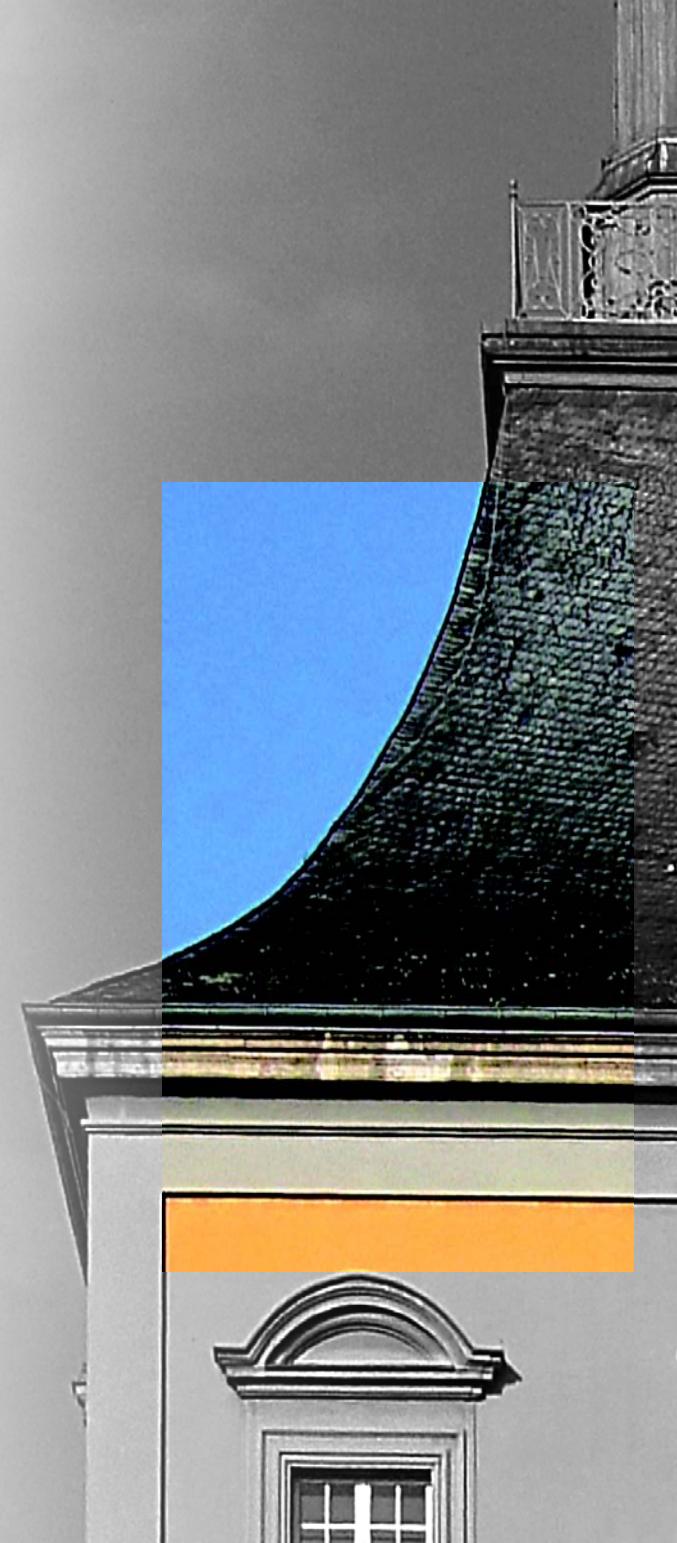


VORLESUNG
NETZWERKSICHERHEIT

SOMMERSEMESTER 2022
MO. 14-16 UHR



KAPITEL 8

SSH

remote shell (rsh)

- 1977 als Teil der BSD-Remote-Utils veröffentlicht
- TCP-Port 514
- Authentifikation über
 - Gewählten Benutzernamen und
 - IP-Port-Kombination des anfragenden Rechners
- Übertragung im Klartext

Kerberos war möglich, damit auch Verschlüsselung, aber Export eingeschränkt

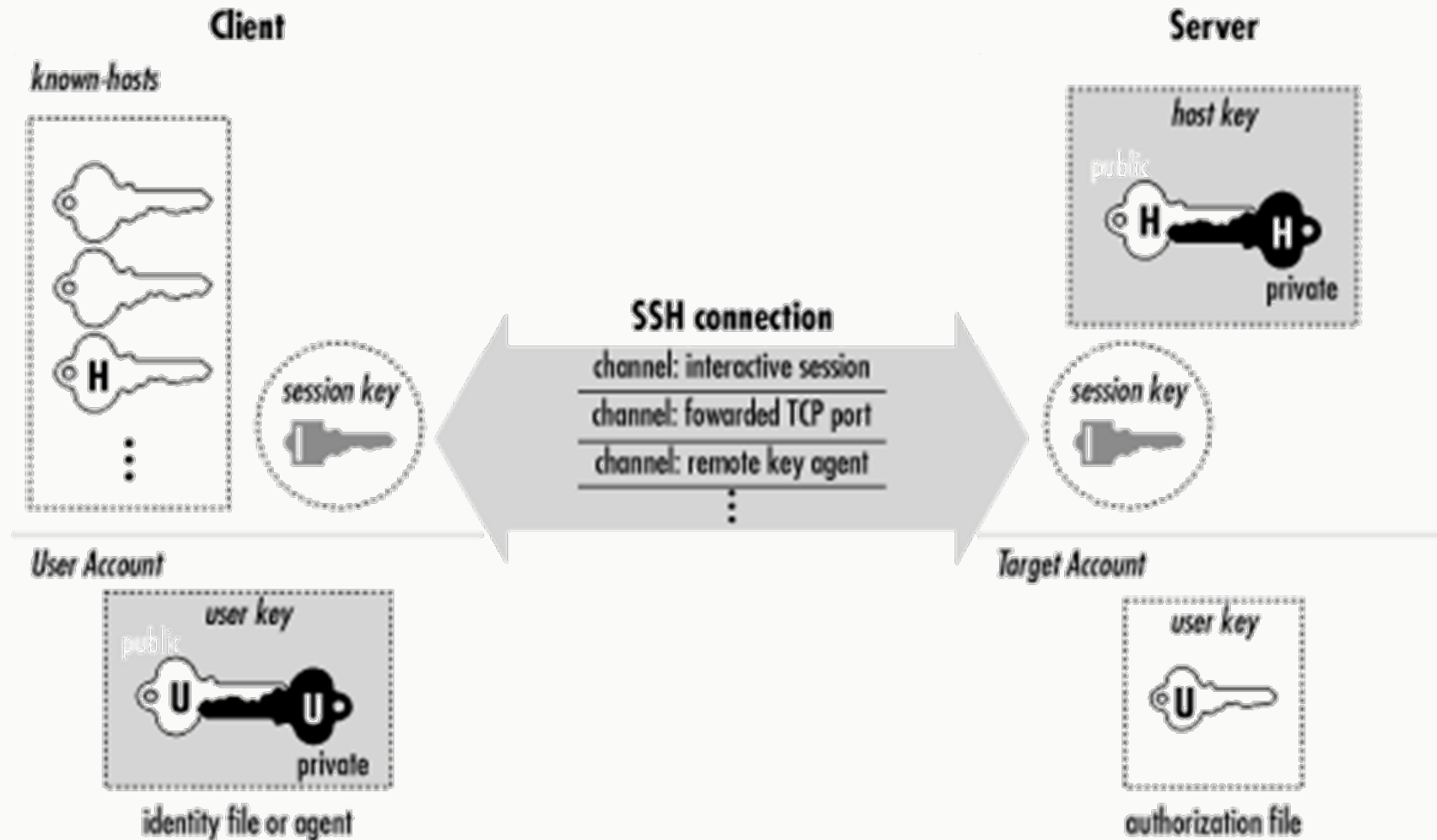
Weitere Tools:

- rcp (= remote copy)
- rlogin (=remote login)
- rexec (=remote execute)

secure shell (ssh)

- Protokoll und Werkzeug
- TCP-Port 22
- Version 1 im Jahre 1995 als sichere Alternative zu den r-tools entwickelt
- Erst Open-Source, anschließend als kommerzielles Produkt vermarktet
- Protokoll-Version SSH-2 bereits 1996 veröffentlicht
- RFC 4251 erst im Januar 2006 veröffentlicht

SSH-ARCHITEKTUR



Im Gegensatz zu SSH-1 gibt es 3 Protokolle(-Teile)

- SSH Transport Layer Protocol (SSH-TRANS)
- SSH Authentication Protocol (SSH-AUTH)
- SSH Connection Protocol (SSH-CONN)

Funktionen SSH-TRANS

- Server-Authentifikation
- Algorithmus-Aushandlung
- Austausch des Sitzungsschlüssels
- Verschlüsselung
- Integrität
- Sitzungs-ID (Perfect Forward Secrecy mit ephemeral DH)
 - Ephemeral RSA verursacht deutlich höhere Kosten!

Server-Authentifikation

- Fingerabdruck des öffentlichen Schlüssels eines PK-Schlüsselpaares
 - Schlüssel wird vom Server an den Client gesendet
 - Client speichert den "Host-Key" in der Datei ~/.ssh/known_hosts

- Unterschiedliche Formate der Known-Hosts-Datei

```
192.168.153.101 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAA
```

- Ohne Eintrag oder bei Änderung des Host-Keys gibt es eine Warnung

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.  
ECDSA key fingerprint is SHA256:9uKyQ8p01MrghLVm84XQBNBtAT2nDYfKPkgYhBc8ycI.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.  
Password:
```

- Wie lässt sich auf dem Server der Host-Key-Fingerabdruck bestimmen?

```
[matze@tschita] ~ $ ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub  
256 SHA256:9uKyQ8p01MrghLVm84XQBNBtAT2nDYfKPkgYhBc8ycI root@tschita (ECDSA)
```

- Abschalten der Überprüfung des Host-Keys

```
[matze@tschita] ~ $ ssh -oStrictHostKeyChecking=no localhost  
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.  
Password:
```


SSH TRANSPORT LAYER PROTOCOL

Server-Authentifikation

- Welche Methoden zur Server-Fingerprint-Prüfung gibt es noch?
 - Hinterlegen des Fingerprints im DNS
 - Erstellen der Hostkeys für den DNS-Eintrag

```
[matze@tschita] ~ $ ssh-keygen -r localhost
localhost IN SSHFP 1 1 a39aac708632c51b748107af1d0ec97eca2ba890
localhost IN SSHFP 1 2 1420a8bddfe5be92818bdfb36ac09264867027c9d840910bd34c05ffa6270832
localhost IN SSHFP 2 1 d6d3f71d2e7ac4908cd9d79b1cf62bae1b929b59
localhost IN SSHFP 2 2 a54c01e182d23adcb88cd4597d1c3382e1204b73a9ec1dba3f3180186ccbe21b
localhost IN SSHFP 3 1 afebd4f79d771fa6250b35eaec9f2393a3bfeba9
localhost IN SSHFP 3 2 3b1d2f1f5eed9cdf8c0c7bd809809daf975b70f6472d7651800e96157191f7c
localhost IN SSHFP 4 1 945f1f158c8ee5def001f457b163ba93d335d99e
localhost IN SSHFP 4 2 0b73bf2e92974ac52e0364945770026610917c1718f6ddd75a06c92fb82591e5
```

- Was bedeuten die Zahlen?

| 1 | 2 | 3 | 4 | 5 | 6 |
|-----|-----|-------|---------|---|-------|
| RSA | DSA | ECDSA | ED25519 | - | ED448 |
| | | 1 | 2 | | |
| | | SHA-1 | SHA-256 | | |

(rfc4255, rfc7479, rfc8709)

Server-Authentifikation

- DNS-SSH-Fingerprint-Überprüfung im Client aktivieren (z.B. mit OpenSSH ab Version 6.6)
 - In der Config:

VerifyHostKeyDNS yes

- Auf der Kommandozeile

ssh -o VerifyHostKeyDNS=yes hostname

- Manuelle Prüfung (z.B. mit dig)

dig hostname sshfp +dnssec +multi

SSH TRANSPORT LAYER PROTOCOL

Algorithmus-Aushandlung

- Clientseitig: `ssh -c <cipher spec>`

```
The supported ciphers are:
```

```
3des-cbc  
aes128-cbc  
aes192-cbc  
aes256-cbc  
aes128-ctr  
aes192-ctr  
aes256-ctr  
aes128-gcm@openssh.com  
aes256-gcm@openssh.com  
chacha20-poly1305@openssh.com
```

```
The default is:
```

```
chacha20-poly1305@openssh.com,  
aes128-ctr, aes192-ctr, aes256-ctr,  
aes128-gcm@openssh.com, aes256-gcm@openssh.com
```

- Funktionen SSH-AUTH
 - Annahme: SSH-TRANS war erfolgreich
 - Server authentifiziert
 - Sichere Verbindung aufgebaut
 - Sitzungs-ID erzeugt
 - Clientauthentifikation
 - Publickey
 - Passwort
 - Hostbasiert

SSH AUTHENTICATION PROTOCOL

- Publickey

- Erzeugung:

- Typen: dsa | ed

- Zuordnung (~/.ssh)

```
[matze@tschita] ~ $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/matze/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/matze/.ssh/id_ed25519
Your public key has been saved in /home/matze/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:YGhlyfNKAj7SCvOHEFn6t39/qFzby6HSCmDU8JnNwp8 matze@tschita
The key's randomart image is:
```

```
+--[ED25519 256]--+
|.oo . *.
|. = . X B
|.B = + = .
|. + = . E
|  + o
|  . . . . .
|  . oo.o=..
|  ..+=+.+.
+-----[SHA256]-----+
```

```
[matze@tschita] ~ $ echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINvXSs2L7FIGZPyceFlDvt3rQ8DXYLv"
```

- Nutzung:

```
[matze@tschita] ~ $ ssh -i .ssh/id_ed25519 localhost
[matze@tschita] ~ $
```

Optische Hostkey-Verifikation

```
[matze@tschita] ~ $ ssh -o VisualHostKey=yes localhost
Host key fingerprint is SHA256:9uKyQ8p01MrghLVm84XQBNBtAT2nDYfKPkgYhBc8ycI
+---[ECDSA 256]---+
|++o=oBoo      |
|+E* + O o     |
| = = = X      |
|. o X + +     |
|. B * oS      |
|. = * . .     |
|. o = . .     |
|. o o . .     |
|. . + .       |
+----[SHA256]-----+
Password:
```

SSH AUTHENTICATION PROTOCOL

Password

- Das Benutzerpassword ist dem Benutzer auf dem Server zugeordnet (z.B. über passwd in /etc/shadow)
- Überprüfung im Hintergrund meist mit Pluggable Authentication Module (PAM)
- SSH-Login mit Passwort-Prompt

```
[matze@tschita] ~ $ ssh localhost
Password:
Password:
[matze@tschita] ~ $
```

SSH AUTHENTICATION PROTOCOL

- Hostbasiert
 - Authentifikation anhand der IP-Adresse und der Host-Keys
 - Host-Keys der Clients müssen auf dem Server hinterlegt werden (/etc/ssh/ssh_known_hosts)

```
tschita ~ # ssh-keyscan localhost | tee -a /etc/ssh/ssh_known_hosts
# localhost:22 SSH-2.0-OpenSSH_8.3
# localhost:22 SSH-2.0-OpenSSH_8.3
# localhost:22 SSH-2.0-OpenSSH_8.3
# localhost:22 SSH-2.0-OpenSSH_8.3
# localhost:22 SSH-2.0-OpenSSH_8.3
localhost ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDaOnhoOr2JbFbkkSDGkx
ZF+kepMGU0olHOLL14rYyQcAjeYzt7FkJtLEip/17VcHK7vr+0qtfutphmvQP4G4CHo4
WPFJfynwm38zkzDSDtHJchImEfe++/wc6sCLnldVahZ9
localhost ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbml
localhost ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPXMRF2cVQ86gEH7kWcFpv
```

/etc/ssh/ssh_config

```
HostbasedAuthentication yes
EnableSSHKeysign yes
```

/etc/ssh/sshd_config

```
HostbasedAuthentication yes
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts no
```


- Funktionen SSH-CONN
 - Kompression
 - TCP-Port-Weiterleitung (inkl. X-Weiterleitung)
 - Authentifikations-Agent-Weiterleitung
 - Interaktive Sitzungen
 - Entfernte Programmausführung
 - Flusskontrolle
 - Pseudo-Terminals
 - Terminal-Nutzung (Modi, Fenstergröße)
 - Signal-Weitergabe

- Kompression
 - Datenstrom wird komprimiert
 - Geringere Bandbreite
 - Höhere Last auf Server- und Clientseite
- TCP-Port-Weiterleitung (inkl. X-Weiterleitung)
 - Lokale-Weiterleitungen

```
[matze@tschita] ~ $ ssh -L <bind_addr>:<port>:<target>:<port>
```

- Remote-Weiterleitungen

```
[matze@tschita] ~ $ ssh -R <bind_addr>:<port>:<target>:<port>
```

- Dynamischer Proxy (SOCKS)

```
[matze@tschita] ~ $ ssh -D <port>
```

■ Authentifikations-Agent-Weiterleitung

```
[matze@tschita] ~ $ eval $(ssh-agent)
Agent pid 21785
[matze@tschita] ~ $ ssh-add .ssh/id_ed25519
Enter passphrase for .ssh/id_ed25519:
Identity added: .ssh/id_ed25519 (matze@tschita)
[matze@tschita] ~ $ ssh -A -i .ssh/id_ed25519 172.17.0.2
matze@2f0c939fc33f:~$ ssh-add -l
256 SHA256:QbyH65JAd62mzwjWP4zDEesx3Z73rCn9eR0yq/xkZ0o matze@tschita (ED25519)
matze@2f0c939fc33f:~$ ls .ssh/
authorized_keys
matze@2f0c939fc33f:~$ git clone git@git.cs.uni-bonn.de:wuebbel/netzwerksicherheit.git
Cloning into 'netzwerksicherheit'...
The authenticity of host 'git.cs.uni-bonn.de (131.220.6.90)' can't be established.
ECDSA key fingerprint is SHA256:NhGbk13IQhgZ2WfD1Fd7HJk+cJUWJZ+WWqrbxGqol0c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Failed to add the host to the list of known hosts (/home/matze/.ssh/known_hosts).
remote: Enumerating objects: 209, done.
remote: Counting objects: 100% (209/209), done.
remote: Compressing objects: 100% (159/159), done.
remote: Total 209 (delta 87), reused 112 (delta 40), pack-reused 0
Receiving objects: 100% (209/209), 74.69 MiB | 1.74 MiB/s, done.
Resolving deltas: 100% (87/87), done.
matze@2f0c939fc33f:~$ █
```

SSH CONNECTION PROTOCOL

- Interaktive Sitzungen
- Entfernte Programmausführung

```
[matze@tschita] ~ $ ssh -A -i .ssh/id_ed25519 google.com hostname -f  
google.com
```

- Pseudo-Terminals
 - Standard: wie aufrufendes Terminal (ssh -T aus / ssh -t erzwungen)
 - Notwendig für SSH-Control-Channel / Escape-Sequenzen

```
matze@2f0c939fc33f:~$ ~?  
Supported escape sequences:  
~. - terminate connection (and any multiplexed sessions)  
~B - send a BREAK to the remote system  
~C - open a command line  
~R - request rekey  
~V/v - decrease/increase verbosity (LogLevel)  
~^Z - suspend ssh  
~# - list forwarded connections  
~& - background ssh (when waiting for connections to terminate)  
~? - this message  
~~ - send the escape character by typing it twice  
(Note that escapes are only recognized immediately after newline.)
```

SSH VERWENDEN

- Jeder Benutzer kann seine eigene SSH-Client-Konfiguration unterhalten
 - `~/ .ssh/config`

```
Host google.com
    Hostname google.com
    User matze
    Port 1234
    IdentityFile ~/.ssh/keys.d/google.com
    LocalForward 8443 localhost:443
    LocalForward 7767 localhost:7767
    LocalForward 3389 192.168.122.160:3389
```

- Umfangreiche Konfigurationen aufteilen
 - `Include conf.d/*.conf`

Jump-Hosts (SSH über SSH)

- Beispiel Uni-Rechner (SSH-Hosts hera/zeus.cs.uni-bonn.de)
- Zugriff auf internen Lab-Rechner von hera/zeus aus?

```
Host hera
    HostName hera.cs.uni-bonn.de
    User wuebbel
    IdentityFile ~/.ssh/keys.d/hera.cs.uni-bonn.de

Host jh_itsec
    HostName itsec.cs.uni-bonn.de
    User wuebbel
    IdentityFile ~/.ssh/keys.d/itsec.cs.uni-bonn.de
    ProxyJump hera
```

```
[matze@tschita] ~/.ssh $ ssh jh_itsec
Last login: Sun Jul 12 22:37:44 2020 from hera.informatik.uni-bonn.de
-bash-4.1$
-bash-4.1$ exit
Abgemeldet
Connection to itsec.cs.uni-bonn.de closed.
```

- Kaskaden: SSH via Jump via Jump via Jump via Jump....

- „Tunnel“-Modus (TUN-Device)
- Server-Konfiguration anpassen

```
PermitRootLogin prohibit-password
PermitTunnel yes
```

- SSH-Verbindung aufbauen (-w <LocalDevice>:<RemoteDevice> | any)

```
tschita ~ # ssh -v -w any root@jh_itsec
```

- Lokal-Device konfigurieren

```
tschita ~ # ifconfig tun0 up
tschita ~ # ip addr add 10.10.20.1/32 peer 10.10.20.2 dev tun0
```

- Remote-Device konfigurieren

```
[root@apis ~]# ip addr add 10.10.20.2/32 peer 10.10.20.1 dev tun0
[root@apis ~]# ifconfig tun0 up
```

- Routen anpassen, IP-Forwarding , Firewall-Settings (SNAT)

```
[root@apis ~]# route -n
Kernel IP Routentabelle
Ziel          Router        Genmask         Flags Metric Ref    Use Iface
10.10.20.1    0.0.0.0      255.255.255.255 UH          0      0        0 tun0
```

SSH-BENUTZER EINSCHRÄNKEN

- Benutzerrechte einschränken (z.B. Jump only)
 - Betroffene Benutzer in einer Gruppe jumpssh
 - /etc/ssh/sshd_config

```
Match Group jumpssh
  AllowAgentForwarding no
  AllowTcpForwarding yes
  X11Forwarding no
  PermitTunnel no
  GatewayPorts no
  ForceCommand echo 'This account can only be used for ProxyJump (ssh -J)'
```

- Einschränkung in Datei authorized_keys

```
command="/usr/bin/whoami",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINvXSs2L7FIGZPyceFlDvt3rQ8DXYLWwSOVJ0od7D9sz user@host
```


SCP

- Secure Copy

SFTP

- Secure File Transfer (nicht zu Verwechseln mit FTPs)

Viele Werkzeuge verwenden SSH „im Hintergrund“

- rsync

```
[matze@tschita] /tmp/v1 $ rsync -v -e ssh -r matze@google.com:/home/matze/.ssh ./
receiving incremental file list
.ssh/
.ssh/authorized_keys

sent 47 bytes  received 226 bytes  546.00 bytes/sec
total size is 95  speedup is 0.35
```

- git clone (haben wir schon gesehen)
- SSHfs (FUSE-Dateisystem für Linux)

Konfiguration eines SSH-Servers

- Ist der Betrieb notwendig?
 - Ja, SSH ist Mittel der Wahl zur Fernwartung von Servern
- Muss sich ein Root-Benutzer anmelden?
 - Nein, bzw. lässt sich der Root-Login auf einen Login mit Zertifikat beschränken
- Welche Rechte haben „einfache“ Nutzer?
 - Gedanke an „Privilege Escalation“
 - Kommandos einschränken, z.B. über eine Custom-Shell / ForceCommand
- Brute-force-Angriffe gegen SSH
 - Port-Wechsel (Nicht-Standard-Port ist fast komplette Mitigation)
 - Fail2Ban („Ein Tropfen auf den heißen Stein“)
 - SSH-Brute-force-Mirroring (als „Gegenangriff“ bei Port-Wechsel)

Vielen Dank für die Aufmerksamkeit!

Fragen?

Keine nächste Vorlesung:

- Heute ist die letzte!

Nächste Übung:

- Dienstag, 12. Juli 2022 – 16 Uhr
- Abgabe des Übungszettels 12 bis morgen – 16 Uhr