67. UKW-Tagung 2022

Lasst uns den Nebel lichten !

Eine Einführung in das Mesh-VPN NEBULA

Michael Dörr
techno.turtle@gmx.net

Das Mesh-VPN-Netzwerk NEBULA wurde Ende 2019 als freie Software (ohne großes Marketing) bei *Github* veröffentlicht. Bis heute findet man im Internet nur sehr wenige Anleitungen, die ganz konkret auf die Anwendung und Konfiguration von NEBULA eingehen. Dies soll mit diesem Vortrag etwas einfacher werden.

Inhaltsverzeichnis

---

(1) Einleitung

Ursprung des Mesh-VPN NEBULA

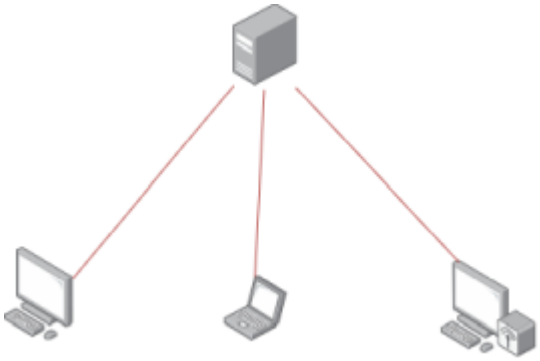
- NEBULA wurde ursprünglich von der Firma SLACK entwickelt, um die weltweiten Standorte einfach und sicher zu verbinden.
    - Introducing Nebula, the open source global overlay network from Slack
- Die Veröffentlichung als Open-Source bei *Github* erfolgte im Spätjahr 2019.
    - Dort werden mehrmals im Jahr neue Releases zum Download bereitgestellt:
    - https://github.com/slackhq/nebula
- Die ehemaligen Entwickler haben Anfang 2020 SLACK verlassen und eine eigene Firma gegründet: “Defined Networking”.
    - https://www.defined.net/nebula/
    - https://github.com/DefinedNet/nebula-docs

Im Gegensatz zum *Wireguard*-Protokoll ist *NEBULA* in der deutschen Medienlandschaft (z.B. bei Heise oder in diversen Linux-Heften) bisher noch nicht präsentiert worden.

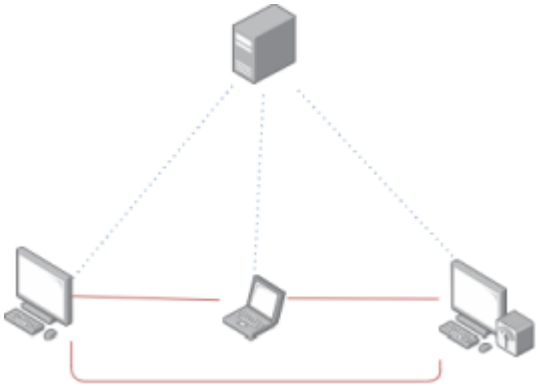
Architektur von NEBULA

Im Gegensatz zu einem klassischen Client- und Server-Netzwerk, werden bei einem vermaschten Netzwerk alle Anwendungsdaten direkt zwischen den Kommunikationspartnern (End-to-End) ausgetauscht. Dabei werden spezielle Techniken genutzt (ähnlich zu TURN im VoIP-Bereich), um Firewalls und NAT-Gateways zu durchtunneln.

- Bei einem **Hub-n-Spoke** Netzwerk werden alle Pakete über einen zentralen Server gesendet.



    - Dies hat höhere Latenzzeiten und niedrigeren Durchsatz zur Folge.
- Bei einem **Mesh** Netzwerk werden, wenn möglich, die Datenpakete auf direktem Weg zwischen den Endgeräten gesendet.



    - Im ersten Schritt erfolgt eine Discovery-Phase über einen zentralen Server (sog. *Leuchtturm*), der alle Clients und ihre Adressen lernt.
- Ein Nebula-VPN basiert auf den Protokollen UDP und IPv4. Als äussere Transportschicht kann aber auch IPv6 verwendet werden.
- Der Treiber verwendet diverse Techniken (NAT hole punching, Relaying) um Konnektivität auch unter schwierigen Bedingungen, z.B. hinter Firewalls oder mehrfachen NAT-Umgebungen, herstellen zu können.

Funktionsweise von NEBULA

- Es gibt 3 Arten von Instanzen, die für ein Nebula-Netzwerk wichtig sind:

    1. Eine Certificate Authority (**CA**-Host) zum Erstellen und Erneuern von Zertifikaten.

2. Discovery-Nodes (sog. **Lighthouses**) mit einer festen, öffentlichen IP- oder DNS-Adresse, die über alle aktiven Endgeräte Buch führen.
3. Alle Endgeräte (**Nodes**) benötigen einen konfigurierten Nebula-Treiber.

- *Alle* Nebula-Hosts brauchen, ausser dem *Nebula-Treiber*, *zwei Zertifikate* und *eine YAML*-Konfigurationsdatei.
- Die Zertifikate bescheinigen die Identität von Nebula-Hosts und die eventuelle Zugehörigkeit zu sogenannten Security-Groups.
- Der Nebula-Treiber kann mit Hilfe dieser Gruppen Firewall-Regeln anwenden und Traffic filtern.
- Der Nebula-Netzwerktreiber kann bei *Github* für diverse Betriebssysteme und CPUs heruntergeladen werden: Linux, FreeBSD, Windows, MacOS, Android, iOS
    - https://github.com/slackhq/nebula/releases/tag/v1.6.0

---

(2) **NEBULA** Workshop

Workshop Teil #1

Zertifikate für alle Hosts erstellen

1. Zuerst muss auf dem **CA**-Host ein Zertifikat für die Certificate Authority erstellt werden:

```
nebula-cert ca -name "Nebula Workshop" -duration "10000h"
chmod 400 ca.key; chmod 444 ca.crt
mkdir -p CA; chmod 700 CA; mv ca.key ca.crt CA
```

- Erzeugt die Dateien `ca.key` und `ca.crt` im Directory `./CA`
- **ACHTUNG**: die Datei `ca.key` ist der Schlüssel zur Sicherheit eines Nebula-VPNs und muss daher entsprechend geschützt werden!

1. Mit diesen beiden Dateien kann ein Host-Zertifikat für's **Lighthouse** ausgestellt werden:

```
nebula-cert sign -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" \
-name "lighthouse" -ip "192.168.100.1/24"
mkdir -p lighthouse; cp CA/ca.crt lighthouse/ca.crt
cp lighthouse.key lighthouse/host.key; cp lighthouse.crt lighthouse/host.crt
```

- Dies erstellt die Dateien `lighthouse.key` und `lighthouse.crt`, die später im `pki` Abschnitt in der Datei `config.yaml` auf dem Host `lighthouse` als `host.key` und `host.crt` verwendet werden.
- Auch die Datei `ca.crt` wird im `pki` Abschnitt jeder `config.yaml` Datei benötigt!

3. Danach können Host-Zertifikate für die anderen Nebula-**Nodes** erstellt werden. Die 3 Nebula-Clients `laptop, desktop` und `server` sollen hier als Beispiel dienen.

- Für Host `laptop`:

```
nebula-cert sign -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" \
-name "laptop" -ip "192.168.100.10/24" -groups "workstations,ssh"
mkdir -p laptop; cp CA/ca.crt laptop/ca.crt
cp laptop.key laptop/host.key; cp laptop.crt laptop/host.crt
```

- Für Host `desktop`:

```
nebula-cert sign -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" \
-name "desktop" -ip "192.168.100.50/24" -groups "workstations,ssh"
mkdir -p desktop; cp CA/ca.crt desktop/ca.crt
cp desktop.key desktop/host.key; cp desktop.crt desktop/host.crt
```

- Für Host `server` :

```
nebula-cert sign -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" \
-name "server" -ip "192.168.100.90/24" -groups "servers,admin"
mkdir -p server; cp CA/ca.crt server/ca.crt
cp server.key server/host.key; cp server.crt server/host.crt
```

- Wie man an den ähnlichen Aufrufsequenzen sehen kann, bieten sich Automationslösungen (siehe Links) zum massenhaften Generieren der Zertifikate an.

Workshop Teil #2

Datei `/etc/nebula/config.yaml` erstellen und anpassen

- Auf Linux-basierten Nebula-Hosts wird die Datei `config.yaml`, die zur Konfiguration des Nebula-Netzwerktreibers benötigt wird, standardmäßig im Directory `/etc/nebula` abgelegt.
- Eine Vorlage kann von *Github* heruntergeladen werden:

```
curl -o config.yml
https://raw.githubusercontent.com/slackhq/nebula/master/examples/config.yml
cp config.yml config-lh.yml
cp config.yml config-nd.yml
```

- Die Konfigurationsdatei `config.yaml` ist in zwölf Abschnitte gegliedert:
  `pki, listen, tun, punchy, static_host_map, lighthouse, cipher, local_range, relay, sshd, logging, firewall`
- Der Abschnitt `pki` sieht folgendermaßen aus:

```
pki:
  ca:   /etc/nebula/ca.crt
  cert: /etc/nebula/host.crt
  key:  /etc/nebula/host.key
```

Workshop Teil #3

Die Inhalte der drei Abschnitte `static_host_map, lighthouse` und `listen` machen den Unterschied zwischen Discovery-Nodes (*Leuchttürme*) und alle anderen, "normalen" Nebula-Hosts aus.

- Einstellungen für `config-lh.yaml` für ein *Lighthouse*

```
# The static host map defines a set of hosts with fixed IP addresses on the
internet (or any network).
# IMPORTANT: THIS SHOULD BE EMPTY ON LIGHTHOUSE NODES
static_host_map:

lighthouse:
  # am_lighthouse is used to enable lighthouse functionality for a node.
  # This should ONLY be true on nodes you have configured to be lighthouses
  am_lighthouse: true
  # hosts is a list of lighthouse hosts this node should report to and query from
  # IMPORTANT: THIS SHOULD BE EMPTY ON LIGHTHOUSE NODES
  #hosts:

# Port Nebula will be listening on. The default here is 4242.
# For a lighthouse node, the port should be defined.
# Using port 0 will dynamically assign a port and is recommended for roaming
nodes.
listen:
  # To listen on both any ipv4 and ipv6 use "[::]"
  host: "[::]"
  port: 4242
```

- Beispiel für `config-nd.yaml` für alle anderen Hosts

```
# The static host map defines a set of hosts with fixed IP addresses on the
internet (or any network).
# The syntax is:
```

```
#   "{nebula ip}": ["{routable ip/dns name}:{routable port}"]
static_host_map:
  "192.168.100.1": ["100.64.22.11:4242"]

lighthouse:
  # am_lighthouse is used to enable lighthouse functionality for a node.
  # This should ONLY be true on nodes you have configured to be lighthouses
  am_lighthouse: false
  # hosts is a list of lighthouse hosts this node should report to and query from
  # IMPORTANT: THIS SHOULD BE LIGHTHOUSES' NEBULA IPs! NOT REAL ROUTABLE IPs!
  hosts:
    - "192.168.100.1"
  # interval is the number of seconds between updates from this node to a
lighthouse.
  interval: 60

# Using port 0 will dynamically assign a port and is recommended for roaming
nodes.
listen:
  host: 0.0.0.0
  port: 0
```

Workshop Teil #4

Die weiteren Abschnitte in `config.yaml` sind für Lighthouse- und Standard-Nodes meist identisch.

- Der Abschnitt `punchy` wird benutzt, um "NAT hole punching" einzuschalten und zu konfigurieren.
    - UDP hole punching - Wikipedia

```
punchy:
  punch: true
  respond: true
  delay: 1s
```

- Im Abschnitt `tun` wird das Netzwerk-Interface für Nebula konfiguriert.

```
tun:
  disabled: false
  dev: nebula1
  drop_local_broadcast: false
  drop_multicast: false
  tx_queue: 500
  mtu: 1300
```

Workshop Teil #5

Je nach Anwendungsfall, können in der `config.yaml` Datei für jeden Host individuelle Firewall-Regeln konfiguriert werden.

- Im Abschnitt `firewall` werden die Security-Gruppen (siehe Teil #1) in Firewall-Regeln angewendet.

```
# Nebula security group config
firewall:
  conntrack:
    tcp_timeout: 12m
    udp_timeout: 3m
    default_timeout: 10m
    max_connections: 100000

  outbound:
    # Allow all outbound traffic from this node
    - port: any
      proto: any
      host: any

  inbound:
    # Allow icmp between any nebula hosts
    - port: any
      proto: icmp
      host: any
    # Allow tcp/443 from any host within the VPN
    - port: 443
      proto: tcp
      host: any
    # Allows tcp/22 from any host with group ssh
    - port: 22
      proto: tcp
      groups:
        - ssh
```

Workshop Teil #6

Starten und Stoppen von NEBULA unter Linux

- Folgende Aktionen müssen auf *allen* beteiligten Nodes des Nebula-Mesh ausgeführt werden!
  - Kopiere `ca.crt, host.crt, host.key` und `config.yaml` nach `/etc/nebula/`
  - Kopiere Binaries `nebula` und `nebula-cert` nach `/usr/local/bin/`
  - Herunterladen der “systemd service” Datei und Starten des Nebula-Service

```
curl -o /etc/system/system/nebula.service \
https://github.com/slackhq/nebula/raw/master/examples/service_scripts/nebula.servi
ce
```

```
systemctl start nebula.service && systemctl enable nebula.service
```

- Weitere Tipps unter https://www.defined.net/nebula/quick-start/

Workshop Teil #7

Tipps zur Fehlersuche und Entwanzung

- IP Addressen anzeigen und *Lighthouse* anpingen

```
ip addr show
ping 192.168.100.1
```

- Nebula-Zertifikate anzeigen

```
nebula-cert print -path /etc/nebula/ca.crt
nebula-cert print -path /etc/nebula/host.crt
```

- Status des Nebula-Daemons anzeigen

```
systemctl status nebula.service
```

- Fehlerprotokoll im Abschnitt `logging` der `config.yaml` Datei konfigurieren.

---

(3) **NEBULA** Demo

- Letztes Jahr (2021) habe ich beim Linuxtag der AG Microcomputer die Site-to-Site Kopplung mittels *OPNsense*, *WireGuard* und *IPv6* gezeigt.
    - Das funktioniert prima und stabil: `LU <-> DÜW`
- Dieses Jahr (2022) geht es um den Remote-Zugang mittels *Nebula*, z.B. für Wartungszwecke oder zum Monitoring von Diensten.
    - **DEMO**: `Externes Netz -> DÜW` und `Externes Netz -> LU`

Fragerunde:

Was wollt ihr noch wissen?

- Welche Mesh-VPNs gibt es als Alternative zu *Nebula*?
    - *ZeroTier One* (auch von *OPNsense* unterstützt)
    - *TailScale* (nutzt *WireGuard* als Protokoll)
    - *HeadScale* (Open Source Alternative zu *TailScale*)
    - *NetMaker* (nutzt *WireGuard* als Protokoll; hat *Kubernetes* als Zielgruppe)
- Welche Rolle spielt *Nebula* im Amateurfunk-Bereich?
    - Amateur Radio Emergency Data Network
        - https://www.arednmesh.org/content/nebula-mesh-vpn-tunneling
        - https://www.arednmesh.org/content/nebula-mesh-vpn
- Welche Anwendungen, ausser SLACK, benutzen *Nebula*?
    - Das Jitsi-Meet des *Freifunk München* ist seit Beginn der Corona-Zeit (Anfang 2020) aus den Medien bekannt.
      Die Nutzung von *Nebula* bei **#FFMEET** wurde auf Twitter und Slideshare eher beiläufig erwähnt.
    - Das Mesh-VPN wird für die interne Kopplung der Video-Bridges genutzt, die sich an unterschiedlichen Standorten befinden.
      Nebula wird benutzt, um ein verschlüsseltes Overlay-Netzwerk mit Point-to-Point Verbindungen zu realisieren.
    - Die Konfiguration ist im SALT-Automationsstack der Münchner "versteckt" (siehe Links).

---

(4) Links und Referenzen

Weiterführende Infos zu NEBULA

- Defined Network
    - Erläuterungen zu den Optionen und Einstellungen der `config.yaml` Datei
    - Configuration Reference - Nebula Project

    - Extend network access beyond overlay hosts
- INNOQ
    - Sicher vernetzte Remote-Arbeit
- Ars Technica
    - Nebula VPN routes between hosts privately, flexibly, and efficiently
    - How to set up your own Nebula mesh VPN, step by step
- Lawrence Systems
    - Open Source Mesh VPN Solutions
    - Nebula, the open source global overlay network VPN solution
- Jon The Nice Guy
    - Looking at the Nebula Overlay Meshed VPN Network from Slack
    - Nebula Offline Certificate Management with a Raspberry Pi using Bash
- The Orange One
    - Nebula mesh network - an introduction
    - Unsafe routes with Nebula
    - Exposing your Homelab
- LinuxBlog.xyz
    - VPN mesh with Nebula
- Unreality.xyz
    - Nebula Mesh with SSO
- ServiceMax
    - Slack Nebula Secure Mesh Network Install
    - Slack Nebula Secure Mesh on Docker
    - Slack Nebula Secure Mesh on Synology
- RootIsGod
    - Mini VPN For All!
- Spikefish Solutions
    - Getting started with Nebula
    - Nebula for Remote Access
    - Writing Nebula Firewall Rules
- Haxys.net
    - Configuring Nebula
- Lucas J. Hall
    - Nebula – Getting Started
- Alex Kretzschmar
    - Punching through tricky NAT with Nebula Mesh VPN and OPNSense
- Will Richardson
    - Configure Nebula VPN on iOS/Android

NEBULA Fundstellen bei *Github*

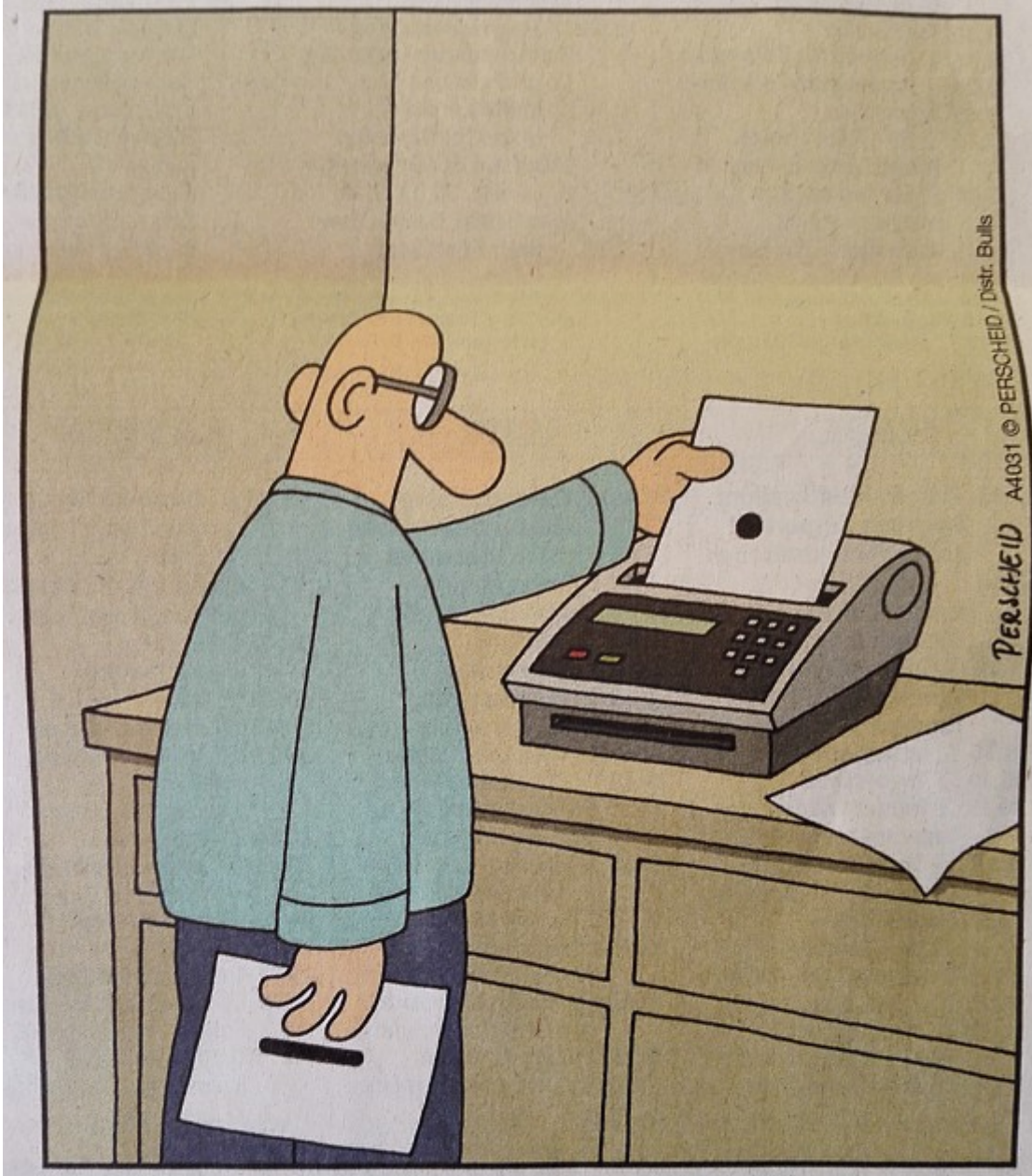
- freifunkMUC/ffmuc-salt-public/nebula
- JonTheNiceGuy/install_nebula: An Ansible Role for deploying Nebula
- JonTheNiceGuy/Nebula-Cert-Maker: A script to create host certs and keys

- henryzhangsta/nebula-vpn-helm: Helm chart for managing Nebula VPN
- RealOrangeOne/infrastructure: Servers, containers and stuff
- unreality/nebula-mesh-admin
- unreality/nebula-helper
- kanniget/nebulamgr: Bulk config tool for generating nebula VPN configs
- b177y/starship
- symkat/MeshMage
- XDGFX/nebula-nursery: Configure Nebula VPN with a bit of extra assistance
- XDGFX/nebula-gui: A test project for an Electron GUI for Nebula VPN
- AndrewPaglusch/Nebula-Ansible-Role: Nebula VPN Installer With Ansible
- chrisx8/docker-nebula: Run Nebula in a Docker container
- elestio/nebula-rest-api: REST API for Nebula client management

Vielen Dank!

Diesen Vortrag kann man auch unter https://tech.dortoka.dynvpn.de/blog/ nachlesen.

Perscheids Abgründe



UM MIR NICHT UNTERSTELLEN ZU LASSEN, ICH WÜRDE MICH DEM TECHNISCHEN FORTSCHRITT VERSCHLIESSEN, HABE ICH MIR ZUM MORSEN EIN FAXGERÄT GEKAUFT.