# Linux allgemein

In diesem Kapitel dokumentiere ich Informationen rund um Linux, die eher allgemeiner Natur sind und nichts im speziellen mit Arch Linux zu tun haben.

# Warnmeldung "kauditd_printk_skb: .. callbacks suppressed" im Protokoll

Diese Warnmeldung gibt die Anzahl der Unterdrückten Warnmeldungen, die vom Netzwerk kommen, aus.

Das ist ein Mechanismus des Kernels um das System u.a. vor  "Denial-of-Service"-Angriffen zu schützen.

```
sudo sysctl -a | grep net.core.message_
```

Es gibt dazu zwei Parameter, die dieses Mechanismus steuern:

```
net.core.message_burst
net.core.message_cost
```

Erläuterungen dazu:

*Diese Parameter werden verwendet, um die Warnmeldungen einzuschränken, die vom Netzwerkcode in das Kernelprotokoll geschrieben werden.  Sie erzwingen eine Ratenbegrenzung, um einen Denial-of-Service-Angriff unmöglich zu machen.  Ein höherer* ***message_cost*** *-Faktor führt zu weniger Nachrichten, die geschrieben werden.  **Message_burst** steuert, wann Nachrichten verworfen werden.  Die Standardeinstellungen begrenzen Warnmeldungen auf eine alle fünf Sekunden.*

Die aktuellen Werten können wie folgt ermittelt werden:

```
sudo sysctl -a | grep net.core.message_
```

Möchte man die Werte ändern geht das mit folgenden Befehlen:

```
sudo sysctl -w net.core.message_cost=0
```

```
sudo sysctl -w net.core.message_burst=0
```

**Achtung:** In einer produktiven Umgebung sollte man diese Werte nicht anpassen!!

# Private Key und Zertifikat mit openssl erstellen

Für die verschlüsselte Übertragung von Syslog-Nachrichten an einen zentralen Log-Server (Graylog ) werden private Schlüssel und Zertifikate benötigt.

Diese können wie folgt erstellt werden:

**Privater Schlüssel**

```
openssl genrsa -des3 -out privat.key 4096
```

Es wird dazu ein Passwort abgefragt, das später beim Erzeugen von Zertifikaten benötigt wird.

Zertifikat

```
openssl req -x509 -new -nodes -key private.key -sha256 -days 9999 -out zertifikat.pem
```

# Eigene IP-Adresse(n) ermitteln

Unter Linux kann man mit folgendem Befehl die eigene(n) IP-Adresse(n) ermitteln:

```
ifconfig
```

# Speicherverbrauch auf Datenträgern analysieren

## Auf einem Server ohne GUI

Auf Systemen ohne GUI kann man zu Ermittlung der Belegung der Datenträger das Tool Ncdu nutzen. Es wird im Terminal wie folgt ausgeführt:

```
sudo ncdu
```

Direkt nach dem Start scannt das Tool ab dem aktuellen Verzeichnis sämtliche Dateien im aktuellen und in den Unterverzeichnissen und stellt das Ergebnis im Terminal dar.

Man kann mit den Cursor-Tasten und der Enter-Taste durch die Verzeichnisse navigieren.

## Auf Systemen mit Plasma-Desktop (KDE)

Auf Systemen mit Plasma-Desktop kann das Tool Filelight dazu genutzt werden, Datenträger hinsichtlich Speicherverbrauch zu analysieren.

# Eigenes Script mittels systemd starten

Es ist relativ einfach ein eigenes Script mit dem systemd beim Systemstart starten zu lassen. Dieses geschieht mit dem bekannten Befehl "sytemctl start".

Zunächst muss eine Service-Datei für den Dienst angelegt werden:

```
[Unit]
Description=Wireguard Verbindung WG1 aktiv halten

Wants=network.target
After=syslog.target network-online.target

[Service]
Type=simple
ExecStart=python3 /home/thomas/scripts/python/wg1_active.py
Restart=on-failure
RestartSec=10
KillMode=process

[Install]
WantedBy=multi-user.target
```

Mit dieser Service-Datei wird ein von mir geschriebenes Python-Script beim Systemstart gestartet.

Mit dem Script wird eine Wireguard-Verbindung "am Leben" gehalten.

Die Service-Datei darf nicht ausführbar sein.

Ort für die Service-Datei (unter Arch Linux):

```
/etc/systemd/system
```

Quelle: https://secinfinity.net/so-fuhren-sie-ein-linux-programm-beim-start-mit-systemd-aus/

# Standard-Shell festlegen

Möchte man z.B. zsh als Standard-Shell festlegen, gibt man folgenden Befehl ein (zsh muss natürlich installiert sein):

```
chsh -s /usr/bin/zsh
```

Da das eine Einstellung für einen Benutzer ist, muss der Befehl von dem Benutzer ausgeführt werden, der die Standard-Shell ändern möchte.

# SSD-Temperatur loggen

An einem RaspberryPI 4+ ist eine NVME-SSD über einen USB-Adapter angeschlossen. Es die Temperatur der SSD in eine Logdatei geschrieben werden.

Die folgende Lösung hat die Google-KI erstellt:

```
#!/bin/bash
# Pfad zur Logdatei (anpassen!)
LOGFILE="/home/DEIN_BENUTZERNAME/ssd_temp.log"
DEVICE="/dev/sda"

# Zeitstempel
DATE=$(date "+%Y-%m-%d %H:%M:%S")

# Extrahiert nur die Zahl aus der Zeile "Temperature: 46 Celsius"
TEMP=$(sudo smartctl -a $DEVICE | grep "^Temperature:" | awk '{print $2}')

# In Datei schreiben
echo "$DATE - SSD Temp: $TEMP°C" >> "$LOGFILE"
```

Das erstellte Bash-Script muss ausführbar gemacht werden:

```
chmod +x ssd_temp_log.sh
```

## Automatisierung via cron-job.

Crontab öffnen:

```
sudo crontab -e
```

Zeile am Ende einfügen, damit die Temperatur alle 5 Minuten in die Logdatei geschrieben wird:

```
*/5 * * * * /pfad/zu/deinem/ssd_temp_log.sh
```

## Aktuelle Temperatur im Terminal alle 5 Sekunden anzeigen (immer nur der aktuelle Wert)

```
watch -n 5 "sudo smartctl -a /dev/sda | grep '^Temperature:'"
```

# SSH-Zugriffe (Port 22) nur aus dem internen Netz zulassen

Ich habe eine VPN-Verbindung zwischen einem externen Server und einem Raspberry Pi in meinem Heimnetz. Dadurch kann man sich vom Server aus per SSH mit dem Raspberry Pi verbinden (natürlich müssen Benutzer und Passwort ebenfalls bekannt sein).

Um dies zu verhindern kann man mit Hilfe der IPTables-Firewall eine Regel auf dem Raspberry Pi erstellen, die diese verhindert.

Erster Schritt: Zugriff vom internen Netz zulassen:

```
iptables -A INPUT -s 192.168.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
```

Zweiter Schritt: Sämtliche Zugriffe von Außen sperren:

```
iptables -A INPUT -p tcp -m tcp --dport 22 -j DROP
```

Diese Befehle ändern die Einstellungen zunächst temporär. Damit dieses permanent aktiv ist muss unter Ach Linux noch folgender Befehl eingeben werden:

```
iptables-save -f /etc/iptables/iptables.rules
```

---

Quellen:


1. https://www.gutefrage.net/frage/wie-kann-ich-meinen-ssh-port-von-aussen-blockieren-mit-iptables
2. https://wiki.archlinux.org/title/iptables

# Kommandozeilen Befehle

Um den geeigneten Befehl für die Kommandozeile zu finden, kann folgende Seite sehr gut helfen:

CMD Generator

# Systemd Service-Datei anpassen

Eine Systemd Service-Datei soll nicht direkt angepasst werden.

Es gibt dafür eine Override-Möglichkeit. Mit dieser lassen sich Inhalte in der Service-Datei überschreiben bzw. anpassen.

Mit folgendem Kommando erzeugt und editiert man eine entsprechende Override-Datei (Beispiel für zoraxy):

```
sudo systemctl edit zoraxy
```

Die Override-Datei wird mit dem Inhalt der aktuellen Service-Datei angelegt. Die Zeilen sind aber alle auskommentiert.

Es ist nicht mehr zulässig an den aus der Service-Datei übernommen Inhalten direkt etwas anzupassen. Stattdessen soll oberhalb des auskommentierten Bereiches der Block mit den angepassten Zeilen eingefügt werden. Beispiel:

```
### Editing /etc/systemd/system/zoraxy.service.d/override.conf
### Anything between here and the comment below will become the contents of the drop-in file

[Service]
ExecStart=
ExecStart=/usr/bin/zoraxy -log /var/log/zoraxy -port=:49999 -fastgeoip=true 2>&1 | logger &

### Edits below this comment will be discarded



### /usr/lib/systemd/system/zoraxy.service
# [Unit]
# Description=Zoraxy Reverse Proxy Server
#
# # start not befor network is online
# After=network-online.target
# Wants=network-online.target
```

```
#
# [Service]
# Type=simple
#
# # start zoraxy as root
# User=root
# Group=root
#
# # folder where zoraxy config & runtime data are located
# WorkingDirectory=/usr/lib/zoraxy
#
# # use absolute path for zoraxy always
# # options:
# # -fastgeoip=true               faster GeoIP-lookup, but more RAM ...
# #                               faster GeoIP-lookup, but uses more RAM ...
# # -log /var/log/zoraxy          log to file / dont create local log folder
# # -port=:8008                   use port instead of defalut (8000)
#
# ExecStart=/usr/bin/zoraxy -log /var/log/zoraxy -port=:8000 2>&1 | logger &
# ExecStop=/usr/bin/kill "$MAINPID"
#
#
# [Install]
# WantedBy=multi-user.target
```

Mit Zeile 4 und 5 passe ich die Zeile 35 entsprechend meinen Vorstellungen an.

Im Standard ist der Editor zum anpassen "nano". Mir persönlich gefällt der Editor "micro" ( https://micro-editor.github.io/ besser. Nachdem man "micro" installiert hat, kann man mit folgendem Aufruf auch "mirco" als Editor nutzen:

```
sudo env EDITOR=micro systemctl edit zoraxy
```