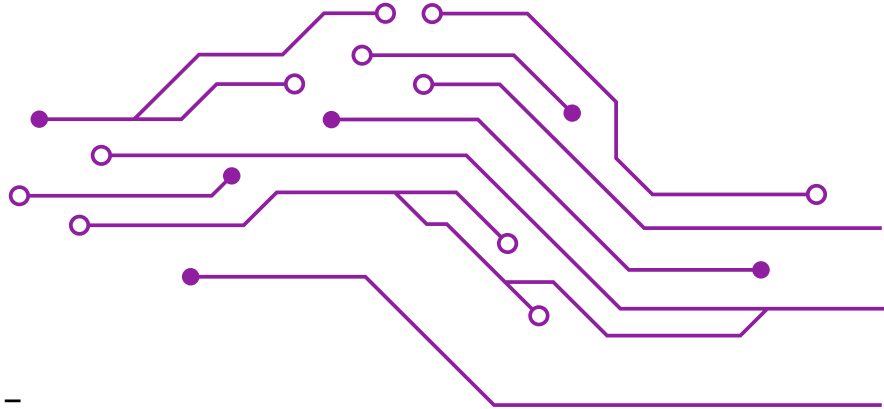
Agile Penguin

SSH – Basics



Rein in die SSH und sicher durch den Tunnel zurück –
Grundlagen der Fernwartung für IT-Sicherheit und Datenschutz

15.11.2023 | Jona Sander

Agenda

01 | Grundlagen

02 | Verbindungsaufbau

03 | Konfiguration

04 | Der Umgang in der Konsole

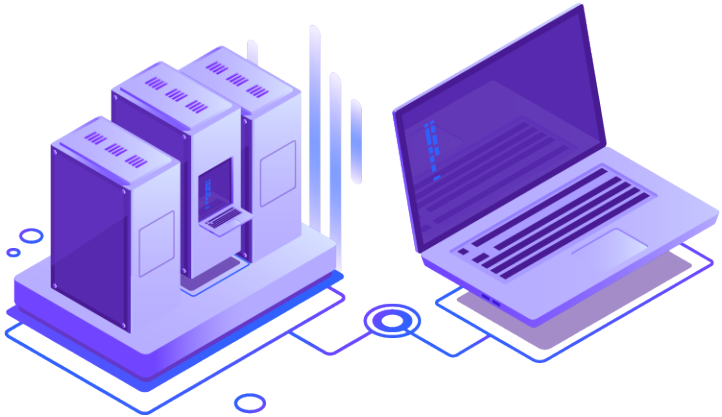
05 | Praktisches aus der Praxis

06 | Hardening

Grundlagen

Einführung – Was kann SSH?

- Mit SSH (SecureSHell) kann remote auf einen Server zugegriffen werden.
- “Remote” Ausführen von Tasks auf einem Server
- Basis für weitere Implementierungen wie SCP/SFTP (Secure copy / SSH file transfer protocol)



Überblick – Eckdaten

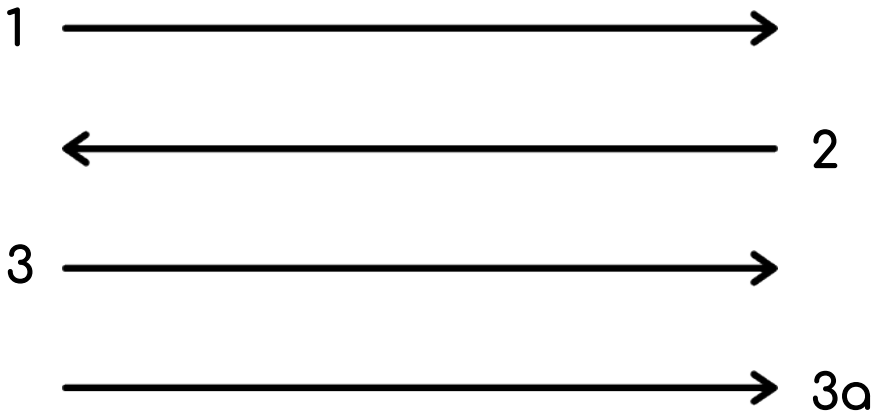
- SSH – SecureSHell
- Entwickelt 1995 in Helsinki
- Sichere Verbindung über Netzwerk
- Server – Client
- Implementierungen für alle OS
- Port 22
- Ersatz für Telnet (Port 23) und FTP (Port 21)
- X11-Forwarding (erlaubt es, Programme mit grafischer Oberfläche via SSH remote Rechner zu starten)

Verbindungsaufbau

SSH – Verbindungsaufbau

Client

Server



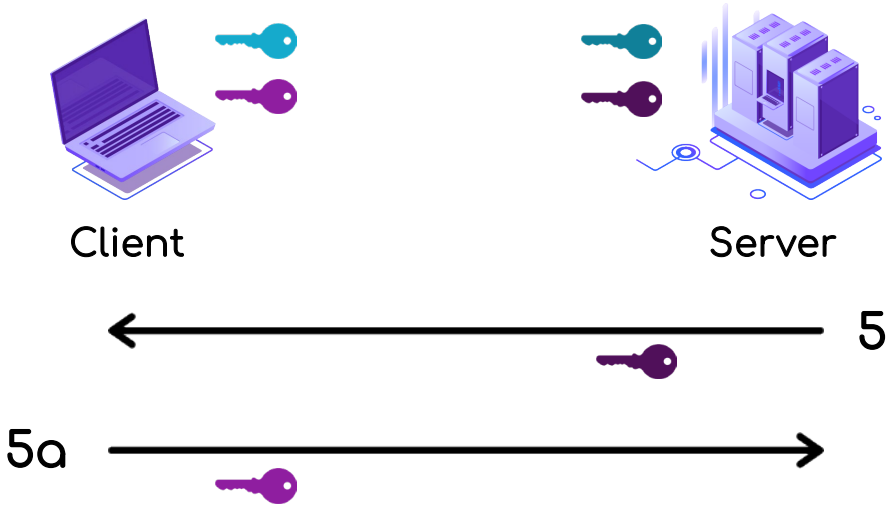
1. Client schickt Anfrage
2. Server antwortet mit supported Protokollversion
3. Client bestimmt zu nutzende Version

3a. Server bestätigt Version

SSH – Verbindungsaufbau

4. Sendet verfügbare Algorithmen
4a. verhandeln zu nutzenden Algorithmus

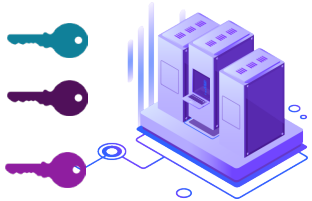
SSH – Verbindungsaufbau



5. Sendet public key y
5a. Sendet public key x

SSH – Verbindungsaufbau

Client



Server

5b ←

→ 5c

Secret integer a

$A = g^a \bmod p$

Secret integer b

$B = g^b \bmod p$

$s = B^a \bmod p$ SessionKey $s = A^b \bmod p$

Session-Key per DIFFIE-HELLMAN

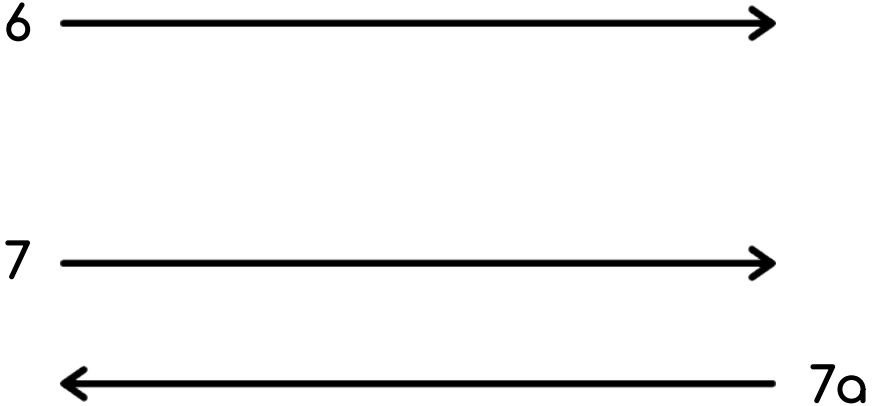
5b. Sendet Primzahlen G und P
5a. Generierung des Session Keys

DIFFIE-HELLMAN

SSH – Verbindungsaufbau

Client

Server



6. Verschlüsselte Authentifizierung per user und password oder key
7. Verschlüsselte Kommandoanfrage

7a. Verschlüsselte Kommandoergebnisse

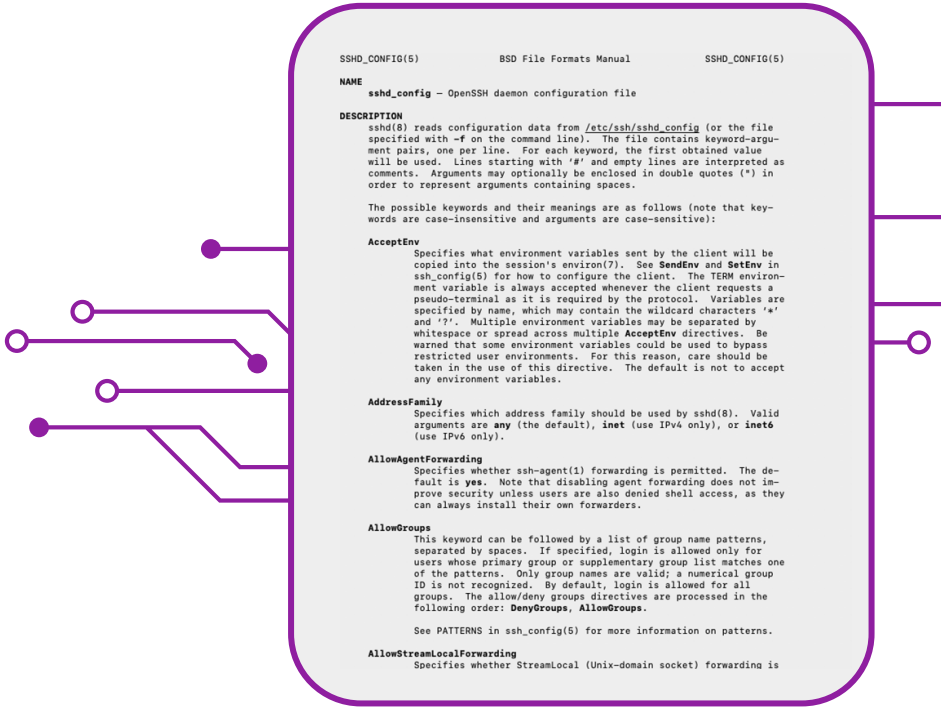
Konfiguration

Konfiguration

- Server-Config `/etc/ssh/sshd_config`
- Client-Config `/etc/ssh/ssh_config`

Konfiguration Server

- Details unter
  `man sshd_config`
- Welcher Port?
- Wer darf wie?



SSHD_CONFIG(5) BSD File Formats Manual SSHD_CONFIG(5)

NAME

sshd_config — OpenSSH daemon configuration file

DESCRIPTION

sshd(8) reads configuration data from /etc/ssh/sshd_config (or the file specified with **-f** on the command line). The file contains keyword-argument pairs, one per line. For each keyword, the first obtained value will be used. Lines starting with '#' and empty lines are interpreted as comments. Arguments may optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

The possible keywords and their meanings are as follows (note that keywords are case-insensitive and arguments are case-sensitive):

AcceptEnv

Specifies what environment variables sent by the client will be copied into the session's environ(7). See **SendEnv** and **SetEnv** in ssh_config(5) for how to configure the client. The TERM environment variable is always accepted whenever the client requests a pseudo-terminal as it is required by the protocol. Variables are specified by name, which may contain the wildcard characters '*' and '?'. Multiple environment variables may be separated by whitespace or spread across multiple **AcceptEnv** directives. Be warned that some environment variables could be used to bypass restricted user environments. For this reason, care should be taken in the use of this directive. The default is not to accept any environment variables.

AddressFamily

Specifies which address family should be used by sshd(8). Valid arguments are **any** (the default), **inet** (use IPv4 only), or **inet6** (use IPv6 only).

AllowAgentForwarding

Specifies whether ssh-agent(1) forwarding is permitted. The default is **yes**. Note that disabling agent forwarding does not improve security unless users are also denied shell access, as they can always install their own forwarders.

AllowGroups

This keyword can be followed by a list of group name patterns, separated by spaces. If specified, login is allowed only for users whose primary group or supplementary group list matches one of the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed for all groups. The allow/deny groups directives are processed in the following order: **DenyGroups**, **AllowGroups**.

See PATTERNS in ssh_config(5) for more information on patterns.

AllowStreamLocalForwarding

Specifies whether StreamLocal (Unix-domain socket) forwarding is

Konfiguration Server - User Access

- SSH benötigt einen berechtigten User
  - Lokal
  - Directory Server
- User / Usergruppen kann das Nutzen entweder
  - Erlaubt werden
  - Verboten werden

```
/etc/ssh/sshd_config:
...
AllowUser leigh vishal stefan  # only allow these users to log in

DenyUser bob mary paulina      # allow all users except these


AllowGroup admins developers   # only allow users in these groups to log in

DenyGroup sales marketing      # allow all users except those in these
                               # groups
```

Konfiguration Server – User Access

- Dedizierte Zugänge konfigurieren für
  - User
  - IP-Adresse

```
/etc/ssh/sshd_config:
...
Match Address 10.0.1.0/24
  PermitRootLogin yes
  PasswordAuthentication yes     # root may log in with password from
                                 # 10.0.1.0/24 addresses only

Match User alice bob Address *
  PasswordAuthentication no
  PubkeyAuthentication yes       # alice and bob may log in from anywhere
                                 # using only their key, not their password
```

Der Umgang in der Konsole

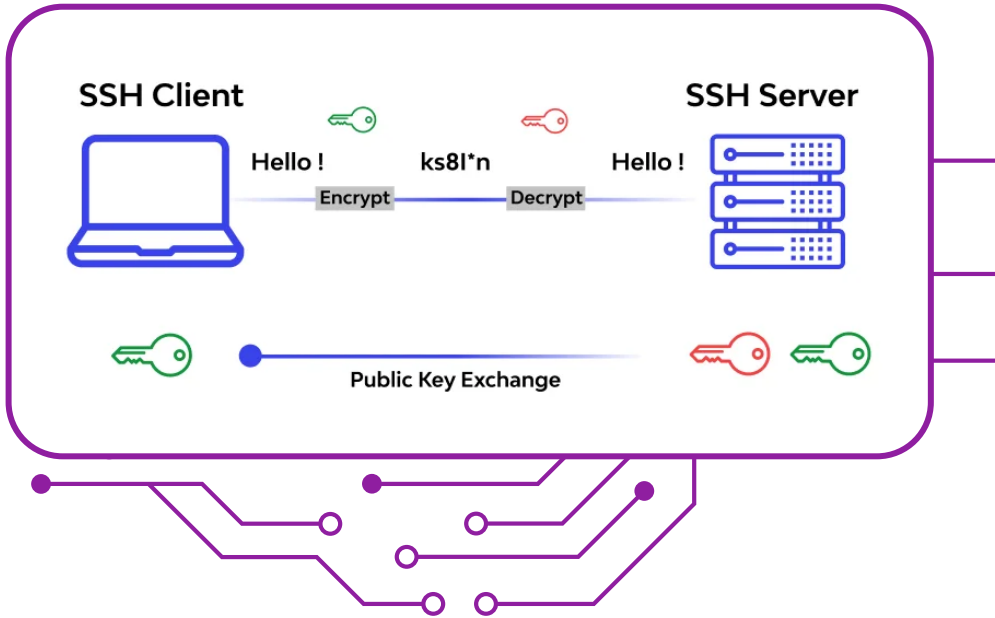
SSH - Verbindung

- `ssh <name>@<server>`
- Fingerprints aller Verbindungen in `~/.ssh/known_hosts`
- Security Feature
- Verbindung beendet mit `exit`

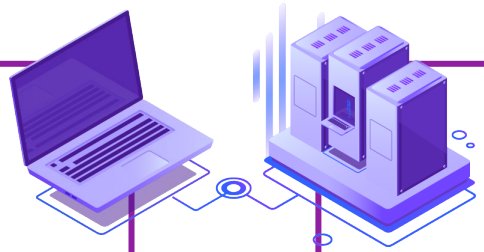
```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:IAPvOCYVu9LEQxV/wjA2SdXh72f0fZydbg4fptBHmBE.
Please contact your system administrator.
Add correct host key in /Users/jonasander/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /Users/jonasander/.ssh/known_hosts:1
Host key for 192.168.178.35 has changed and you have requested strict checking.
Host key verification failed.



jonasander@hephaistos ~ % ssh jona@192.168.178.35
The authenticity of host '192.168.178.35 (192.168.178.35)' can't be established.
ED25519 key fingerprint is SHA256:TRXp5m6XLu9yKxkDcrLioYIBOj6Bwb0r6ecwM0JLViQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

SSH – Verbindung mit keys

- Sicherer als mit Passwort
- “Bequem”
- Automatisierbar



SSH – Verbindung mit keys



Client

Erstellung des Key-Pairs

Server:

Speichern des Client-Keys nach **`~/.ssh/authorized_keys`**

Client

Übermittlung des Public-Keys an den Server:

```
ssh-copy-id -i ~/.ssh/<keyName>.pub <user>@<server>
```

SSH Verbindung – KeyGen

- **ssh-keygen**
- Pfadangabe zur Speicherung des Key-Pairs
- (Optional) Passworteingabe

```
[jona@pudel ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jona/.ssh/id_rsa): /home/jona/.ssh/exampleKey
Created directory '/home/jona/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jona/.ssh/exampleKey
Your public key has been saved in /home/jona/.ssh/exampleKey.pub
The key fingerprint is:
SHA256:G2a1R9xNGZjfFNqB36YfzxNwtJU336Kcgu2AOdvL3EU jona@pudel
The key's randomart image is:
+---[RSA 3072]----+
|              ++*|
|           . =oB=|
|          . o.=o@|
|         . o ..=*|
|       oSo..Eoo+ |
|      +ooooo+ .o |
|       +.o ..  o+|
|      .o....   .+|
|        +..     .|
+----[SHA256]-----+
```

Vorsicht!

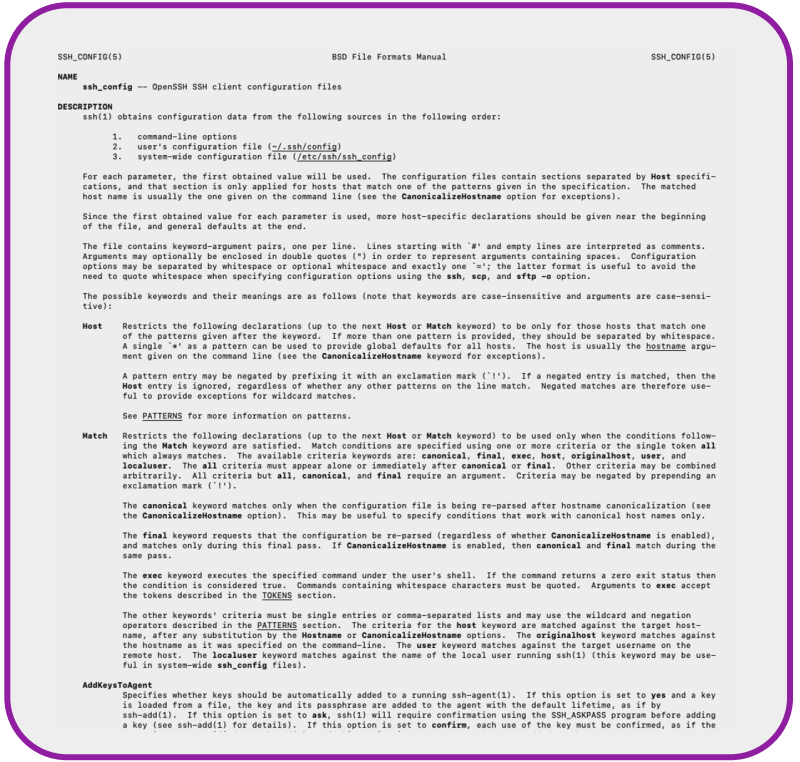
Sollte der private key bekannt werden, sollte unverzüglich ein neues Key-Pair erzeugt werden.

SSH Verbindung per key

- Auf dem Server muss die Authentifizierung per “Pubkey” eingeschaltet werden
- (optional) Passwortauthentifizierung disablen
- sshd Neustarten
- `ssh <user>@<server> -i ~/.ssh/<privKeyFile>`

Konfiguration Client

- User-gebundene Configs unter **`~/.ssh/config`**
- Sytemweite Configs unter **`/etc/ssh/ssh_config`**



SSH_CONFIG(5) BSD File Formats Manual SSH_CONFIG(5)

NAME

ssh_config -- OpenSSH SSH client configuration files

DESCRIPTION

ssh(1) obtains configuration data from the following sources in the following order:

1. command-line options
2. user's configuration file (~/.ssh/config)
3. system-wide configuration file (/etc/ssh/ssh_config)

For each parameter, the first obtained value will be used. The configuration files contain sections separated by **Host** specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is usually the one given on the command line (see the **CanonicalizeHostname** option for exceptions).

Since the first obtained value for each parameter is used, more host-specific declarations should be given near the beginning of the file, and general defaults at the end.

The file contains keyword-argument pairs, one per line. Lines starting with `#' and empty lines are interpreted as comments. Arguments may optionally be enclosed in double quotes (") in order to represent arguments containing spaces. Configuration options may be separated by whitespace or optional whitespace and exactly one `='; the latter format is useful to avoid the need to quote whitespace when specifying configuration options using the **ssh**, **scp**, and **sftp -o** option.

The possible keywords and their meanings are as follows (note that keywords are case-insensitive and arguments are case-sensitive):

Host Restricts the following declarations (up to the next **Host** or **Match** keyword) to be only for those hosts that match one of the patterns given after the keyword. If more than one pattern is provided, they should be separated by whitespace. A single `*' as a pattern can be used to provide global defaults for all hosts. The host is usually the hostname argument given on the command line (see the **CanonicalizeHostname** keyword for exceptions).

A pattern entry may be negated by prefixing it with an exclamation mark (`!'). If a negated entry is matched, then the **Host** entry is ignored, regardless of whether any other patterns on the line match. Negated matches are therefore useful to provide exceptions for wildcard matches.

See PATTERNS for more information on patterns.

Match Restricts the following declarations (up to the next **Host** or **Match** keyword) to be used only when the conditions following the **Match** keyword are satisfied. Match conditions are specified using one or more criteria or the single token **all** which always matches. The available criteria keywords are: **canonical**, **final**, **exec**, **host**, **originalhost**, **user**, and **localuser**. The **all** criteria must appear alone or immediately after **canonical** or **final**. Other criteria may be combined arbitrarily. All criteria but **all**, **canonical**, and **final** require an argument. Criteria may be negated by prepending an exclamation mark (`!').

The **canonical** keyword matches only when the configuration file is being re-parsed after hostname canonicalization (see the **CanonicalizeHostname** option). This may be useful to specify conditions that work with canonical host names only.

The **final** keyword requests that the configuration be re-parsed (regardless of whether **CanonicalizeHostname** is enabled), and matches only during this final pass. If **CanonicalizeHostname** is enabled, then **canonical** and **final** match during the same pass.

The **exec** keyword executes the specified command under the user's shell. If the command returns a zero exit status then the condition is considered true. Commands containing whitespace characters must be quoted. Arguments to **exec** accept the tokens described in the TOKENS section.

The other keywords' criteria must be single entries or comma-separated lists and may use the wildcard and negation operators described in the PATTERNS section. The criteria for the **host** keyword are matched against the target hostname, after any substitution by the **Hostname** or **CanonicalizeHostname** options. The **originalhost** keyword matches against the hostname as it was specified on the command-line. The **user** keyword matches against the target username on the remote host. The **localuser** keyword matches against the name of the local user running ssh(1) (this keyword may be useful in system-wide **ssh_config** files).

AddKeysToAgent

Specifies whether keys should be automatically added to a running ssh-agent(1). If this option is set to **yes** and a key is loaded from a file, the key and its passphrase are added to the agent with the default lifetime, as if by ssh-add(1). If this option is set to **ask**, ssh(1) will require confirmation using the SSH_ASKPASS program before adding a key (see ssh-add(1) for details). If this option is set to **confirm**, each use of the key must be confirmed, as if the

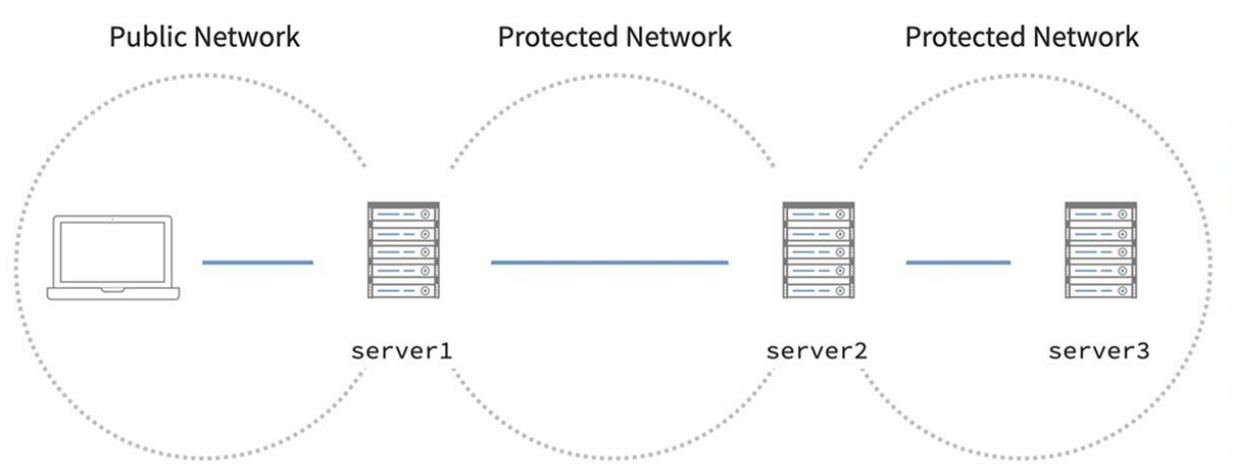
Praktisches aus der Praxis

Praktische Tasks

- SFTP – nutzt die verschlüsselte SSH-Verbindung und SSH-Credentials
- SFTP ist die sichere Variante von FTP
- SCP ist die sichere Variante von CP
- SCP nutzt SSH-Verbindung

Bastion Host

Multistep SSH Connections




```
ssh -J user@server1,user@server2 user@server3
```

Bastion Host

- Steht im öffentlichen / erreichbaren Netzwerk
- Hat ebenfalls Zugang zu geschütztem Bereich
- `ssh -J user@server1,user@server2 user@server3`
- Multistep SSH-Connection
- JumpServer dienen als Türsteher

MoSh – Mobile Shell

- Mobile Shell
- Implementiert ssh
- Fehlertolerant bei Verbindungsabbrüchen
- Wird auf Server und Client benötigt
- Nutzt die Syntax von SSH

Hardening

Securing SSH Server

- Kein Login als Root-User
- Nur das Nutzen von Keys statt Passwort
- Usermanagement:
  Nicht jeden User für SSH berechtigen
- Server nicht direkt im Internet verfügbar, sondern hinter VPN
- IPS (Intrusion Prevention Software), wie beispielsweise Fail2ban
- (SSH-Port ändern)

PermitRootLogin no
(und/(oder) nur per Keys)

PasswordAuthentication no

PubKeyAuthentication yes

AllowUser / DenyUser

Vielen Dank!

Bei Fragen stehe
ich gerne zur Verfügung.
j.sander@agile-penguin.com