Wireguard Proxy

Einrichtung

Beschreibung:

Wireguardproxy, dient dazu einen wirguardclient über http zu tunnel.
Das bedueted auf dem Client Rechner läuft ein bridge Client (Der braucht keine Admin rechte) der den wireguard traffic in den http tunnel jagt. und der Server nimm den traffic an und leitet ihn an den wireguard Server weiter.
Der eigetliche wireguard client hat dann als IP Adresse nicht mehr den wireguard Server sondern 127.0.0.1 nämlich den Client rechner weil auf dem gleichen port der http proxy client lauscht

Verbindungsschema

```
Echte Wireguard Client (127.0.0.1:51820) -> HTTP PROXY_CLIENT_LOKAL -> Internet -> HTTP_PROXY_SERVER -> echter wireguard Server (127.0.0.1:51820)
```

Installtion Docker container:

Dieser braucht keinerlei Konfiguration.
Einfach auf dem wirgeuard Server starten fertig.

Download docker-compose Datei : docker-compose.yml

und Dockerfile : Dockerfile

Docker compose

```
services:
  wstunnel:
    build: .
    image: wstunnel-custom
    container_name: wstunnel
    restart: unless-stopped
    network_mode: host
    # Laeuft als root -> darf Port 443 binden
    # Kein Gefummel mit Entrypoint oder Capabilities noetig
```

```
    # network_mode: host wird benötigt damit wstunnel
    # den WireGuard Port auf localhost erreichen kann (127.0.0.1:51820)
```

Dockerfile:

```
FROM debian:bookworm-slim

ARG TARGETARCH
ENV DEBIAN_FRONTEND=noninteractive

# curl fuer den Download, ca-certificates fuer HTTPS
RUN apt-get update && \
    apt-get install -y --no-install-recommends curl ca-certificates tar && \
    rm -rf /var/lib/apt/lists/*

# Aktuelle Version von GitHub API ermitteln und Binary herunterladen
RUN set -eux; \
    VERSION=$(curl -fsSL https://api.github.com/repos/erebe/wstunnel/releases/latest \
        | grep '"tag_name"' | sed 's/.*"tag_name": *"\([^"]*\)".*/\1/'); \
    VERNUM="${VERSION#v}"; \
    case "$(uname -m)" in \
        x86_64)  ARCH="amd64" ;; \
        aarch64) ARCH="arm64" ;; \
        armv7*)  ARCH="armv7" ;; \
        *)       echo "Unsupported arch: $(uname -m)"; exit 1 ;; \
    esac; \
    echo "Downloading wstunnel ${VERSION} for ${ARCH}..."; \
    curl -fsSL
"https://github.com/erebe/wstunnel/releases/download/${VERSION}/wstunnel_${VERNUM}_linux_${ARCH}.tar.g
z" \
        | tar -xz -C /usr/local/bin wstunnel; \
    chmod +x /usr/local/bin/wstunnel; \
    wstunnel --version

EXPOSE 443

ENTRYPOINT ["/usr/local/bin/wstunnel"]
CMD ["server", "--restrict-to", "127.0.0.1:51820", "wss://0.0.0.0:443"]
```

Proxy Client installtion:

Windows:

Hier den Servernamen und die IPs anpassen

Download bat : start_wstunnel.bat

```
@echo off
setlocal enabledelayedexpansion
:: ============================================================
::  wstunnel Client - WireGuard over WebSocket
::  Anpassen: PROXY_SERVER = Domainname eures wstunnel-Servers
:: ============================================================

set PROXY_SERVER=vpn.example.com

:: Lokaler UDP Port fuer WireGuard (muss mit WG Endpoint uebereinstimmen)
set LOCAL_WG_PORT=51820

:: Remote WireGuard Port auf dem Server
set REMOTE_WG_PORT=51820

:: wstunnel Binary
set WSTUNNEL=%~dp0wstunnel.exe

:: GitHub Download URL (latest release, Windows x64)
set DOWNLOAD_URL=https://github.com/erebe/wstunnel/releases/latest/download/wstunnel_x86_64-pc-windows-msvc.exe

:: ============================================================

echo.
echo  ##############################################
echo   wstunnel WireGuard Proxy
echo  ##############################################
echo.
echo  Server : %PROXY_SERVER%
echo  Tunnel : UDP localhost:%LOCAL_WG_PORT% --^> %PROXY_SERVER%:%REMOTE_WG_PORT%
echo.
```

```
:: --- Pruefen ob wstunnel.exe vorhanden ---
if exist "%WSTUNNEL%" goto :start

echo  [!] wstunnel.exe nicht gefunden.
echo  [*] Versuche automatischen Download via wget...
echo.

:: wget verfuegbar?
where wget >nul 2>&1
if %errorlevel% neq 0 (
    echo  [!] wget nicht gefunden. Versuche PowerShell...
    goto :download_ps
)

wget --no-check-certificate -q --show-progress -O "%WSTUNNEL%" "%DOWNLOAD_URL%"
if %errorlevel% neq 0 (
    echo  [!] wget Download fehlgeschlagen. Versuche PowerShell...
    goto :download_ps
)
goto :download_ok

:download_ps
echo  [*] Lade via PowerShell (Invoke-WebRequest)...
powershell -NoProfile -Command ^
  "Invoke-WebRequest -Uri '%DOWNLOAD_URL%' -OutFile '%WSTUNNEL%' -UseBasicParsing"
if %errorlevel% neq 0 (
    echo.
    echo  [FEHLER] Download fehlgeschlagen!
    echo  Bitte manuell herunterladen:
    echo  %DOWNLOAD_URL%
    echo  und als wstunnel.exe in diesen Ordner legen.
    echo.
    pause
    exit /b 1
)

:download_ok
if not exist "%WSTUNNEL%" (
    echo  [FEHLER] wstunnel.exe nach Download nicht gefunden!
```

```
    pause
    exit /b 1
)
echo  [OK] wstunnel.exe erfolgreich heruntergeladen.
echo.

:start
echo  Bitte WireGuard Endpoint auf 127.0.0.1:%LOCAL_WG_PORT% setzen!
echo  Fenster offen lassen solange VPN aktiv ist.
echo.
pause

"%WSTUNNEL%" client ^
  -L "udp://%LOCAL_WG_PORT%:127.0.0.1:%REMOTE_WG_PORT%?timeout_sec=0" ^
  "wss://%PROXY_SERVER%:443"

echo.
echo  wstunnel beendet.
pause
```

Linux:

Hier den Servernamen und die IPs anpassen

Download bat : start_wstunnel.sh

```
#!/bin/bash
# ============================================================
#  wstunnel Client - WireGuard over WebSocket
#  Anpassen: PROXY_SERVER = Domainname eures wstunnel-Servers
# ============================================================

PROXY_SERVER="vpn.example.com"

# Lokaler UDP Port fuer WireGuard (muss mit WG Endpoint uebereinstimmen)
LOCAL_WG_PORT=51820

# Remote WireGuard Port auf dem Server
REMOTE_WG_PORT=51820

# wstunnel Binary (im selben Ordner oder im PATH)
```

```
WSTUNNEL="wstunnel"

# ============================================================

# Farben
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m'

echo ""
echo -e "${GREEN} *** wstunnel WireGuard Proxy ***${NC}"
echo -e " Server : ${YELLOW}${PROXY_SERVER}${NC}"
echo -e " Tunnel : UDP localhost:${LOCAL_WG_PORT} --> ${PROXY_SERVER}:${REMOTE_WG_PORT}"
echo ""
echo -e "${YELLOW} Bitte WireGuard Endpoint auf 127.0.0.1:${LOCAL_WG_PORT} setzen!${NC}"
echo " Skript laufen lassen solange VPN aktiv ist."
echo ""

# Pruefen ob wstunnel vorhanden
if ! command -v "$WSTUNNEL" &>/dev/null && [ ! -f "./$WSTUNNEL" ]; then
    echo -e "${RED} FEHLER: wstunnel nicht gefunden!${NC}"
    echo " Download: https://github.com/erebe/wstunnel/releases"
    echo " Binary nach /usr/local/bin/wstunnel kopieren oder in diesen Ordner legen."
    exit 1
fi

# Lokales wstunnel bevorzugen falls vorhanden
if [ -f "./wstunnel" ]; then
    WSTUNNEL="./wstunnel"
    chmod +x "$WSTUNNEL"
fi

echo -e "${GREEN} Starte Tunnel...${NC}"
echo ""

# Sauber beenden bei Strg+C
trap 'echo -e "\n${YELLOW} Tunnel beendet.${NC}"; exit 0' SIGINT SIGTERM
```

```
"$WSTUNNEL" client \
  -L "udp://${LOCAL_WG_PORT}:127.0.0.1:${REMOTE_WG_PORT}?timeout_sec=0" \
  "wss://${PROXY_SERVER}:443"

echo ""
echo -e "${YELLOW} wstunnel beendet.${NC}"
```

Der echte Wireguard Client:

Jetzt muss nur nur noch im Echten wireguard client der endpunkt auf 127.0.0.1 geändert werden

```
Endpoint = 127.0.0.1:51820
```