SRS: Sicherer Root-Server

Das Projekt „SRS: Sicherer Root-Server“ soll ein paar Wege aufzeigen, wie ein Server im Internet vor möglichen Angriffen abgesichert werden kann. Ein Anspruch auf Vollständigkeit und kompletter Richtigkeit wird dabei nicht erhoben.

SSH

Der Verbindungsaufbau zu den meisten Root-Servern im Internet (zumindest gilt das für linux-basierte Server) wird wohl über SSH von einem Clientrechner aus erfolgen. Dieser Aufbau kann natürlich direkt und ohne einen eigenen Schlüssel durchgeführt werden. Die nachfolgend beschriebenen Einstellungen bieten aber einen zusätzlichen Schutz.

SSH-Schlüssel

Die Erstellung des Schlüssels kann sowohl unter Windows (sofern OpenSSH installiert ist) oder unter Linux durchgeführt werden, die Wege sind dabei gleich.

Schlüsselerstellung unter Linux:

```
user@linux-client:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
/home/user/.ssh/ssh-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/ssh-key
Your public key has been saved in /home/user/.ssh/ssh-key.pub
The key fingerprint is:
SHA256:F++5Vvf7HhmFJtf8qHXJzIpGNKyJWhyqYQ/ny8w4LtQ user@linux-client
The key's randomart image is:
+--[ED25519 256]--+
|   |
|          .    + |
|       .  .+. + +|
|      o o +o.++.+|
|  .+ o +So... o*o|
| ..EB o  ....+o.+|
|.  . +     o+o +.|
| . .= .   . ..  o|
|  oo.=     ..  o=|
+----[SHA256]-----+
user@linux-client:~$
```

Schlüsselerstellung unter Windows:

```
C:\Users\User>ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\User/.ssh/id_ed25519):
C:\Users\User\.ssh\ssh-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\User\.ssh\ssh-key.
Your public key has been saved in C:\Users\User\.ssh\ssh-key.pub.
The key fingerprint is:
SHA256:b7yGNOQuxms597KeVijiZtRGShwdhKxxKL0c2HN4FlE user@windows-client
The key's randomart image is:
+--[ED25519 256]--+
| + +oBE.         |
|o O O .          |
| o % .           |
|  + o . .        |
|   . + oS.       |
|    + + =o.      |
|   o + = ++      |
|    + B *o..     |
|   o o.B+=o      |
+----[SHA256]-----+

C:\Users\User>
```

Wenn eine „Passphrase“ vergeben wurde, ist es sehr wichtig, sich diese zu merken oder an einem sicheren Ort abzulegen. Auch der private Schlüssel („ssh-key“) sollte an einem Ort abgelegt werden, der nicht öffentlich zugängig ist.

Der öffentliche Schlüssel kann im zweiten Schritt auf den Root-Server kopiert werden:

```
user@linux-client:~$ ssh-copy-id -i .ssh/ssh-key.pub user@root-server
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/ssh-
key.pub"
The authenticity of host 'root-server (192.168.178.101)' can't be
established.
ED25519 key fingerprint is
SHA256:F++5Vvf7HhmFJtf8qHXJzIpGNKyJWhyqYQ/ny8w4LtQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
user@root-server's password:

Number of key(s) added: 1
```

```
Now try logging into the machine, with:   "ssh 'user@root-server'"
and check to make sure that only the key(s) you wanted were added.

user@linux-client:~$
```

SSH-Konfiguration

Als nächstes sollte nun die SSH-Konfiguration auf dem Root-Server angepasst werden. Die Anmeldung mittels Passwort sollte deaktiviert und ausschließlich eine Anmeldung über einen SSH-Schlüssel zugelassen werden. Weiterhin werden stärke SSH-Host-Schlüssel generiert und zugelassen, sowie ein direkter Login vom „root“ verboten. Einige der nachfolgenden Punkte wurden aus dem Beitrag der Seite blog.buettner.xyz entnommen.

Schritt 1: Löschen und Erstellen der SSH-Host-Schlüssel:

```
root@root-server:~# rm -rf /etc/ssh/ssh_host_*key*
root@root-server:~# ssh-keygen -t ed25519 -f ssh_host_ed25519_key -N ""
</dev/null
Generating public/private ed25519 key pair.
Your identification has been saved in ssh_host_ed25519_key
Your public key has been saved in ssh_host_ed25519_key.pub
The key fingerprint is:
SHA256:OXE0gyPaGxJ+PSRF4k78llrqiq4lK+KTmCBc8tIyrQ4 root@root-server
The key's randomart image is:
+--[ED25519 256]--+
|      .oo.+      |
|    .oo.+. o     |
|   . ++=...      |
| . .+o+.o=       |
|. *  o.oS.       |
|o= +  .= .       |
|E.B   o          |
|*B . .           |
|B=+ ...          |
+----[SHA256]-----+
root@root-server:~# ssh-keygen -t rsa -b 4096 -f ssh_host_rsa_key -N ""
</dev/null
Generating public/private rsa key pair.
Your identification has been saved in ssh_host_rsa_key
Your public key has been saved in ssh_host_rsa_key.pub
The key fingerprint is:
SHA256:JWQoy2dPfeYOyt0HyTIYUnvpHuUB+o+VhPqC5pFYvOQ root@root-server
The key's randomart image is:
+---[RSA 4096]----+
|       .o        |
|    . .o. .      |
|   . o ..+.+     |
|    o.+ =o= *    |
|     o++SB O +   |
```

```
|     = o+ B O    |
|    . Eo = @ .   |
|      o.+ = + .  |
|     o.  .   .   |
+----[SHA256]-----+
root@root-server:~# mv -v ssh_host_ed25519_key* ssh_host_rsa_key* /etc/ssh/
Datei umbenannt 'ssh_host_ed25519_key' -> '/etc/ssh/ssh_host_ed25519_key'
Datei umbenannt 'ssh_host_ed25519_key.pub' ->
'/etc/ssh/ssh_host_ed25519_key.pub'
Datei umbenannt 'ssh_host_rsa_key' -> '/etc/ssh/ssh_host_rsa_key'
Datei umbenannt 'ssh_host_rsa_key.pub' -> '/etc/ssh/ssh_host_rsa_key.pub'
```

Schritt 2: Erstellen einer SSH-Gruppe und Nutzer hinzufügen:

```
root@root-server:~# groupadd ssh-user
root@root-server:~# usermod -a -G ssh-user user
```

Schritt 3: Löschen und Erstellen einer neuen SSH-Konfigurationsdatei „`/etc/ssh/sshd_config`“ mit folgenden Parametern und Werten:

```
# Lauschen auf dem SSH-Port
Port 22

# Setzen des Kex-Algorythmus
KexAlgorithms curve25519-sha256@libssh.org

# Ältere Protokoll-Version nicht zulassen
Protocol 2

# Verwendung der stärkeren Schlüssel
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

# Verbieten von Authentifizierung über Passwort
PasswordAuthentication no
ChallengeResponseAuthentication no
PubkeyAuthentication yes

# Nutzer aus der SSH-Gruppe zulassen
AllowGroups ssh-user

# Setzen der erlaubten Kryptografien und MAC-Algorythmen
Ciphers chacha20-poly1305@openssh.com
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-
etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com

# Logging
SyslogFacility AUTH
LogLevel INFO
```

```
# Wartezeit für kompletten Login
LoginGraceTime 120

# 'root'-Login mittls Passwort verbieten
PermitRootLogin prohibit-password

# Strikten Modus setzen
StrictModes yes

# Rhost- und Shost-Dateien des Benutzer nicht lesen
IgnoreRhosts yes

# Hostbasierte Authentifizierung verbieten
HostbasedAuthentication no

# Leere Passwörter verbieten
PermitEmptyPasswords no

# Grafik über SSH verbieten
X11Forwarding no

# Letzten Login anzeigen
PrintLastLog yes

# Nachricht des Tages nicht anzeigen
PrintMotd no

# Sitzung am Leben erhalten
TCPKeepAlive yes

# Gebietsschema umgehen
AcceptEnv LANG LC_*

# SFTP erlauben
Subsystem sftp /usr/lib/openssh/sftp-server

# Nutzung von PAM verbieten
UsePAM no
```

Nach der Anpassung muss der SSH-Dienst neu gestartet werden.

Kommando-Beschränkung

Eine weitere Verschärfung der Sicherheit ist es, dem Benutzer, der sich am Root-Server anmeldet nur einen einzigen Befehl zu erlauben, im nachfolgenden Fall „su“ (um Administratorrechte zu erlangen). Hierfür wird der nachfolgende Text am Anfang der Datei „`~/.ssh/authorized_keys`“ (damit ist die

Datei des Benutzers gemeint, der sich per SSH anmeldet) davorgesetzt:

```
command="/usr/bin/su -" ssh-ed25519 AAA...aazp7dgx3 user@root-server
```

SSH-Login

Jetzt kann die SSH-Verbindung mittels des privaten Schlüssels aufgebaut werden:

```
user@linux-client:~$ ssh -i .ssh/ssh-key user@root-server
Enter passphrase for key '.ssh/ssh-key':
Password:
root@root-server:~#
```

Sollte beim ersten Verbindungsaufbau die nachfolgende Fehlermeldung angezeigt werden, muss die Berechtigung des privaten Schlüssels angepasst werden:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0750 for '.ssh/ssh-key' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key ".ssh/ssh-key": bad permissions
```

Der nachfolgende Befehl setzt ausschließlich das Leserecht (400) für den Besitzer der Datei:

```
user@linux-client:~$ chmod 400 .ssh/ssh-key
```

Zwei-Faktor-Authentifizierung

Eine andere zusätzliche Absicherungsmethode für SSH ist die Zwei-Faktor-Authentifizierung mittels OTP. Hier wurden einige Punkte der Anleitung von Thomas Krenn entnommen.

Schritt 1: Installation der notwendigen Pakete:

```
root@root-server:~# apt install libpam-google-authenticator
```

Schritt 2: Ausführen des Google-Authenticator:

```
user@root-server:~$ google-authenticator

Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP secret
to Google:
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/
service@projekt-srs-
debian-1%3Fsecret%3DAXRKVTZD4QX7VKOQAYYIFIUX5Q%26issuer%3Dprojekt-srs-
```

```
debian-1
```

```
Your new secret key is: AXRKVTZT4QX8VKOQAXYIFDUX4Q
Enter code from app (-1 to skip): 835713
Code confirmed
Your emergency scratch codes are:
  76406132
  95407134
  53780518
  73229382
  93605280

Do you want me to update your "/home/user/.google_authenticator" file? (y/n)
y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the
server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If
you
experience problems with poor time synchronization, you can increase the
window
from its default size of 3 permitted codes (one previous code, the current
```

```
code, the next code) to 17 permitted codes (the 8 previous codes, the
current
code, and the 8 next codes). This will permit for a time skew of up to 4
minutes
between client and server.
Do you want to do so? (y/n) n

If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every
30s.
Do you want to enable rate-limiting? (y/n) y
user@root-server:~$
```

Schritt 3: Aktivieren der PAM-Module in der Datei „`/etc/pam.d/sshd`“:

```
...
# Standard Un*x authentication.
#@include common-auth

...

# Google Authenticator
auth required pam_google_authenticator.so
```

Hinweis: Die Zeile „`@include common-auth`“ muss auskommentiert werden, die Zeile „`auth required pam_google_authenticator.so`“ kann am Ende der Datei angehängt werden.

Schritt 4: Anpassen der SSH-Konfiguration in der Datei „`/etc/ssh/sshd_config`“:

```
ChallengeResponseAuthentication yes
UsePAM yes
AuthenticationMethods publickey,keyboard-interactive
```

Nach der Anpassung muss der SSH-Dienst neu gestartet werden.

SSH-Login mit OTP

Jetzt kann die SSH-Verbindung mit privatem Schlüssels und OTP aufgebaut werden:

```
user@linux-client:~$ ssh -i .ssh/ssh-key user@root-server
Enter passphrase for key '.ssh/ssh-key':
(user@root-server) Verification code:
Passwort:
root@root-server:~#
```