Den Nebel lichten !

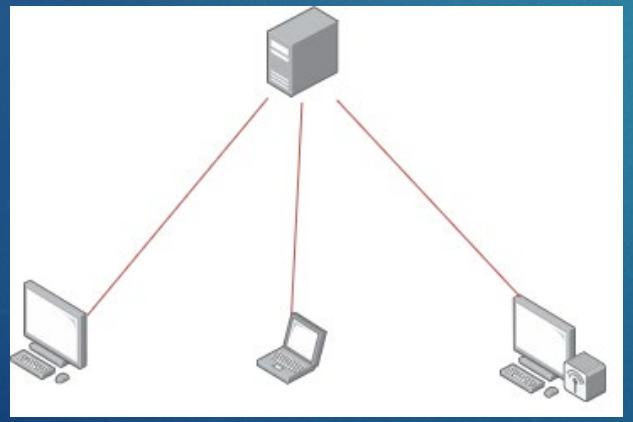
EINFÜHRUNG IN DAS MESH-VPN **NEBULA**

Inhaltsverzeichnis

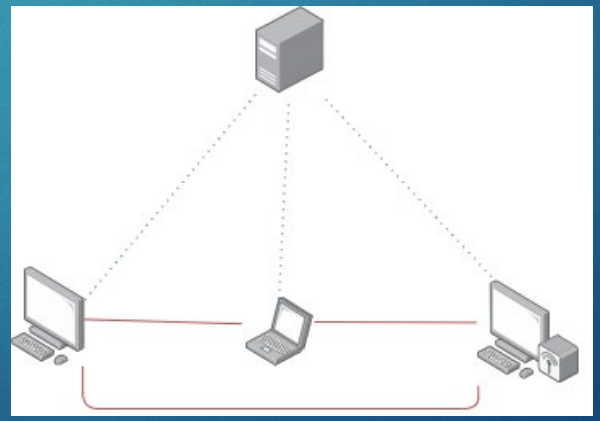
- Wie funktioniert ein Mesh-VPN?
- Nebula Workshop
- Live Demo
- Q & A
- Links & Referenzen

Architektur von **NEBULA**

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



- Bei einem Mesh Netzwerk werden die Datenpakete auf direktem Weg zwischen den Endgeräten gesendet.
- Es erfolgt vorab eine Discovery-Phase über einen zentralen Server (**Lighthouse**).



Ursprung des Mesh-VPN **NEBULA**

- Ursprünglich von SLACK Engineering entwickelt,
  zur einfachen und sicheren Vernetzung der weltweiten Standorte.
  - Introducing Nebula, the open source global overlay network from Slack

- Veröffentlichung als Open-Source bei Github im Spätjahr 2019.
  - https://github.com/slackhq/nebula

- Ehemalige Entwickler haben Anfang 2020 SLACK verlassen
  und eine eigene Firma gegründet: "Defined Networking".
  - https://www.defined.net/nebula/
  - https://github.com/DefinedNet/nebula-docs

Funktionsweise von **NEBULA**

- Es gibt 3 Hauptkomponenten:
  - Eine Certificate Authority (**CA**-Host) zum Erstellen und Erneuern von Zertifikaten.
  - Discovery-Nodes (sog. **Lighthouses**) mit einer festen, öffentlichen IP- oder DNS-Adresse, die Buch führen über alle aktiven Endgeräte.
  - Alle Endgeräte (**Nodes**) benötigen einen konfigurierten Nebula-Treiber.
- Den Nebula-Netzwerktreiber kann man bei Github für diverse Betriebs-systeme und CPUs herunterladen:
  - Linux, FreeBSD, Windows, MacOS, Android, iOS
  - https://github.com/slackhq/nebula/releases/tag/v1.5.2
- Alle Nebula-Hosts brauchen, ausser dem Nebula-Treiber, zwei Zertifikate und eine YAML-Konfigurationsdatei.
- Zertifikate bescheinigen die Identität von Nebula-Hosts und die eventuelle Zugehörigkeit zu sog. Security-Groups.
- Der Nebula-Treiber kann mit Hilfe dieser Gruppen Firewall-Regeln anwenden und Traffic filtern.
- Nebula-VPN basiert auf UDP und IPv4. Als äussere Transportschicht kann aber auch IPv6 verwendet werden.
- Nebula verwendet diverse Techniken (NAT hole punching) um Konnektivität auch unter schwierigen Bedingungen herstellen zu können.

Nebula Workshop #1a
Zertifikate erstellen

- Zuerst muss auf dem **CA**-Host ein Zertifikat für die **CA** erstellt werden
  - `./nebula-cert ca -name "Nebula Workshop" -duration "10000h"`
  - `mkdir CA; mv ca.* 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!
- Damit kann ein Host-Zertifikat für's **Lighthouse** ausgestellt werden
  - ```
    ./nebula-cert sign
    -name "lighthouse"
    -ca-crt "CA/ca.crt"
    -ca-key "CA/ca.key"
    -ip "192.168.100.1/24"
    ```
- Erstellt die Dateien `lighthouse.key` und `lighthouse.crt`, die später in der Datei `config.yaml` auf dem Host *lighthouse* als `host.key` und `host.crt` verwendet werden.
- Auch die Datei `ca.crt` wird in jeder `config.yaml` Datei benötigt!
    ```

Nebula Workshop #1b
Zertifikate erstellen

- Danach können Host-Zertifikate für die anderen Nebula-**Nodes** erstellt werden
  - `./nebula-cert sign -name "laptop" -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" -ip "192.168.100.10/24" -groups "workstations,ssh"`
  - `./nebula-cert sign -name "desktop" -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" -ip "192.168.100.50/24" -groups "workstations,ssh"`
  - `./nebula-cert sign -name "server" -ca-crt "CA/ca.crt" -ca-key "CA/ca.key" -ip "192.168.100.90/24" -groups "servers,admin"`

- Erstellt die Dateien `laptop.key` und `laptop.crt`, die auf dem Host *laptop* in der `config.yaml` Datei als `host.key` und `host.crt` benutzt werden.
- Erstellt die Dateien `desktop.key` und `desktop.crt`, die auf dem Host *desktop* in der `config.yaml` Datei als `host.key` und `host.crt` benutzt werden.
- Erstellt die Dateien `server.key` und `server.crt`, die auf dem Host *server* in der `config.yaml` Datei als `host.key` und `host.crt` benutzt werden.

Nebula Workshop #2
Datei /etc/nebula/config.yaml

- Auf Linux-basierten Nebula-Hosts wird die Datei `config.yaml` im Directory `/etc/nebula` benutzt, um den Nebula-Netzwerktreiber zu konfigurieren.
- 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-node.yml`

- Die Konfigurationsdatei `config.yaml` ist in elf Abschnitte gegliedert:
  - `pki, listen, tun, punchy, static_host_map, lighthouse, cipher, preferred_ranges, 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
    ```
    ```

Nebula Workshop #3a
config.yaml für 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
```

Nebula Workshop #3b
config.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
```

Nebula Workshop #4
Weitere Abschnitte in config.yaml

- Der Abschnitt **punchy** wird benutzt, um "NAT hole punching" einzu-schalten 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
  ```

Nebula Workshop #5
Firewall Regeln in config.yaml

- Im Abschnitt **`firewall`** werden die Security-Gruppen in Firewall-Regeln angewendet.

- ```
  # Nebula security group
  #   configuration 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
  ```

Nebula Workshop #6
Starten und Stoppen unter Linux

- 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
  - `curl -o /etc/system/system/nebula.service https://github.com/slackhq/nebula/raw/master/examples/service_scripts/nebula.service`
- … und den Service starten
  - `systemctl start nebula.service`
  - `systemctl enable nebula.service`

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

Nebula Workshop #7
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`
- Logging in `config.yaml` Datei konfigurieren

NEBULA Demo

- Letztes Jahr (2021) habe ich 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: Rahnenhof → DÜW und Rahnenhof → LU

Fragerunde:
Was wollt ihr noch wissen?

- Andere Mesh-artige VPNs 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)

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
- LinuxBlog.xyz
  - VPN mesh with Nebula

Noch mehr Infos zu **NEBULA**

- 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
- Unreality.xyz
  - Nebula Mesh with SSO

NEBULA Fundstellen bei Github

- 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

Vielen Dank!

MICHAEL DÖRR

(SEIT 1982 BEI DER AG MC)