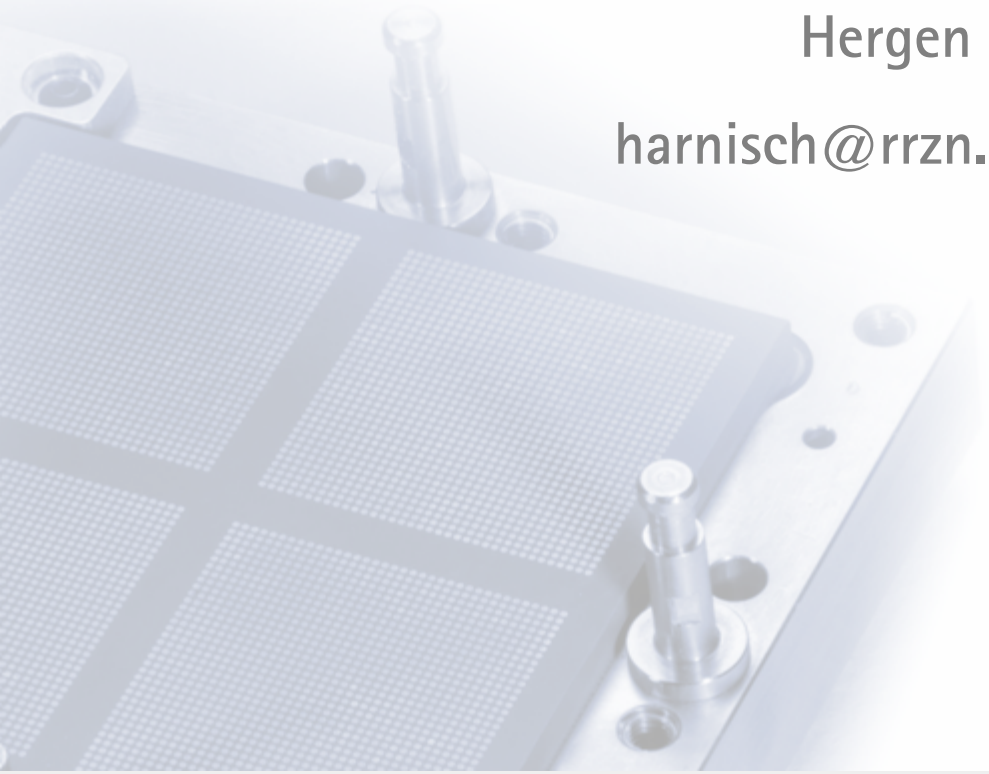
SSH

Hergen Harnisch

harnisch@rrzn.uni-hannover.de



1. Bruteforce-Angriffe
2. Zugriffsbeschränkung:
   - User
   - Quell-IP
   - Keys
3. Versuchsbeschränkung:
   - Fail2ban
   - TCP-Wrapper
4. Command-Beschränkung

Bruteforce-Angriffe

Durchprobieren von Username/Passwort-Kombinationen am SSH-Dienst:

- übliche Usernames: `root`, `guest`, `test`, `mueller`
- teilweise schnell, teilweise langsam & kontinuierlich
- teilweise eine Quell-IP, häufig verteilt (Bot-Netz)
- Wörterbuch-Attacken, angepasst an Deutschland

- kein Implementationsfehler von SSH
- bekanntes Problem von Passwort-Authentifizierung

Zugriffsbeschränkung: User

Maßnahmen

- Reduktion auf nötige User
  - Alt-Accounts löschen
  - nicht jeder benötigt Fernzugriff
  - inaktiven Accounts Fernzugriff nehmen
- letzten Login anzeigen, Logins aufzeichnen
- Remote-Login nur mit personalisierten Accounts
  keine root-Login, erleichtert Nachvollziehbarkeit & Passwort-Änderung
- User zu guten Passwörtern anhalten, besser zwingen

Umsetzung

z.T. technisch (s.u.), aber auch: Organisation, Aufklärung, ...

R|R|Z|N

Umsetzung User-Maßnahmen

Einträge in `/etc/ssh/sshd_config`

- `AllowUsers` / `DenyUsers`:
  einzelne User zulassen oder ausschließen
- `AllowGroups` / `DenyGroups`:
  Mitglieder von Gruppen zulassen oder ausschließen
  z.B. Gruppe `sshusr` ($\neq$ `ssh`) anlegen, analog zu `audio`, `floppy`
- `PermitRootLogin no`
- `PrintLastLog yes`

Sonstiges

gute Passwörter durch regelmäßige Crack-Versuche oder PAM;
Zugangsrestriktionen auch über PAM möglich

Zugriffsbeschränkung: Quell-IP

Maßnahme

Einschränkung der zugelassenen Systeme, z.B. nur `130.75.*`.

Vorteile

wirkungsvoll gegen Scans von Außerhalb & schnell umsetzbar

Nachteile

Umgehung durch legitime Nutzer meist durch Zwischenhosts:

Problem: Passwort-Eingabe auf evt. unsicherem Zwischenhost

Zugriffsbeschränkung: Quell-IP

Maßnahme

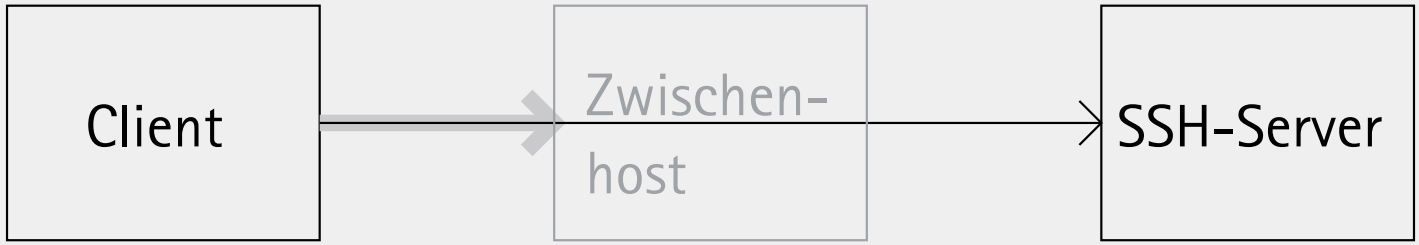
Einschränkung der zugelassenen Systeme, z.B. nur `130.75.*`.

Vorteile

wirkungsvoll gegen Scans von Außerhalb & schnell umsetzbar

Nachteile

Umgehung durch legitime Nutzer meist durch Zwischenhosts:



Tunnel wird meist sowieso nicht genutzt

Umsetzung

TCP-Wrapper

- SSH mit TCP-Wrapper-Support übersetzen, meist Standard
- nur erlaubt, wenn in `hosts.allow` oder nicht in `hosts.deny`
  - `/etc/hosts.allow`:
    ```
    sshd: 130.75.0.0/16 127.0.0.1/32
    ```
  - `/etc/hosts.deny`:
    ```
    sshd: ALL
    ```
- komplizierte Bedingungen mit `*`, Listen, `EXCEPT` möglich

IP-Tables

zusätzliche Absicherung über IP-Tables

Zugriffsbeschränkung: Keys

Maßnahme

Bruteforce-Angriffe nutzen Passwort-Authentifizierung, daher Verzicht auf Passwörter.

Umsetzung

Verwendung alternativer Authentifizierungen

- Public-/Private-Key-Paare ⟵ *Mittel der Wahl!*
- PKI & Zertifikate ⟵ wenige Clients/Server
- Einmal-PW: S/Key, OTP; Tokens ⟵ aufwändig bzw. teuer
- (keine Nutzerauthentifizierung: hostbasiert) ⟵ unsicher

Zugriffsbeschränkung: Keys

Public-/Private-Keys

- Prinzip entspricht den Keys in einer PKI
- Key-Paar: geheimer Private-Key, publizierbarer Public-Key
- Private-Key nicht ratbar, da lang & komplex
- Private-Key ist geheim zu halten, mit Passphrase zu schützen; allerdings mit ssh-Agent bequem sitzungsweise ladbar
- Public-Key dient der Überprüfung des Private-Keys
- Key-Paare in SSH:
  - Host-Keys: meist automatisch generiert, Kenntnis schützt vor Man-in-the-Middle
  - User-Keys: zu generieren, ggf. mehrere Paare, authentifiziert User

Zugriffsbeschränkung: Keys

User-Key-Paar

- auf Client: Schlüsselgenerierung mit `ssh-keygen`
- Public-Key auf Server kopieren, dort anhängen an `~/.ssh/authorized_keys`
- auf Client in `~/.ssh/config` evt.

  `Host` *Servername oder -IP*

  `User` *Username auf Server*

  `IdentityFile ~/.ssh/`*Private-Key-File*

Beispiel Key-Dateien

Private-Key ~/.ssh/id_rsa

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,93BA59259CCFA1AC

q0qYbAzCt1Bs7Jv74gO1fioMbusEjDo0i7xHYyJ8Gras7s+dfiv+80sPVyY/uLDv
...
DmlkuAAfSXTV3K3xTnyHt865p2dNAJe1b5qc5/0PU7ZY0bmIx8bjgw== -----END RSA
PRIVATE KEY-----
```

Public-Key ~/.ssh/id_rsa.pub

```
ssh-rsa AAAAB3N ...9290V0Aw== User@Host
```

R|R|Z|N

Erzwingung Key-Authentifizierung

in `/etc/ssh/sshd_config`:

- alle zwingen: `PasswortAuthentication no`
- mindestens `root`: `PermitRootLogin without-password`

Neuerdings möglich: fallweise Erzwingung

mittels `Match`-Direktive bedingte (u.A. IP, Nutzer) Konfiguration:

```
PubkeyAuthentication yes
ChallengeResponseAuthentication no
PasswordAuthentication no
KbdInteractiveAuthentication no

Match Address 130.75.0.0/16
  PasswordAuthentication yes
```

Versuchsbeschränkung:

Ziel

Anzahl möglicher Rateversuche reduzieren / begrenzen

erreichen mit

Timeouts nach fehlgeschlagenen Versuchen;
allerdings Gefahr von DoS, daher eigene IPs whitelisten

bei SSH

SSH bietet `MaxAuthTries` in `/etc/ssh/sshd_config`:

- begrenzt Versuchszahl je Verbindung
- aber erneute Verbindung sofort möglich
  $\longrightarrow$ praktisch wirkungslos

Versuchsbeschränkung: Fail2ban

- Blocken von Hosts mit zuviel Fehlversuchen in definierten Zeiträumen über IPTables oder TCP-Wrapper *(action)*
- Automatische Aufhebung der Blockierung nach gewisser Zeit
- Mitlesen der Logs (z.B. `/var/log/auth.log`), unterstützt u.A. sshd, Apache *(filter)*
- eigentlich ein Python-Skript, inzwischen relativ komplex
- aufgeteilt in Server-Prozess im Hintergrund und Konfigurations-Client
- Sammlung vorgefertigter Filter und ggf. ausgeführter Aktionen:

  Filter `/etc/fail2ban/filter.d/*.conf`, z.B. `sshd.conf`

  Aktionen `/etc/fail2ban/action.d/*.conf`, z.B. `iptables.conf`
- diese auswählbar in `/etc/fail2ban/jail.conf` bzw. `jail.local`
- Voreinstellung Debian-Paket: ssh-Schutz über iptables

`http://www.fail2ban.org/`

Versuchsbeschränkung: TCP-Wrapper

Sehr rudimentär mit „Bordmitteln“, führt auch zum Lockout legitimer Nutzer:

- `/etc/hosts.allow`:
  ```
  sshd: ALL EXCEPT /var/local/ssh-blacklist:
          spawn (echo '%a' >> /var/local/ssh-blacklist &)
  ```
- `/etc/hosts.deny`:
  ```
  sshd: /var/local/ssh-blacklist
  ```
- Crontab zum Löschen der Datei `ssh-blacklist` alle 5 Minuten

Wirkung:

- Bei jeder ssh-Verbindung (erfolgreicher Login oder fehlgeschlagener Versuch) wird die IP in die Blacklist eingetragen.
- Die Blacklist wird nach maximal 5 Minuten komplett gelöscht.

Command-Beschränkung

Automatischer Datenaustausch zwischen Servern häufig per ssh.

- auf Server A ist Key für ssh-Login auf Server B hinterlegt
- echter Shell-Zugang auf Server B ist weitgehendes Recht
- Einschränkung auf gewünschte Dateikopie sinnvoll

Umsetzbar mit `ForceCommand` in `sshd_config` oder speziell für Key, Einschränkungen in `authorized_keys` als Option vor Key spezifizieren:

```
command=".ssh/wrapper Server-A",no-agent-forwarding,
        no-port-forwarding,no-pty,no-user-rc,no-X11-forwarding
```

Analog auch Einschränkungen für Nutzer realisierbar

Wrapper-Skript

- kann in Abh. der im Bsp. angegebenen Kommandozeilenoption filtern
- erhält eigentlich durch Client gewolltes Kommando in der Umgebungsvariablen `SSH_ORIGINAL_COMMAND`

```
#!/usr/bin/python
import os
import sys
if sys.argv[1]=="TEST":
  print repr(os.environ['SSH_ORIGINAL_COMMAND'])
elif sys.argv[1]=="Server-A":
  # nur Dateiempfang, nur in festgelegte Datei
  os.execv('/usr/bin/scp',('/usr/bin/scp','-t', \
    '/home/hergen/srv-A'))
```

R|R|Z|N

Literatur

- heise Security 21.7.09, D. Bachfeld: „Gewaltfrei – SSH schnell und einfach vor Login-Angriffen schützen“
  http://www.heise.de/security/artikel/SSH-vor-Brute-Force-Angriffen-schuetzen-270140.html