SSH-Zugang mit 2FA absichern

Am Beispiel eines Ubuntu 20.04 – Servers zeigen wir Ihnen, wie Sie ihren SSH-Zugang absichern und mit einem zweiten Faktor (2FA) zur Anmeldung versehen. Wechseln Sie dazu in den privilegierten Benutzermodus

```
sudo -s
```

und installieren das notwendige Modul für 2FA auf Ihren Server:

```
apt install libpam-google-authenticator
```

Verlassen Sie nun den sudo- Modus

```
exit
```

und initiieren das Modul

```
google-authenticator
```

Gehen Sie den Dialog wie folgt durch:

```
Do you want authentication tokens to be time-based (y/n) y
```

Sichern Sie sich die folgenden Daten sowie den QR-Code

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

Do you want to disallow multiple uses of the same authentication<br>token? This restricts you to one login about every 30s, but it increases<br>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.<br>In order to compensate for possible time-skew between the client and the server,<br>we allow an extra token before and after the current time. This allows for a<br>time skew of up to 30 seconds between authentication server and client. If you<br>experience problems with poor time synchronization, you can increase the window<br>from its default size of 3 permitted codes (one previous code, the current<br>code, the next code) to 17 permitted codes (the 8 previous codes, the current<br>code, and the 8 next codes). This will permit for a time skew of up to 4 minutes<br>between client and server.<br>Do you want to do so? (y/n) n

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

Nun ist 2FA bereits eingerichtet und wir passen noch SSH, als sudo Benutzer, an

```
sudo -s
```

Öffnen Sie die Datei

```
nano /etc/pam.d/common-auth
```

und fügen am Ende der Datei, in einer neuen Zeile die folgende Zeile ein:

```
auth required pam_google_authenticator.so nullok
```

Kopieren Sie nun die ssh-Konfiguration und bearbeiten diese dann

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

nano /etc/ssh/sshd_config
```

Setzen Sie

```
AuthenticationMethods keyboard-interactive
```

sowie

```
ChallengeResponseAuthentication yes
```

und starten den Dienst **ssh** neu.

```
service ssh restart
```

Des Weiteren empfehlen wir den Root-Zugang via SSH zu verhindern und auf bestimmte Benutzer einzuschränken.

```
...
AllowUsers benutzername
...
PermitEmptyPasswords no
PermitRootLogin no
...
```

Die gesamte sshd_config könnte in Verbindung mit 2FA so aussehen:

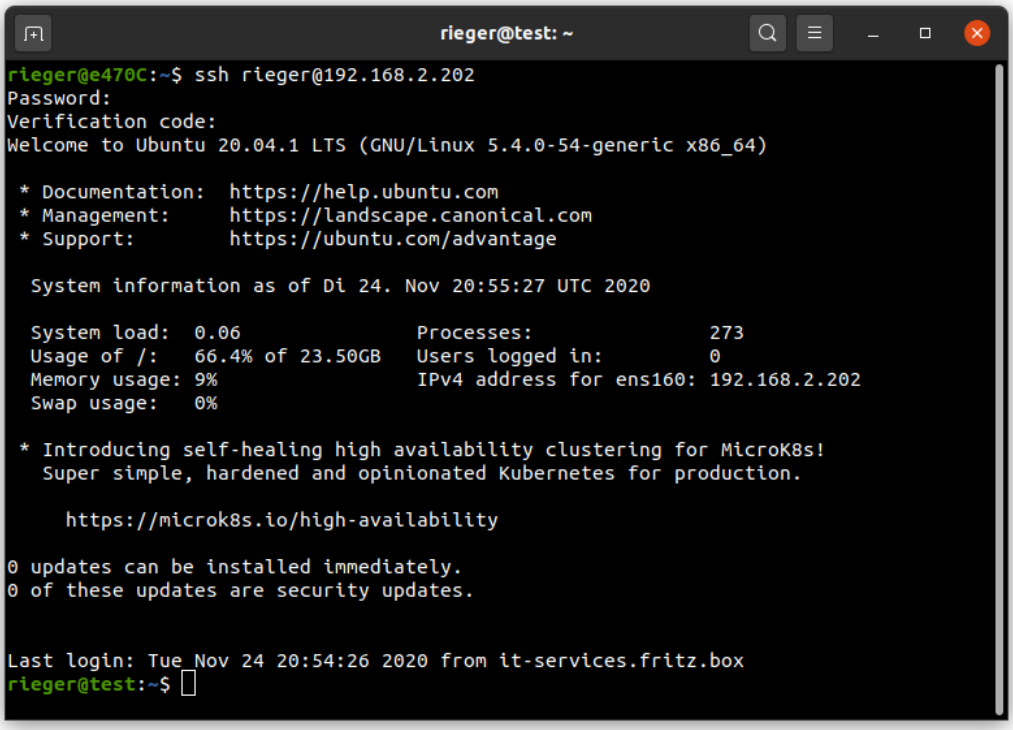
```
Port 22
Protocol 2
AcceptEnv LANG LC_*
AllowTcpForwarding no
AllowUsers benutzername(n)
AuthenticationMethods keyboard-interactive
ChallengeResponseAuthentication yes
ClientAliveInterval 300
ClientAliveCountMax 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
HostbasedAuthentication no
IgnoreRhosts yes
IgnoreUserKnownHosts yes
LogLevel INFO
LoginGraceTime 30s
MaxAuthTries 3
MaxSessions 3
PasswordAuthentication yes
PermitEmptyPasswords no
PermitRootLogin no
PrintMotd yes
PrintLastLog yes
Subsystem sftp /usr/lib/openssh/sftp-server
SyslogFacility AUTH
StrictModes yes
```

```
TCPKeepAlive yes

UseDNS yes

UsePAM yes

X11Forwarding no

X11DisplayOffset 10

X11UseLocalhost no
```

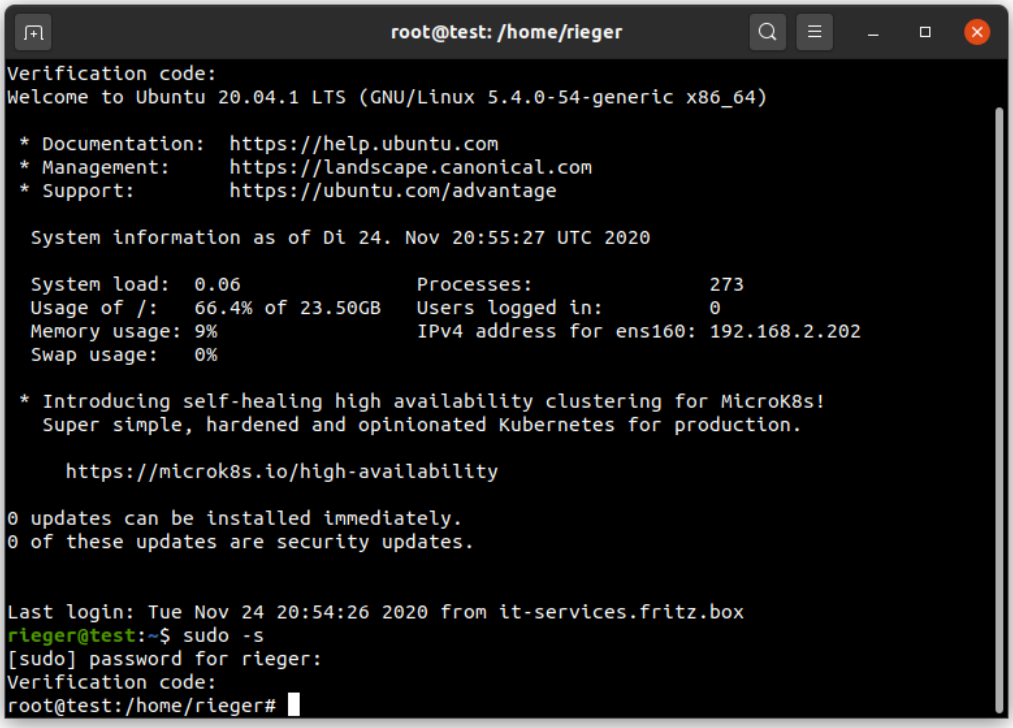
Wir deaktivieren zudem den Root-Account, indem wir als „normaler“ Benutzer diesen befehl absetzen

```
exit

sudo passwd -dl root
```

Die Anmeldung per SSH sowie der wechsel in den sudo-Modus stellt sich ab sofort wie folgt dar:



- ---



-

Die Absicherung Ihres SSH-Zugangs wurde erfolgreich abgeschlossen und so wünsche ich Ihnen viel Spaß! Über eine Spende würden sich meine Frau, meine Zwillinge und ich sehr freuen!