und ~/.ssh/config ausgelesen • für jeden Parameter gilt der erste gefundene Wert
etc/ssh/ssh_known_hosts ~/.ssh/known_hosts	• enthält öffentliche Host-Schlüssel der SSH-Server, auf die man bereits via SSH zugegriffen hat (Schlüsseldatenbank) • wird automatisch angelegt und erweitert, wenn bei Bejaung der Abfrage (ausgewertet entsprechend Wert von StrictHostKeyChecking des Clients) • Syntax Version 2: Optionen Schlüsseltyp Key(base64-Format) Kommentar from="*.zurquelle.de,!pc.hacker.de" ssh-rsa ….FXlnllfOs=Kommentar
~/.ssh/ id_rsa ~/.ssh/ id_rsa.pub ~/.ssh/ id_dsa ~/.ssh/ id_dsa.pub	• das RSA oder DSA-Schlüsselpaar des Users (Version 2) • muss erst mit "ssh-keygen –t dsa –b 1024" erzeugt werden • Inhalt der id_dsa.pub kann für eine automatische Anmeldung auf einen Remote-Rechner kopiert werden (dort in die authorized_keys anhängen)
~/.ssh/identity ~/.ssh/identity.pub	• RSA1-Schlüsselpaar des Users zur Authentifizierung bei Version 1
Start als Stand-Alone-Dienst	
/etc/init.d/sshd rcsshd (SuSE)	• Parameter: start | stop | restart | reload | status - Skript startet/stoppt ssh manuell
insserv sshd (SuSE)	• erzeugt die Links in den Runleveln automatisch (alternativ Runleveleditor)
Dokumentation	
man: sshd (8), ssh (1), scp (1), ssh-add (1), ssh-agent (1), ssh-keygen (1), make-ssh-known-hosts (1),	

Client:

- Windows-Clients:
  - Putty öffnet Remote-Shell (http://www.chiark.greenend.org.uk/~sgtatham/putty/)
  - WinSCP zum Kopieren von Dateien über Rechnergrenzen hinweg

- Linux-Client:

  gftp: grafisch, mit Dateibrowsing lokal und remote
  konqueror: fish://Servername
  /usr/bin/ssh:
  - "**ssh servername**" öffnet eine Remote-Verbindung
  - "**ssh –v servername**" "verbose mode", SSH wird geschwätzig für Debugging
  - "**ssh -c blowfish servername**" öffnet eine Remote-Verbindung mit schnellem Blowfish
  - "**ssh –l username servername**" öffnet eine Remote-Verbindung mit diesem Account
  - "**ssh username@servername**" w. o.
  - "**ssh servername Kommando**" Führt einen Remote-Befehl aus
  - "**ssh –f servername Kommando**" Führt einen Remote-Befehl im Hintergrund aus
  - "**ssh –X server Kommando**" startet X-Anwendungen auf dem fremden Rechner (Display :11.0), die dann lokal dargestellt werden. Hierbei wird die lokale DISPLAY-Variable gleich passend gesetzt (-X aktiviert temporär X11-Forwarding bei nicht gesetztem "X11Forwarding yes" der /etc/ssh/sshd_config und/oder aktivierten "ForwardX11 no " der /etc/ssh/ssh_config)

  - "**ssh –f –L Clientport:Server:Serverport [–C] user@Server Kommando**"
    errichtet einen SSH-Tunnel von local nach remote
    der Datenverkehr erfolgt über den lokalen Client-Port, die Übertragung aber über den sicheren Tunnel
    Von remote nach local wird statt des Parameters -L der Parameter -R benutzt
    der Parameter -o GatewayPorts=Yes stellt auch anderen Rechnern diesen Tunnel zur Verfügung

  - "**slogin servername**" öffnet eine Remote-Verbindung

  - "**scp –p username@Quellhost:/Quellpfad /Zielpfad**" zum verschlüsselten Kopieren von Daten über Rechnergrenzen hinweg (-p preserve: Rechte und Zeitangaben der Datei bleiben erhalten)

  - "**sftp username@server:Datei1 Datei2**" Secure FTP (nur bei SSH2)
    Vorteil gegenüber scp: Einsatz von Wildcards bei den Dateinamen
    Damit es funktioniert muss serverseitig der Eintrag in der sshd_config freigeschaltet sein wie:
    Subsystem sftp /usr/lib/ssh/sftp-server

  - "**ssh-keygen –t dsa –b 1024**" erzeugt DSA-Schlüssel
    - für Version 1 nur nur Typ rsa1 → Schlüsel nach ~/.ssh/identity[.pub])
    - für Version 2 nur Typen dsa und rsa → Schlüsel nach ~/.ssh/id_{d,r}sa[.pub]

    Passphrase nur sinnvoll bei Notebooks zum Key-Schutz
    nachträglich den privaten Schlüssel verschlüsseln mit: ssh-keygen –p

  - "**ssh-keyscan**" erstellt eine Liste mit bekannten öffentlichen Host-Keys einer Domain:
    ssh-keyscan -4 –t rsa,das –f /etc/ssh/ssh_hosts >> /etc/ssh/ssh_known_hosts (für IPv 4 und SSHv2)
    die abzufragenden Rechner stehen in der zu erstellenden Hostliste ssh_hosts mit folgender Syntax:
    192.168.1.100 doc.zurquelle.de,doc,192.168.1.100 (enthält FQDN, Alias und IP)

  - "**ssh-agent**" zur priv. Schlüsselverwaltung und zur Vereinfachung des Logins bei gesetzter Passphrase merkt sich einmal eingegebene Passphrasen bei wiederholten SSH-Verbindungen,
    dazu wird mit "eval `ssh-agent` eine Sub-Shell gestartet ("set | grep SSH" listet die nötigen Variablen)
    weiterhin wir ein Unix-Domain-Socket errichtet (angezeigt durch "netstat –ax | grep ssh")
    Hinzufügen der Schlüssel mit "ssh-add ~/.ssh/id_dsa", Listen: "ssh-add –l", Löschen aller keys: -D

  - "**ssh-add**" fordert die Eingabe der Passphrase für jeden gefundenen privaten Schlüssel unter ~/.ssh/ und registriert diese Schlüssel beim SSH-Agent

Beispiele:

- ssh paul@192.168.32.75 (öffnet Remotekonsole für Benutzer Paul)
- ssh –X 192.168.32.75 /opt/kde/bin/ksokoban & (Aufruf im X-Terminal)
- scp paul@192.168.32.75: . (kopiert von remote aus dem Heimatverzeichnis von Paul in das lokale Arbeitsverzeichnis)
- ssh -f -L 7777:192.168.1.1:80 192.168.1.1 (öffnet SSH-Tunnel vom lokalen Rechner zu einem Webserver. Zu erreichen ist die Webseite jetzt lokal unter http://localhost:7777)