Den Raspberry Pi absichern

Den User pi bei Systembefehlen zum Passwort zwingen

Das Bequeme an dem Benutzer pi ist auch gleichzeitig sein verderben, wenn Fremde sich Zugriff verschafft haben. Er darf alle Systembefehle mit **sudo** ausführen und muss noch nicht einmal dafür ein Passwort eingeben. Da nützt das stärkste Passwort nichts, wenn dieses nicht zum Einsatz kommt. Das solltest du auch ändern, auch wenn die Passworteingabe dich wahrscheinlich nervt.

Öffne dazu im Terminal mit dem Befehl sudo vi /etc/sudoers.d/010_pi-nopasswd die sudoers-Einstellungen für pi.

Nun änderst du **pi ALL=(ALL) NOPASSWD: ALL** zu **pi ALL=(ALL) PASSWD: ALL** und speicherst die Datei entsprechend ab, STRG + X und danach Y oder J. Damit hat pi immer noch mittels **sudo** Berechtigung Systemkommandos auszuführen, aber es wird nun eine Passworteingabe erzwungen. An dieser Stelle ist es wichtig zu erwähnen, dass du genau auf die korrekte Schreibweise achten musst, da sonst die modifizierte Datei unbrauchbar ist.

Sicherheitsrelevante Updates automatisch ausführen

Die tägliche Arbeit von einem Systemadministrator ist und bleibt zu prüfen, ob es sicherheitsrelevante Updates für ein Betriebssystem oder Software gibt und ob diese auch installiert wurden. Gerade Sicherheitslücken, wie aktuell Log4j zeigen, wie wichtige diese Aufgabe ist. Updates für die diversen Pakete auf deinem Raspberry Pi wird es immer wieder geben, jedoch sind die sicherheitsrelevanten Patches die wohl wichtigsten. Wie du sicherlich bereits weißt, kannst du mit dem Befehl aus Code 7 deinen Raspberry Pi auf dem aktuellen Stand halte. Aber gerade diese Updates vergisst man sehr schnell und so werden sicherheitsrelevante Updates nicht auf deinen Raspberry Pi installiert.

sudo apt update && sudo apt dist-upgrade -y

An dieser Stelle hilft das Tool **unattended-upgrades**, welches bei Raspberry Pi OS (Stand 12/2021) nicht automatisch mitinstalliert ist. Mit Code 8 im Terminal kann dieses schnell nachinstalliert werden.

```
sudo apt install unattended-upgrades
```

Damit beginnt das Tool aber noch nicht seinen Dienst, sondern es muss noch aktiviert werden. Tippe dazu ins Terminal den Befehl aus Code 9 ein.

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

Code 9: unattended-upgrades nachkonfigurieren

Anschließend wird dir im Terminal die Frage gestellt, ob du **stabile sicherheitsrelevante Updates automatisch installieren möchtest**, siehe Abbildung 6, was du mit **Ja** bestätigst.

SSH-Server ein bisschen sicherer machen

Der Raspberry Pi ist ein super System und seit dem Raspberry Pi 4 ist dieser auch ein ausgezeichneter Ersatz als Desktop-PC. Doch wer z.B. einen kleinen Cloudserver mit dem Pi betreibt, der wird von außen eher per SSH auf das System zugreifen wollen, um diesen zu verwalten. Ohne z.B. einen VPN-Tunnel ist es eher untypisch sich via Remotedesktop oder VNC auf sein System zu

schalten. Gerade bei den SSH-Grundeinstellungen kannst du an zwei Stellen optimieren. Zum einen kannst du den SSH-Port ändern und auf jeden Fall solltest du den Root-Zugriff unterbinden. Beide Einstellungen müssen in der sshd_config vorgenommen werden, welche du mit Code 7 im Terminal öffnest.

```
sudo vi /etc/ssh/sshd_config
```

Um den SSH-
Port zu ändern, suchst du nach der Zeile **#Port 22**
und änderst diesen z.B. zu **Port 777**.
Speicherst du dies ab und startest den Dienst oder den Raspberry Pi neu, ist der SSH-Zugang nur noch über den Port 777 statt über den Port 22
möglich. Gerade diese Änderung ist insofern hilfreich, da Leute, die gerade erst mit dem hacken beginnen,
hier schon bei Angriffen via SSH scheitern. Geübte Hacker werden jedoch schnell den Port finden, da es diverse Portscanner gibt,
die Ports und ihre Funktion erfragen. Wenn du den Port geändert hast, musst du bei PuTTY oder deinem SSH-Einwahltool darauf achten,
diesen mit anzugeben

Eine wesentlich klügere Anpassung ist der Parameter **#PermitRootLogin** mit dem Standardwert **prohibit-passwort** anzupassen. Vom Prinzip sollte und darf root keinen Zugriff via SSH erhalten, da es das schlimmste ist, was dir bzw. uns passieren kann. Der Benutzer root darf alles auf dem System und Hacker können so den größtmöglichen Schaden auf dem Raspberry Pi und deinem Netzwerk anrichten. Daher sollte dies Zeile wie folgt aussehen.

PermitRootLogin no

Mittels Code 11 im Terminal können die gespeicherten Änderungen direkt angewendet werden. Solltet ihr die Änderungen über einen SSH-Zugang gemacht haben, so bleibt eure SSH-Verbindung in den meisten Fällen bestehen.

```
sudo service ssh restart
```

SSH nur mit Private key file

SSH ist in Kombination mit einem starken Passwort schon eine sichere Sache, doch man weiß nie, ob das Passwort wirklich sicher ist. Besser wäre es doch, wenn es eine absolut sichere Methode geben würde. Diese sichere Methode heißt **Private key file authentification** die mit jedem gängigen

Betriebssystem möglich ist. Der Grundgedanke ist der, dass du ein Schlüsselpaar aus public key und private key erstellst. Ersteres ist bildlich gesprochen das Schloss, welches mittels des eigenen Schlüssels, dem private key, dir den Zugang gewährt.

Zunächst erstellst du dir ein Schlüsselpaar. Eine ausführliche Anleitung dazu findest du hier. Die Option mit dem **Key passphrase** ist nur dann wichtig, wenn du die größtmögliche Sicherheit willst. In unserem Beispiel lässt du **Key passphrase** und **Confirm passphrase** einfach leer. Damit musst du später beim Einwählen mit dem **Private key** nicht noch den **Key passphrase**, nicht zu verwechseln mit deinem Passwort,eingeben. Speichert euch sowohl den public key und den private key.

Ohne PuTTYGen oder das Tool zu Generierung des Schlüsselpaares zu Schließen erstellt ihr auf dem Raspberry Pi nun eine neue Datei im Unterordner .ssh eures Benutzers pi, mit dem Befehl aus Code 12.

```
mkdir /home/pi/.ssh
vi /home/pi/.ssh/authorized_keys
```

Ggf. ist dieser Ordner und die Datei schon vorhanden, sofern du mittels SSH auf dem Raspberry Pi gearbeitet hast.

Nun kopierst du den Public key von PuTTYGen und fügst diesen in die eben geöffnete **authorized_keys** – Datei ein, siehe Abbildung 8, und speichert diese ab.

Als nächstes, damit die Authentifikation auch funktioniert, müssen die Berechtigungen noch angepasst werden. Dies geschieht mit den beiden Befehlen aus Code 13.

```
sudo chmod 700 /home/pi/.ssh
sudo chmod 600 /home/pi/.ssh/authorized_keys
```

Im letzten Schritt stellen wir den SSH-Server so ein, dass nur noch Zugriff mit dem **private key** möglich ist. Öffne dazu noch einmal die Datei sshd_config im Terminal mit dem Befehl aus Code 14.

```
sudo nano /etc/ssh/sshd_config
```

Suche die Zeile **#PasswordAuthentication yes** und ändere diese zu **PasswordAuthentication no**

Speichere die Datei wieder ab und starte den Dienst mit

```
sudo service ssh restart
```

neu.

Den Raspberry Pi mit der Firewall UFW ausrüsten

Jedes Gerät, welches über das Internet erreichbar ist, sollte durch eine starke Firewall abgesichert werden. Virenschutz ist genauso wichtig, aber mit einer gut eingestellten Firewall können einige Angriff im Keim erstickt werden und viele Viren gibt es für Linux (noch) nicht. Linux hat hierfür die Firewall UFW, was die Abkürzung für Uncomplicated Firewall ist. Unkompliziert ist die Firewall wirklich, daher solltest du diese auf deinem Raspberry Pi mit dem Befehl aus Code 16 installieren.

```
sudo apt install ufw
```

Die Firewall UFW installieren

Wenn der Paketmanager UFW installiert wurde, ist der Dienst noch nicht aktiv! Das ist auch gut so, da sonst ggf. unsere SSH-, Remotedesktop- oder VNC-Verbindung gekappt werden könnte. Damit UFW mit der Arbeit beginnen kann, müssen wir erst einmal Regeln erstellen, welche Ports überhaupt erlaubt sind. Für gängige Anwendungen reichen die in Tabelle 1 freigegebenen Ports, wobei die Liste hier nicht vollständig ist und je nach verwendetem Dienst stark abweicht.

Pos	Port	Beschreibung
1	80	Standardport für Webadresse mittels http
2	443	Standardport für Webadresse mittels https
3	22	Der Standardport für SSH-Zugang, ggf. musst du diesen durch deinen gewählten in der sshd_config ersetzen
4	5900	Standardport für VNC-Verbindungen
5	3389	Standardport für Remotedesktopverbindungen

Tabelle 1: Ports für häufig genutzte Dienste

Wenn du weitere Dienste auf dem Raspberry Pi benutzt, musst du die entsprechenden Ports für den Dienst ermitteln und freigeben.

Damit nun z.B. ein Webserver auch übers Internet erreichbar ist, davon ausgehen das am Router die Ports auch freigegeben und weitergeleitet werden, reichen die Befehle aus Code 17.

```
sudo ufw allow 80
sudo ufw allow 433
```

Damit sind die Ports 80 und 443 schon einmal freigeschaltet und über das Internet und lokal im Netzwerk erreichbar. Willst du hingegen diese Ports nicht zulassen, dann geht dies mit den Befehlen aus Code 18.

```
sudo ufw deny 80
```

```
sudo ufw deny 433
```

Willst du auf Nummer sicher gehen, kannst du per default alle Ports mittels des Befehles aus Code 18 sperren, musst dann aber im Gegenzug alle benötigten Ports einzelnd wieder freischalten.

```
sudo ufw default deny
```

```
Hast du alles eingestellt, muss nun noch UFW gestartet und bei jedem Neustart aktivi
ert werden. Dies geschieht mit dem Befehl aus
```

```
sudo ufw enable
```

Code 20: UFW aktivieren

Sofern du alles richtig eingestellt hast, sollte die Firewall nun laufen und deine Ports sollten entsprechend freigeschaltet oder blockiert sein. Solltest du Einstellungen per SSH vorgenommen haben und nach der Aktivierung keiner Verbindung mehr bekommen, ist wahrscheinlich der Port nicht freigegeben, den wir weiter oben geändert haben. Die Firewall kannst du über den Befehl aus Code 20 wieder deaktivieren und dann musst noch einmal deine Ports überprüfen.

```
sudo ufw disable
```

UFW deaktivieren

Da UFW sehr mächtig ist und hier in der Kürze nicht alles erklärt werden kann, empfehle ich dir die Wiki-Seite von ubuntuusers.de, hier kannst du alles wichtige nachlesen.

Eindeutige ID: #1015
Verfasser: n/a
Letzte Änderung: 2022-03-21 10:31