

# Virtual Private Networks mit Wireguard

Freier Software Abend  
24. Februar 2021

Harald Weidner  
hweidner@gmx.net

# Virtual Private Network (VPN)

- Virtuelles gesichertes Netzwerk auf Basis eines physischen (ggf. unsicheren) Netzes
- Basiert auf Kryptographie und Netzwerktechnik

# VPN Einsatzmöglichkeiten

- Site-To-Site – Verbindung von Firmennetzen
  - Fully Meshed (jede Außenstelle direkt mit jeder vernetzt)
  - Hub and Spoke (sternförmig, zentraler Internetzugang)
- Node-to-Site (Road Warrior)
- Node-to-Node
- Öffentliches VPN
  - Verschlüsselung trotz offenem WLAN
  - Umgehung von Geoblocking (z.B. Videostreams)
  - Anonymität

# VPN History

- IPsec
  - Ursprünglich aus IPv6 Standardisierung
  - z.B. FreeS/WAN, {Strong|Open|Libre}Swan, KAME, isakmpd, Cisco VPN, vpnc
- SSL-VPNs
  - Anwendungen im Browser oder HTTPS Tunnel
  - z.B. Cisco AnyConnect, openconnect
- Weitere Lösungen
  - z.B. CIPE, PPTP, OpenVPN

# Probleme vieler VPN Lösungen

- Komplexität
  - z.B. OpenVPN ~600k, StrongSwan ~400k LOC
  - Code Audit schwierig
- Schwierige Konfiguration
- Veraltete Annahmen zu Netztopologie (v.a. IPsec)
  - Dynamische IP-Adressen, Roaming, NAT, Firewalls
- Performance der Implementierung
  - Ältere Kryptoverfahren (z.B. lange RSA-Schlüssel)
  - VPN im Userspace (Kontextwechsel)

# Wireguard

- Relatives junges VPN Verfahren (seit 2017)
- Autor: Jason A. Donenfeld / Edge Security
- Im Kernel seit Linux 5.6 (30. März 2020)
- Backports für ältere Kernel (ab Linux 3.10)
- Userspace-Implementierungen für andere Betriebssysteme
  - Windows, MacOS, Android, IOS, BSD u.a.
- Moderne Kryptographie-Verfahren und Protokolle
- Offene Spezifikation, von Kryptologen untersucht
- ~4800 LOC

# Wireguard – Technik

- Kommunikation über UDP
  - Standardport 51820
  - Keine getrennten Kanäle für Steuerung und Nutzdaten
  - IP-over-UDP, IPv4 und IPv6 beliebig kombinierbar
- Unter Linux vollständig im Kernel
  - Kernelmodul: wireguard
  - Interfaces normalerweise wg0, wg1, wg2, ...
  - CLI Tools: wg, wg-quick
  - Einbindung in Systemd

# Wireguard – Kryptographie

Verwendung von modernen, effizienten und von Kryptologen analysierten Verfahren:

- Noise protocol framework (Netzwerkprotokolle)
- Curve25519 (ECDH Schlüsselaustausch)
- ChaCha20 (Stromchiffre)
- Poly1305 (Message Authentication Code)
- BLAKE2 (Kryptographische Hashfunktionen)
- SipHash (Hash für Dictionaries)
- HKDF (Schlüsselerzeugungsfunktion)



# Wireguard – Protokoll

- Weitgehend zustandslos
  - Ein Tunnelpaket pro Nutzdatenpaket
  - ggf. mit vorherigem Handshake
- Handshake / Rekey alle zwei Minuten
  - Forward Secrecy
- Sequenznummern und Zeitstempel
  - Schutz vor Replay-Angriffen
- Cookies
  - Schutz vor Überlastung / DDoS

# Wireguard Schlüsselerzeugung

- 256 Bit Schlüssel (ECDH Curve25519, entspricht 3072 Bit RSA)
- BASE-64 Codierung

```
# wg genkey > secretkey  
# cat secretkey  
cKask3Ee2C9hlZnXw3gqPCwlydSJThV0fBc+HfsZnF8=  
  
# wg pubkey < secretkey > publickey  
# cat publickey  
pavY1sXnXSmimt8XA6RL4G6NWaD6rJAXgLmq1ltg8nA=
```

# Wireguard Netzwerk-Integration

- Kryptokey Routing
  - Verknüpfung von Routen mit Public Keys
- Einkommende Pakete
  - Ingress Filter: Pakete werden verworfen, wenn Signatur nicht zu IP-Adresse passt
- Ausgehende Pakete
  - Wahl des Public Key passend zur Route

# Wireguard Konfigurationsbeispiel

```
# Local site (DE Office Cologne)
[Interface]
Address      = 10.49.50.1/24
ListenPort   = 52480
PrivateKey    = MIBWyPP0ZL9zsQ2TryjI4nz0b3sISUt/1LuTpcSmvXA=

# CH Office (Zurich)
[Peer]
PublicKey     = uHyx1GWrkny7j+iyamIsY1IreYpJfCXnnfnbMQue0Cc=
Endpoint      = vpn.ch.example.com:52480
AllowedIPs    = 10.41.80.0/24

# UK Office (London)
[Peer]
PublicKey     = GrwU+Nqpx9Kfqg0H3ADi/GlWZ9UtXq0cM00X+C7FXDQ=
Endpoint      = vpn.uk.example.com:52480
AllowedIPs    = 10.44.1.0/24

# Administrator
[Peer]
PublicKey     = fyzGcq6Yek/EFxZCfu961cttRRb0UnVDSm0G271JG8=
AllowedIPs    = 10.49.50.100/32
```

# Wireguard Konfiguration

- Konfigurationsdatei
  - /etc/wireguard/<Interface>.conf (z.B. wg0.conf)
- Aktivierung

```
# wg-quick up wg0  
# ping 10.41.80.1  
# wg-quick down wg0
```

- Einbindung in Systemd

```
# systemctl start wg-quick@wg0.service  
# systemctl enable wg-quick@wg0.service  
# systemctl stop wg-quick@wg0.service  
# systemctl disable wg-quick@wg0.service
```

# Alternative Konfiguration

```
# ip link add dev wg0 type wireguard
## -oder-
# wireguard-go wg0

# ip addr add 10.49.50.1/24 dev wg0

# wg set wg0 listen-port 52480 \
    private-key /etc/wireguard/privatekey

# wg set wg0 peer uHyx1GWrkny7j+iyamIsY1IreYpJfCXnnfnbMQue0Cc= \
    endpoint vpn.ch.example.com:54280 \
    allowed-ips 10.41.80.0/24

# wg set wg0 peer GrwU+Nqpx9Kfqg0H3ADi/GlWZ9UtXq0cM00X+C7FXDQ= \
    endpoint vpn.uk.example.com:54280 \
    allowed-ips 10.44.1.0/24

# ip link set wg0 up
```

# Weitere Konfigurationsoptionen

Option	Bedeutung
DNS	DNS-Server während VPN aktiv
MTU	MTU explizit setzen
PresharedKey	Zusätzliches Secret für Key Exchange (Post Quantum)
PersistentKeepalive	Intervall für regelmäßige Pakete
PreUp, PostUp, PreDown, PostDown	Hooks für eigene Kommandos
Table	Explizite Angabe der Routingtabelle (Policy based Routing)
SaveConfig	Speichern der Konfiguration beim Shutdown

# Nachteile / Limits von Wireguard

- Unterstützt nur UDP
  - Manchmal an Firewalls nicht erlaubt
  - Keine Standard-Proxies (CONNECT) verwendbar
- Nur Tunnel (L3), keine Bridge (L2)
- Keine Protokoll für Anonymität
  - Vorheriger Schlüsselaustausch erforderlich
  - Fixe IP-Adressen / Ranges für alle Teilnehmer
  - Dauerhafte Speicherung aller letzten Endpoints (bis zum Reboot des Servers)
- Noch keine Unterstützung für NetworkManager, ifupdown u.A.



# Wireguard-UI

- Einfaches webbasiertes Konfigurationstool für Wireguard-Server
- Funktionen
  - Administration des Wireguard Servers
  - Verwaltung der Clients-Konfigurationen
- Geschrieben in Go
- Erster Commit: 18. April 2020
- <https://github.com/ngoduykhanh/wireguard-ui>

# Wireguard-UI Installation

```
docker run -d --name wireguard-ui \
  --restart unless-stopped \
  -p 80:5000 \
  -v /srv/wireguard-ui/db:/app/db \
  -v /etc/wireguard:/etc/wireguard \
  ngoduykhanh/wireguard-ui
```

# Wireguard-UI Systemd Integration

```
#/etc/systemd/system/wgui.service
```

```
[Unit]
```

```
Description=Restart WireGuard
```

```
After=network.target
```

```
[Service]
```

```
Type=oneshot
```

```
ExecStart=/usr/bin/systemctl restart wg-quick@wg0.service
```

```
# /etc/systemd/system/wgui.path
```

```
[Unit]
```

```
Description=Watch /etc/wireguard/wg0.conf for changes
```

```
[Path]
```

```
PathModified=/etc/wireguard/wg0.conf
```

```
[Install]
```

```
WantedBy=multi-user.target
```

# Tailscale

- Service zur Verwaltung von Wireguard Clients
- Peer-to-Peer VPN, kein VPN-Server nötig
- Direkte Kommunikation (wenn möglich, ansonsten über Proxy des Herstellers)
- Linux, Windows, MacOS, iOS, Android
  
- Linux- und Android Client unter BSD Lizenz
- Open Source Version des Control-Servers angekündigt
- Alternative Control-Server Implementierung: Headscale