TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Sebastian Schrader

Flexible Skalierung und Segmentierung mittels eduVPN/OpenVPN

82. DFN-BT

Dresden, 25.03.2025

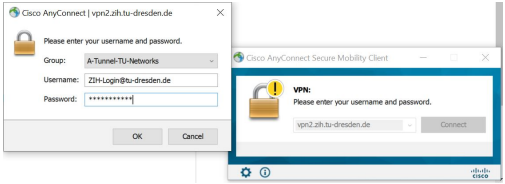
ZIH
Center for Information Services & High Performance Computing

Einleitung

Ausgangssituation VPN@TUD

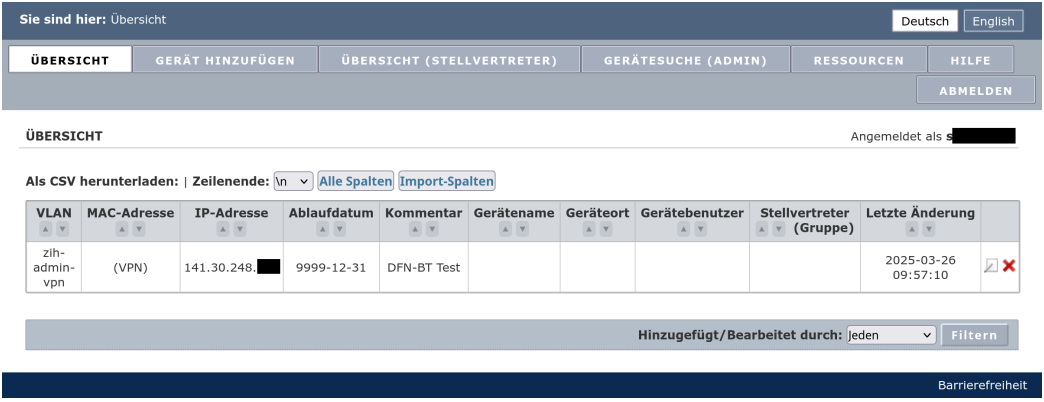
Cisco SSL VPN @ TUD

- Cisco AnyConnect (mittlerweile Secure Client) SSL VPN
- Passwort-Authentifizierung mit Hochschullogin
- Über 200 VPN-Ressourcen
  - VPN-Ressource = Name + ein oder mehrere IP-Präfixe
  - Zweck: Segmentierung[a]
  - Auswahl per Suffix am Nutzernamen (`nutzername@vpn-ressource`)
  - RADIUS-Server macht Authentifizierung, Authorisierung und IP-Addressvergabe



[a] Oder für kreative Würgarounds für bspw. Lizenzserver ohne Authentifizierung

IP-Adressverwaltung

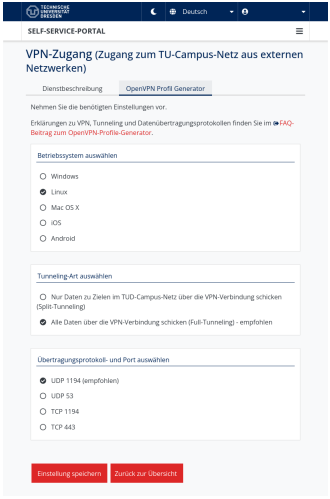


OpenVPN @ TUD

- Unmittelbar zu Beginn der Corona-Pandemie rasch aufgebaut um Kapazitäts-Engpässe im VPN zu verhindern
- Use Case: Simples VPN ohne spezielle Anforderungen/Segmentierung für die breite Masse

OpenVPN @ TUD: Profilgenerator

Profil-Erstellung über Self-Service-Portal



Multi-Faktor-Authentifizierung (MFA)

- VPN ist gängiges Angriffsziel
  - Standardisierte Schnittstelle hilft Angreifern Zugangsdaten durchzuprobieren
  - Angreifer erlangt über VPN Zugriff auf das interne Netz
  - Lateral Movement
- Hochschule beschließt flächendeckend MFA für VPN zu etablieren
- Wenn irgendwie möglich bitte als SSO per SAML
  - User Experience (UX)
  - Sicherheit
- Umfangreiche Evaluation der MFA-Optionen

Migration von Cisco AnyConnect zu SAML?

- Prinzipiell unterstützt Cisco ASA/Secure Firewall Authentifizierung von SSL VPN-Clients per SAML
- SAML-Implementierung jedoch inkompatibel mit unserem RADIUS-Ansatz
  - VPN-Ressource über RADIUS-Realm
  - Authorisierung über den RADIUS-Server
  - IP-Adressvergabe per RADIUS-Antwort-Attribut
- VPN-Ressourcen müssten grundsätzlich anders implementiert werden
- Bevor wir das tun, prüfen wir uns erstmal um, was es sonst noch möglich wäre

Anforderungen an eine VPN-Lösung

- Skalierbarkeit: Viele VPN-Ressourcen
- Breite Client-Unterstützung (mindestens Linux, macOS, Windows, Android, IOS)
- Hochverfügbarkeit: Mehr als Maschine stellt die gleiche VPN-Ressource zur Verfügung
- Einfache Migration von existierenden Lösung
- Multi-Faktor-Authentifizierung per SAML/SSO
- Von uns kontrollierte IP-Addressvergabe

Gründe für Entscheidung zur Migration zu eduVPN

- Positive Rückmeldungen zu OpenVPN
- eduVPN basiert ebenfalls auf OpenVPN (und WireGuard)
- Zahlreiche andere Einrichtungen sind ebenfalls erfolgreich migriert
- Keine Kosten Lizenzen/Hardware für OpenVPN/eduVPN
- Gute Skalierbarkeit und Erweiterbarkeit
- Authentifizierung per SAML
- Freie Software (Open Source)
- Stabiles Projekt mit GÉANT-Verankerung

Umsetzung mit eduVPN

eduVPN Profil-Modell vs. Unsere Anforderungen

eduVPN-Modell

eduVPN-Profil legt u.a. fest:

- Anzeigename
- IP-Präfixe
- DNS-Server
- Routen (Full vs Split Tunneling)

Ein eduVPN-Profil wird n Nodes mit jeweils m Ports[a] zugewiesen. Der IP-Adressraum wird dann $n \cdot m$ Teile zerlegt und auf die Nodes und Ports verteilt.

[a] Ports entfallen bei WireGuard.

Unsere Anforderungen

- Jede IP-Adresse/Profil an jeder Node/jedem OpenVPN-Prozess möglich. Hintergrund: 200 VPN-Ressourcen mit teilweise winzigen IP-Präfixen (z.B. /30)
- Beliebig viele OpenVPN-Prozesse (1 Prozess pro CPU) ohne Aufteilung von VPN-Ressourcen auf bestimmte Nodes
- Kontrollierbare IP-Adresse-Zuweisung

eduVPN Profil-Modell vs. Unsere Anforderungen

eduVPN-Modell

eduVPN-Profil legt u.a. fest:

- Anzeigename
- IP-Präfixe
- DNS-Server
- Routen (Full vs Split Tunneling)

Ein eduVPN-Profil wird n Nodes mit jeweils m Ports[a] zugewiesen. Der IP-Adressraum wird dann $n \cdot m$ Teile zerlegt und auf die Nodes und Ports verteilt.

Unsere Anforderungen

- Jede IP-Adresse/Profil an jeder Node/jedem OpenVPN-Prozess möglich. Hintergrund: 200 VPN-Ressourcen mit teilweise winzigen IP-Präfixen (z.B. /30)
- Beliebig viele OpenVPN-Prozesse (1 Prozess pro CPU) ohne Aufteilung von VPN-Ressourcen auf bestimmte Nodes
- Kontrollierbare IP-Adresse-Zuweisung

[a] Ports entfallen bei WireGuard.

Lösung 1: Kleine Präfixe

- OpenVPN simuliert unter Windows eine Ethernet-Schnittstelle (auch für reine IP-Tunnel (Tun) statt TAP)
- 3 Adressen pro Präfix nicht nutzbar
- Bei VPN-Ressourcen mit großen Präfixen kein Problem, apätestens ab IPv4-/30-Präfixen wird es jedoch irgendwann absurd

Lösung 1: Kleine Präfixe

- OpenVPN simuliert unter Windows eine Ethernet-Schnittstelle (auch für reine IP-Tunnel (Tun) statt TAP)
- 3 Adressen pro Präfix nicht nutzbar
- Bei VPN-Ressourcen mit großen Präfixen kein Problem, apätestens ab IPv4-/30-Präfixen wird es jedoch irgendwann absurd

- Lösung:
  - Präfixe der VPN-Ressourcen stammen aus größerem zusammenhängenden Adressraum
  - Dem VPN-Client gegenüber kann behauptet werden, er sei in einem größeren Präfix
  - Das übergeordnete Präfix ist ja im Hochschulnetz und soll sowieso immer über das VPN
- Beispiel:
  - VPN-Resource: 192.0.2.32/30 Network range: 192.0.2.32 - 192.0.2.35 Usable range: 192.0.2.33 - 192.0.2.34
- VPN-Client erhält 192.0.2.32/24 mit Gateway 192.0.2.1 im Tunnel

Lösungsbaustein 2: Hostrouten

- OpenVPN bietet umfangreiche Plugin/Script-Hook-Funktionalität:

  Insbesondere kann die Client-Config (wie IP-Adressen des Clients im VPN) über `client-connect`-Hook dynamisch angepasst werden

Lösungsbaustein 2: Hostrouten

- OpenVPN bietet umfangreiche Plugin/Script-Hook-Funktionalität:

  Insbesondere kann die Client-Config (wie IP-Adressen des Clients im VPN) über `client-connect`-Hook dynamisch angepasst werden

- Unser Ansatz:

  1. Adressvergabe von eduVPN komplett übergehen/ignorieren
  2. IP-Adresse über OpenVPN-Hook beim RADIUS-Server beziehen und dynamisch setzen
  3. Hostroute für Client auf der Node zum Tunnel-Interface des jeweiligen OpenVPN-Prozesses anlegen
  4. Hostrouten per BGP verbreiten
  5. Profit

Lösungsbaustein 3: OpenVPN Hooks

- OpenVPN bietet externe Skripte oder dynamisch geladene Plugins mit C-FFI-Callbacks als Hook-Schnittstelle an
- Zunächst PoC mit Python-Skripten
- State Management mit externen Skripten umständlich
- Stattdessen kleiner Daemon und Shared-Library in Rust
  - Daemon kümmert sich um priviligierte Operationen (Netlink)
  - Macht RADIUS (Authenication *und* **Accouting**)
  - Ersetzt den `vpn-daemon` von eduVPN (Reimplementierung der API)
  - Räumt automatisch auf

RADIUS

- Authentifizierung erfolgt per Zertifikat auf OpenVPN-Protokollebene
- RADIUS-Server macht daher keine Authentifizierung, sondern nur noch Authorisierung (`Service-Type = Call-Check`)
- Notwendige Informationen werden aus dem Subject-Attributen extrahiert
- Kleiner Patch des eduVPN User Portal um diese Informationen zu ergänzen

```
Version: 3 (0x2)
Serial Number:
    19:46:65:c7:0c:07:18:c7:38:78:67:26:3d:8d:7b:13:fa:6f:8c:f9
Signature Algorithm: ED25519
Issuer: CN=vpn-test.tu-dresden.de ovpn ca 20240524
Validity
    Not Before: Mar 26 08:43:46 2025 GMT
    Not After : Mar 26 23:48:46 2025 GMT
Subject:
  ST=vpn-test-allgemein,
  L=full,
  OU=vpn-test-allgemein-full,
  CN=90N4nRIUm8irQCsisUOzxjUzAXvvEt8N8sJ7HqDXrz4=,
  pseudonym=ninu222n,
  role=authorize-
Subject Public Key Info:
    Public Key Algorithm: ED25519
        ED25519 Public-Key:
        pub:
            1d:7e:8f:75:de:4d:bb:79:33:dc:07:54:8a:b3:bd:
            21:ae:8c:c8:3b:1a:fd:0d:84:e4:e5:24:42:5c:a5:
            83:4a
```

Lösungsbaustein 4: ipvs-Load Balancer pro Node

Problem

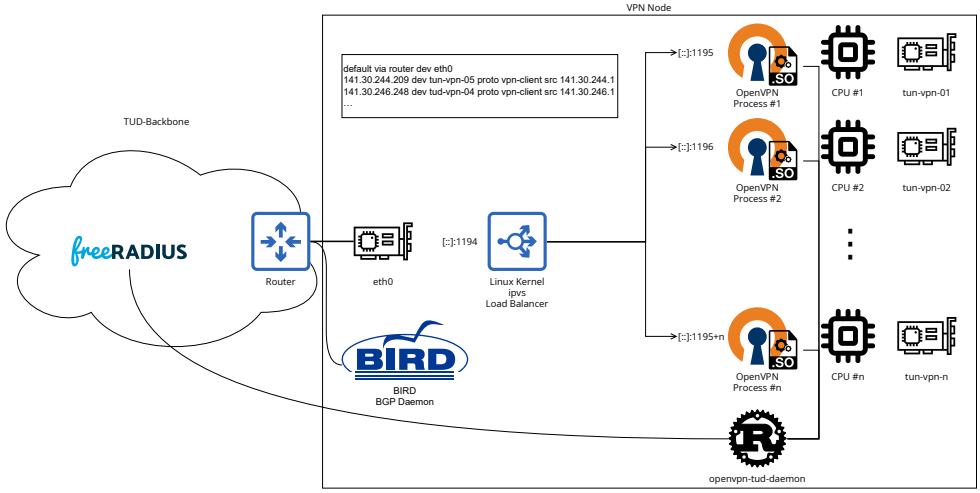
- OpenVPN ist single-threaded[a], Parallelität erfordern separate OpenVPN-Prozesse
- Gezielte Verteiltung bestimmter VPN-Ressource auf bestimmte OpenVPN-Prozesse schwierig.
  - Wie viel Performance braucht eine VPN-Ressource?
  -
- OpenVPN-Verbindungen

Unsere Lösung

- Linux-Kernel bietet einen Load Balancer namens LVS/`ipvs` mit
- Steuerung und Health Checks im User Space mittels `keepalived`
- Mehrere Frontend-Ports trivial realisiebar
  - UDP 1194
  - UDP 53
  - TCP 443
  - TCP 1194

[a] Analog zu WireGuard gibt es mittlerweile auch Kernel-Implementierungen des OpenVPN Datenkanals (Data Channel Offload/DCO)

Stufe 1: Mehrere Prozesse



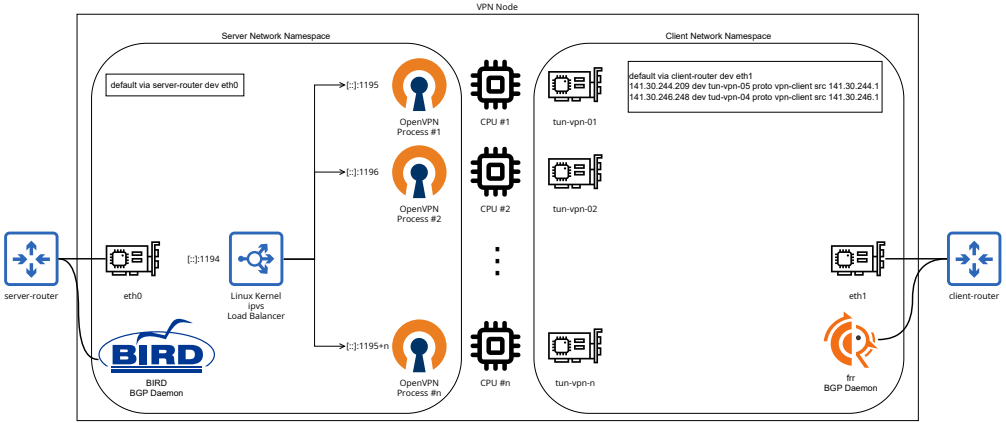
Stufe 1: Mehrere Prozesse

(`openvpn-tud-daemon`, Plugin, RADIUS-Server werden zur Übersichtlichkeit in den folgenden Versionen weggelassen, sind aber weiterhin vorhanden)

Stufe 2: Namespaces zur Trennung von Clients und Server

- Um Client-Traffic und Server-Traffic besser voneinander isolieren zu können, bieten sich Network Namespaces an
  - Komplett separate nftables-Regelwerk
  - Separate BGP-Peerings und Policy
- Funktionsweise:
  1. OpenVPN-Prozess startet im Server Namespace
  2. OpenVPN-Prozess öffnet und initialisiert Tunnel-Interface
  3. Hook verschiebt initialisiertes Interface in Client-Namespace

Stufe 2: Namespaces zur Trennung von Clients und Server

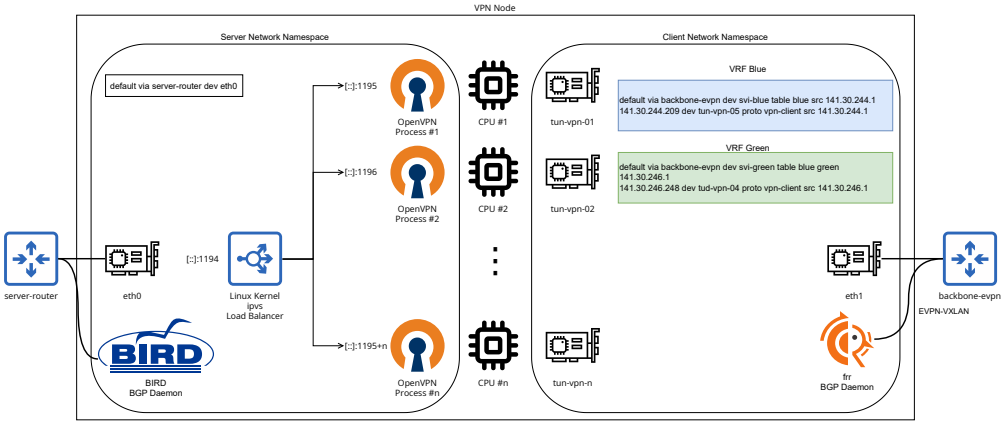


Stufe 3: VRFs zur Segmentierung der Clients untereinander

- nftables-Regelwerk auf VPN-Node liefert Basisschutz, erlaubt aber keine Regeln durch die Verantwortlichen der VPN-Ressourcen
- VPN-Ressourcen sollen durch die gleiche Firewall wie die andere IP-Präfixe (z.B. LAN oder DC) geschützt werden können
- Segmentierung der VPN-Ressourcen untereinander mittels getrennter Routing-Tabellen (VRFs[1])
- BGP-EVPN mit VXLAN um beliebige VRFs transparent zwischen Backbone und VPN-Nodes auszutauschen

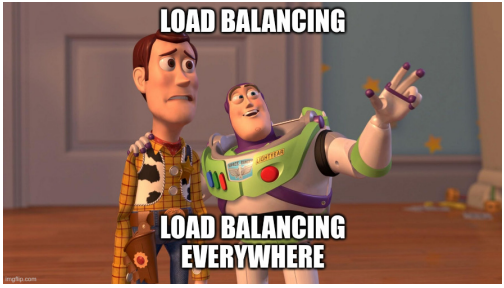
[1] VRFs sind für IP-Präfixe sind vergleichbar mit VLANs für MAC-Adressen

Stufe 3: VRFs zur Segmentierung der Clients untereinander



Load Balancing/Steering

1. eduVPN User Portal wählt einen (logischen) Nodenamen
2. Logischer Nodename kann in mehrere logische IP-Adressen aufgelöst werden und OpenVPN-Client wählt bzw. probiert durch
3. logische VPN-Server-IP-Adresse kann per Anycast von mehreren physischen VPN-Nodes angeboten werden
4. Physische VPN-Node betreibt mehrere OpenVPN-Prozesse mit ipvs Load Balancer

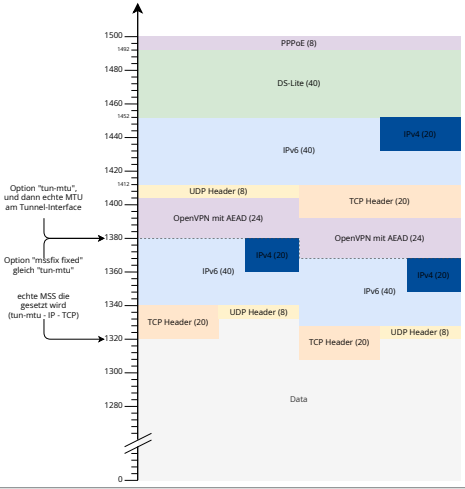


Anmerkungen zu MTU

- (VPN-)Tunnel verringern die MTU
- Fragmentierung unterwegs nur bei IPv4 möglich
- Path MTU Discovery funtioniert zwar oft, leider aber nicht immer
- Beispiel aus der Praxis wo es nicht funktioniert: WLAN im Zug
- Bei WireGuard muss man eine MTU statisch festlegen
- OpenVPN könnte theoretisch MTU dynamisch ermitteln, in der Praxis
- Auswege:
  - Zusätzliche Profile für reduzierte MTU: User Experience
  - Generell geringe MTU: Man verschenkt etwas an maximaler Bandbreite
- Unsere Lösung:
  - MTU 1380 bei UDP-Profilen[2]
  - TCP

[2] eduVPN-Projekt empfiehlt bei WireGuard 1392

Anmerkungen zu MTU



Realisierung

- Upstream-Debian-Pakete mit unseren eigenen Anpassungen neu gebaut
- eduVPN-eigene Skripte und Mechnasimen zur Installation werden *nicht* verwendet
- OpenVPN-Server-Config wird von uns generiert
- Eigene Ansible-Playbooks

Rollout

Ablauf einer Migration

Für eine VPN-Ressource läuft eine Migration wie folgt ab:

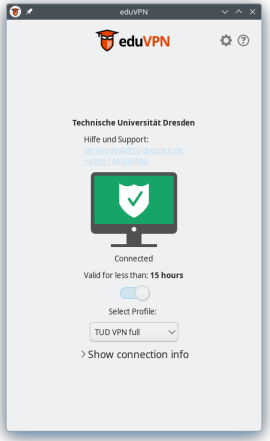
1. Start: VPN-Ressource ist nur per Cisco SSL-VPN verfügbar
2. Parallelbetrieb:
   - VPN-Ressource wird in eduVPN aktiviert
   - Gleichzeitige Nutzung derselben VPN-Ressource per Cisco Secure Client/OpenConnect oder eduVPN möglich
   - Kein Flag-Day(!) an dem alle umstellen müssen
3. Ende: VPN-Ressource wird für Cisco SSL-VPN deaktiviert

Rollout-Strategie

- Zentrale VPN-Ressourcen zuerst
  - VPN-Ressourcen des Rechenzentrums (Friendly User Test/Acceptance Test/Dogfooding)
  - TUD-Admin-VPN: Damit alle IT-Verantwortlichen den neuen Client sinnvoll ausprobieren können
  - Allgemeine VPN-Ressourcen (Allgemeines VPN für alle Angehörige): Damit jede(r), die/der kann/will, den neuen Client sofort benutzen kann
- Seit 12/24: Erste Tranche im Parallelbetrieb
- Ziel: Umstellung aller VPN-Ressourcen bis Ende 2025

Bisherige Resonanz

- User Experience mit eduVPN-Client sehr gut
- Sehr wenig Nachfragen/Probleme
- »Ich muss gar kein Passwort mehr eingeben?«



Work in Progress

WireGuard

- eduVPN-Projekt möchte OpenVPN perspektivisch los werden
- ipvs Load Balancer bei WireGuard nicht notwendig, WireGuard Load Balancing macht der Kernel intern bereits (so gut es geht)
- VRFs und Network Namespaces sollten mit WireGuard unverändert funktionieren
- Dynamische IP-Addressvergabe muss im eduVPN-User-Portal implementiert werden

IPFIX-Flow-Export

- Linux Kernel kennt sowieso alle Verbindungen (conntrack) und kann Events an User Space über neue/gelöschte Verbindungen senden
- ulogd2 (Teil des netfilter-Projekts) kann diese Events auch verarbeiten
- Existierendes ipfix-Output-Plugin leider fundamental kaputt
- Wir reparieren das

SAML-Authentifizierung mit Standard-OpenVPN-Clients

- OpenVPN selbst beherrscht neben Zertifikaten und Nutzername/Passwort auch flexiblere Authentifizierungsmethoden[3][4] (Stichworte: `IV_SSO`, `crtext`, `webauth`)
- `IV_SSO=webauth` öffnet auf (unterstützten) Clients einen Webbrowser mit einer vom OpenVPN-Server gewählten URL. Der Webserver muss dem OpenVPN-Server dann out-of-band Erfolg/Misserfolg signalisieren.
- Idee: Langlebige Zertifikate, welche zusätzliche Authentifizierung per SAML erfordern
- Use Case: Standard-OpenVPN-Clients oder Fallback falls eduVPN nur noch WireGuard unterstützt und wir aus irgendeinem Grund nicht auf WireGuard wechseln können

[3] https://github.com/OpenVPN/openvpn/blob/master/doc/management-notes.txt
[4] https://github.com/OpenVPN/openvpn3/blob/master/doc/webauth.md

Windows SBL/PLAT

- Der Standard-OpenVPN-Client für Windows »OpenVPN GUI« unterstützt *Start Before Logon (SBL)* / *Pre-Logon Access Provider (PLAP)* zur Anmeldung am VPN vor der eigentlichen Windows-Anmeldung
- Jedoch keine Web-Authentifizierung (und damit SAML) bei SBL möglich
- Wir prüfen tatsächlichen Bedarf und erwägen dann nächste Schritte

Q&A