WireGuard – VPN-Schlüssel

Installation (Debian)

```
sudo apt update
```

```
sudo apt install wireguard
```

Version prüfen:

```
wg --version
```

Prinzip

WireGuard ist ein schlankes VPN-Protokoll. Jeder Peer besitzt ein privates und ein öffentliches Schlüsselpaar. Die Schlüssel dienen der gegenseitigen kryptografischen Identifikation und der Ableitung von Sitzungsschlüsseln. WireGuard verwendet moderne festgelegte Kryptografie.

Schlüsselpaar erstellen

Private Schlüssel nicht in gemeinsam lesbaren Verzeichnissen erzeugen. Beispiel in einem geschützten Arbeitsverzeichnis:

```
umask 077
```

Privaten Schlüssel erzeugen:

```
wg genkey > privatekey
```

Öffentlichen Schlüssel daraus ableiten:

```
wg pubkey < privatekey > publickey
```

Die Datei `privatekey` ist geheim; `publickey` wird auf der jeweiligen Gegenstelle hinterlegt.

Peer-Konfiguration

Eine typische Konfiguration enthält auf der lokalen Seite `PrivateKey` und für jeden Peer dessen `PublicKey` sowie `AllowedIPs`. Die konkreten IP-Bereiche und Endpunkte hängen vom Netzwerk ab.

Verbindung starten und beenden

Voraussetzung ist eine korrekt eingerichtete Konfiguration `/etc/wireguard/wg0.conf`. `wg0` ist ein Beispielname.

Starten:

```
sudo wg-quick up wg0
```

Beenden:

```
sudo wg-quick down wg0
```

Die Nutzdaten werden automatisch ver- und entschlüsselt.

Status prüfen

```
sudo wg show
```

Die Ausgabe zeigt unter anderem Peers, Handshakes und übertragene Daten. Sie ist für die Diagnose hilfreich, kann aber Netzwerkdetails offenlegen.

In o.a. Anzeige sind der öffentliche Schlüssel sowie die IP-Adresse zu sehen, aber keine weiteren Client-Informationen.

Eigenes Anzeige-Script

1. Gerätenamen in wg0.conf eintragen (Beispiel):

```
sudo nano /etc/wireguard/wg0.conf
```

```
[Peer]
# Client: Notebook
PublicKey = ...
AllowedIPs = 10.10.0.2/32

[Peer]
# Client: Smartphone
PublicKey = ...
AllowedIPs = 10.10.0.3/32
```

Wichtig ist, dass # Client: jeweils innerhalb des zugehörigen [Peer]-Abschnitts steht.

2. Skript erstellen
Auf dem WireGuard-Server:

```
sudo nano /usr/local/sbin/wg-status
```

Folgenden Inhalt einfügen:

```
#!/usr/bin/env bash
set -euo pipefail

CONFIG="/etc/wireguard/wg0.conf"
INTERFACE="wg0"

if [[ $EUID -ne 0 ]]; then
    echo "Bitte mit sudo ausführen: sudo wg-status" >&2
    exit 1
fi

if [[ ! -r "$CONFIG" ]]; then
    echo "Konfiguration nicht lesbar: $CONFIG" >&2
    exit 1
fi

declare -A CLIENTS

# Gerätenamen und PublicKeys aus der Konfiguration lesen.
while IFS=$'\t' read -r key name; do
    [[ -n "$key" ]] && CLIENTS["$key"]="$name"
done < <(
    awk '
        /^\[Peer\][[:space:]]*$/ {
            if (peer && key != "") print key "\t" name
            peer=1
            key=""
            name="Unbekannt"
            next
        }
        /^\[/ {
            if (peer && key != "") print key "\t" name
            peer=0
            next
        }
        peer && /^[[:space:]]*#[[:space:]]*Client:/ {
            sub(/^[[:space:]]*#[[:space:]]*Client:[[:space:]]*/, "")
            name=$0
            next
        }
        peer && /^[[:space:]]*PublicKey[[:space:]]*=/ {
            sub(/^[[:space:]]*PublicKey[[:space:]]*=[[:space:]]*/, "")
            sub(/[[:space:]]*$/, "")
            key=$0
        }
        END {
            if (peer && key != "") print key "\t" name
```

```
        }
    ' "$CONFIG"
)

echo "WireGuard-Status: $INTERFACE"
echo "================================================================"

# Aktuelle Peer-Daten abfragen.
while IFS=$'\t' read -r key psk endpoint ips handshake rx tx keepalive; do
    [[ -n "$key" ]] || continue

    name="${CLIENTS[$key]:-Unbekannt}"

if [[ "$handshake" =~ ^[0-9]+$ ]] && (( handshake > 0 )); then
    age=$(( $(date +%s) - handshake ))
    (( age < 0 )) && age=0

    days=$(( age / 86400 ))
    hours=$(( (age % 86400) / 3600 ))
    minutes=$(( (age % 3600) / 60 ))
    seconds=$(( age % 60 ))

    parts=()
    (( days > 0 )) && parts+=("$days Tage")
    (( hours > 0 )) && parts+=("$hours Stunden")
    (( minutes > 0 )) && parts+=("$minutes Minuten")
    (( seconds > 0 || ${#parts[@]} == 0 )) && parts+=("$seconds Sekunden")

    handshake_text="vor ${parts[*]}"
else
    handshake_text="Noch kein Handshake"
fi
    rx_mib=$(awk -v n="$rx" 'BEGIN {printf "%.2f", n/1048576}')
    tx_mib=$(awk -v n="$tx" 'BEGIN {printf "%.2f", n/1048576}')

    printf '\nClient:       %s\n' "$name"
    printf 'PublicKey:    %s\n' "$key"
    printf 'VPN-IP:       %s\n' "$ips"
    printf 'Endpoint:     %s\n' "$endpoint"
    printf 'Handshake:    %s\n' "$handshake_text"
    printf 'Empfangen:    %s MiB\n' "$rx_mib"
    printf 'Gesendet:     %s MiB\n' "$tx_mib"

done < <(wg show "$INTERFACE" dump | tail -n +2)
```

3. Ausführbar machen

```
sudo chmod 755 /usr/local/sbin/wg-status
```

4. Link ins Homeverzeichnis

```
ln -s /usr/local/sbin/wg-status ~/wg-status
```

5. Aufrufen

```
sudo ~/wg-status
```

Geräteverwaltung

Für jedes Gerät ein eigenes Schlüsselpaar verwenden. Geht ein Gerät verloren, wird dessen Peer auf den Gegenstellen entfernt und gegebenenfalls ein neuer Schlüssel eingerichtet. Der Austausch eines Schlüssels erfordert die Aktualisierung des zugehörigen öffentlichen Schlüssels auf den Gegenstellen.

Sicherheit

Der private Schlüssel darf nicht öffentlich werden. `AllowedIPs` steuert sowohl Routing als auch die Zuordnung zulässiger IP-Adressen zu Peers; eine sorgfältige Konfiguration ist notwendig. WireGuard ersetzt keine Firewall-Regeln oder Zugriffskontrollen auf Anwendungsebene.

Zurück zur Übersicht

From:
https://wiki.bluegnu.de/ - **wiki**

Permanent link:
https://wiki.bluegnu.de/doku.php/open:it:secure:wireguard

Last update: **2026/11/15 22:07**