3.9 Ports und Transportprotokolle prüfen

Ein erreichbarer Host bedeutet noch nicht, dass der benötigte Dienst erreichbar ist. `ping` prüft hauptsächlich die IP-Erreichbarkeit über ICMP. Es prüft weder einen TCP-Port noch zuverlässig die Funktionsfähigkeit eines Anwendungsdienstes.

Die zentrale Frage dieser Seite lautet:

> Ist der benötigte Dienst über das richtige Transportprotokoll, die richtige Zieladresse und den richtigen Port erreichbar?

---

1. Sicherheits- und Aktionskennzeichnungen

Kennzeichnung	Bedeutung
[RO]	Rein lesender Befehl, der normalerweise keine Konfiguration verändert
[TEST]	Aktiver Test, der Netzwerkverkehr oder Protokollanfragen erzeugt
[PRIV]	Erhöhte Rechte beziehungsweise Administrator- oder Root-Rechte können erforderlich sein
[FILE]	Der Befehl schreibt Ausgaben in eine Datei
[SENS]	Die Ausgabe kann sensible Daten enthalten
[CHANGE]	Der Befehl verändert eine Konfiguration
[DISRUPT]	Der Befehl kann einen Dienst oder eine Verbindung beeinträchtigen

Wichtiger Grundsatz: Portscans und aktive Verbindungstests dürfen nur auf Systemen durchgeführt werden, für die eine entsprechende Berechtigung vorliegt.

---

2. TCP, UDP und ICMP unterscheiden

Protokoll	Eigenschaft	Typisches Verhalten bei einem Test
TCP	Verbindungsorientiert	Vor der Datenübertragung wird eine Verbindung aufgebaut

Protokoll	Eigenschaft	Typisches Verhalten bei einem Test
UDP	Verbindungslos	Datagramme werden ohne vorherigen Verbindungsaufbau gesendet
ICMP	Kontroll- und Diagnoseprotokoll	Übermittelt beispielsweise Echo-Antworten oder Fehlermeldungen
ICMPv6	Kontrollprotokoll für IPv6	Wird unter anderem für Fehler, Neighbor Discovery und Path-MTU-Ermittlung benötigt

TCP-Verbindungsaufbau:

```
Client                    Server
  |                         |
  | -------- SYN ----------> |
  | <----- SYN, ACK ------------ |
  | -------- ACK ----------> |
  |                         |
  |    Verbindung aufgebaut    |
```

Der erfolgreiche TCP-Handshake beweist:

- Der Client konnte ein TCP-Segment zum Ziel senden.
- Das Ziel oder ein vorgeschaltetes System hat geantwortet.
- Der Rückweg zum Client funktioniert.
- Der betreffende TCP-Port akzeptiert grundsätzlich Verbindungen.

Er beweist jedoch noch nicht:

- dass die Anwendung fehlerfrei arbeitet,
- dass eine Anmeldung möglich ist,
- dass TLS-Zertifikate gültig sind,
- dass der Benutzer die benötigten Berechtigungen besitzt,
- dass die richtige Anwendung hinter dem Port antwortet.

Besonderheit bei UDP:

UDP besitzt keinen vergleichbaren Verbindungsaufbau. Bleibt eine Antwort aus, kann das bedeuten:

- Der Dienst ist nicht erreichbar.
- Der Port wird durch eine Firewall gefiltert.

- Der Dienst antwortet nur auf gültige Protokollanfragen.
- Die Antwort geht auf dem Rückweg verloren.
- Der UDP-Dienst antwortet grundsätzlich nicht auf die gesendeten Daten.
- Das Prüfwerkzeug kann den Zustand nicht eindeutig bestimmen.

Ein erfolgreicher UDP-Test benötigt deshalb möglichst eine **gültige Anfrage des jeweiligen Anwendungsprotokolls**, beispielsweise eine DNS-Abfrage anstelle eines beliebigen UDP-Pakets.

---

3. Port, Socket und Verbindung eindeutig beschreiben

Ein Port gehört immer zu einem Transportprotokoll. Die Angabe „Port 53“ ist ohne das Protokoll unvollständig, weil TCP-Port 53 und UDP-Port 53 getrennte Endpunkte sind.

Eine Netzwerkverbindung wird im Normalfall durch folgende Angaben unterschieden:

```
Transportprotokoll
Quell-IP-Adresse
Quellport
Ziel-IP-Adresse
Zielport
```

Beispiel:

```
TCP 192.0.2.25:53144 → 192.0.2.53:443
```

Dabei ist `53144` normalerweise ein temporärer Clientport und `443` der angesprochene Dienstport.

Wichtige Begriffe:

Begriff	Bedeutung
Dienstport	Port, auf dem eine Serveranwendung Verbindungen oder Datagramme erwartet
Quellport	Vom Client verwendeter Port; häufig dynamisch vergeben
Listening Socket	Lokaler Socket, der auf eingehende TCP-Verbindungen wartet

Begriff	Bedeutung
UDP Endpoint	Lokaler UDP-Endpunkt, der Datagramme empfangen kann
Ephemeral Port	Temporärer, vom Betriebssystem vergebener Clientport
Loopback-Adresse	Nur lokal erreichbare Adresse, beispielsweise `127.0.0.1` oder `::1`
Wildcard-Adresse	Bindung an mehrere beziehungsweise alle lokalen Adressen
Verbindungsstatus	Zustand einer TCP-Verbindung, beispielsweise `LISTEN` oder `ESTABLISHED`

---

4. Häufig verwendete Ports einordnen

Die folgende Tabelle enthält typische Standardbelegungen. Anwendungen können jedoch auf abweichende Ports konfiguriert werden.

Dienst	Transportprotokoll	Standardport
SSH	TCP	22
SMTP	TCP	25
DNS	UDP und TCP	53
DHCP-Server	UDP	67
DHCP-Client	UDP	68
HTTP	TCP	80
Kerberos	UDP und TCP	88
NTP	UDP	123
LDAP	TCP und teilweise UDP	389
HTTPS	TCP	443
SMB	TCP	445
LDAPS	TCP	636
Microsoft SQL Server	TCP	1433
MySQL/MariaDB	TCP	3306
RDP	TCP und UDP	3389
PostgreSQL	TCP	5432

Nicht allein auf die Portnummer verlassen: Ein offener TCP-Port 443 beweist nicht, dass dort tatsächlich ein ordnungsgemäßer HTTPS-Dienst läuft.

---

5. Typische Fehlermeldungen richtig interpretieren

Beobachtung	Wahrscheinliche Bedeutung
Verbindung erfolgreich	TCP-Verbindungsaufbau war möglich
`Connection refused`	Ziel wurde erreicht, aber der Port wird nicht angenommen oder aktiv abgelehnt
`Connection timed out`	Keine verwertbare Antwort; Paketverlust, Filterung, Routing- oder Rückwegproblem möglich
`No route to host`	Lokales oder entferntes Routingproblem beziehungsweise entsprechende ICMP-Rückmeldung
`Network is unreachable`	Keine geeignete Route oder Schnittstelle vorhanden
`Host is unreachable`	Ziel oder nächster Hop konnte nicht erreicht werden
`Name or service not known`	Namensauflösung fehlgeschlagen; der Porttest wurde möglicherweise noch nicht ausgeführt
`Address already in use`	Ein anderer Prozess oder Socket verwendet bereits die Adresse beziehungsweise den Port
`Permission denied`	Fehlende Berechtigung oder Sicherheitsrichtlinie verhindert den Zugriff
Verbindung sofort zurückgesetzt	Anwendung, Zielsystem oder Sicherheitskomponente sendet ein TCP-RST
TCP-Test erfolgreich, Anwendung fehlerhaft	Fehler liegt wahrscheinlich oberhalb der Transportschicht

Fehlermeldungen können je nach Betriebssystem, Anwendung und Sprache abweichen.

---

6. Auf dem Server prüfen, ob der Port geöffnet wurde

Zuerst wird direkt auf dem betroffenen Server geprüft:

1. Läuft der erwartete Prozess?
2. Lauscht er auf dem erwarteten Port?
3. Verwendet er TCP oder UDP?
4. An welche IP-Adresse wurde der Socket gebunden?
5. Ist der Dienst nur über IPv4, nur über IPv6 oder über beides erreichbar?

6. Stimmt der Prozess tatsächlich mit dem erwarteten Dienst überein?

Aufgabe	Windows	Linux	macOS
TCP-Verbindungen und Listener	`[RO] Get-NetTCPConnection`	`[RO] ss -tan`	`[RO] netstat -anv -p tcp`
Nur TCP-Listener	`[RO] Get-NetTCPConnection -State Listen`	`[RO] ss -ltn`	`[RO] lsof -nP -iTCP -sTCP:LISTEN`
UDP-Endpunkte	`[RO] Get-NetUDPEndpoint`	`[RO] ss -lun`	`[RO] lsof -nP -iUDP`
TCP-Listener mit Prozess	`[RO] Get-NetTCPConnection -State Listen`	`[RO][PRIV] sudo ss -ltnp`	`[RO][PRIV] sudo lsof -nP -iTCP -sTCP:LISTEN`
UDP-Endpunkte mit Prozess	`[RO] Get-NetUDPEndpoint`	`[RO][PRIV] sudo ss -lunp`	`[RO][PRIV] sudo lsof -nP -iUDP`
Klassische Übersicht	`[RO] netstat -ano`	`[RO] ss -tuln`	`[RO] netstat -anv`

`lsof` ist auf einigen Linux-Systemen nicht standardmäßig installiert. Unter Linux ist `ss` normalerweise das bevorzugte Werkzeug.

Windows – bestimmten TCP-Port prüfen:

```
[RO] Get-NetTCPConnection -LocalPort 443 -ErrorAction SilentlyContinue
```

Windows – nur Listener auf einem bestimmten Port:

```
[RO] Get-NetTCPConnection -State Listen -LocalPort 443 -ErrorAction SilentlyContinue
```

Windows – UDP-Endpunkt prüfen:

```
[RO] Get-NetUDPEndpoint -LocalPort 53 -ErrorAction SilentlyContinue
```

Windows – Prozess zu einem Listener ermitteln:

```
[RO] Get-NetTCPConnection -State Listen -LocalPort 443 |
    Select-Object LocalAddress, LocalPort, State, OwningProcess
```

Anschließend die ermittelte Prozess-ID einsetzen:

```
[RO] Get-Process -Id 4321
```

Alternativ mit klassischen Werkzeugen:

```
[RO] netstat -ano
[RO] tasklist /FI "PID eq 4321"
```

Linux – TCP-Port 443 prüfen:

```
[RO] ss -ltn 'sport = :443'
```

Linux – UDP-Port 53 prüfen:

```
[RO] ss -lun 'sport = :53'
```

Linux – Prozessinformationen anzeigen:

```
[RO][PRIV] sudo ss -ltnp 'sport = :443'
```

```
[RO][PRIV] sudo ss -lunp 'sport = :53'
```

macOS – TCP-Port 443 prüfen:

```
[RO] lsof -nP -iTCP:443 -sTCP:LISTEN
```

macOS – UDP-Port 53 prüfen:

```
[RO][PRIV] sudo lsof -nP -iUDP:53
```

Achtung: Eine leere Ausgabe bedeutet meistens, dass kein passender Socket gefunden wurde. Sie ist nicht automatisch ein Programmfehler.

---

7. Bind-Adressen eines Dienstes beurteilen

Ein Prozess kann laufen und trotzdem nur über eine falsche oder zu stark eingeschränkte Adresse erreichbar sein.

Lokale Adresse	Typische Bedeutung
`127.0.0.1:8080`	Nur über IPv4-Loopback auf demselben System erreichbar
`::1:8080`	Nur über IPv6-Loopback auf demselben System erreichbar
`192.0.2.10:443`	Nur über diese konkrete lokale IPv4-Adresse erreichbar
`0.0.0.0:443`	An alle passenden lokalen IPv4-Adressen gebunden
`[::]:443`	IPv6-Wildcard; ob zusätzlich IPv4 angenommen wird, hängt von Betriebssystem und Anwendung ab

Typischer Fehler:

```
Dienst läuft
↓
Port ist lokal geöffnet
↓
Dienst lauscht aber nur auf 127.0.0.1
↓
Lokaler Test funktioniert
↓
Entfernter Zugriff schlägt fehl
```

Eine Bindung an `0.0.0.0` oder `[::]` bedeutet nicht automatisch, dass der Port von außen erreichbar ist. Firewalls, VLANs, Routing, NAT und Sicherheitsrichtlinien gelten weiterhin.

---

8. TCP-Port von einem Client aus prüfen

Der Test sollte möglichst von dem System oder Netzwerksegment ausgeführt werden, in dem der Fehler tatsächlich auftritt.

Betriebssystem	TCP-Porttest
Windows	`[TEST] Test-NetConnection server.example.internal -Port 443 -InformationLevel Detailed`
Linux	`[TEST] nc -vz -w 5 server.example.internal 443`
macOS	`[TEST] nc -vz -w 5 server.example.internal 443`
Plattformübergreifend mit Nmap	`[TEST] nmap -sT -p 443 server.example.internal`

Windows:

```
[TEST] Test-NetConnection server.example.internal -Port 443 -InformationLevel Detailed
```

Besonders relevante Felder:

```
ComputerName
RemoteAddress
RemotePort
InterfaceAlias
SourceAddress
TcpTestSucceeded
```

Nur das Wesentliche ausgeben:

```
[TEST] Test-NetConnection server.example.internal -Port 443 |
    Select-Object ComputerName, RemoteAddress, RemotePort, SourceAddress, TcpTestSucceeded
```

Wichtig: `Test-NetConnection -Port` prüft einen **TCP-Port**. Es ist kein allgemeiner UDP-Porttest.

Linux und macOS:

```
[TEST] nc -vz -w 5 server.example.internal 443
```

Mehrere TCP-Ports einzeln prüfen:

```
[TEST] nc -vz -w 5 server.example.internal 22
[TEST] nc -vz -w 5 server.example.internal 80
[TEST] nc -vz -w 5 server.example.internal 443
```

Die verfügbaren `nc`-Optionen unterscheiden sich zwischen Implementierungen. Im Zweifel die lokale Hilfe prüfen:

```
[RO] nc -h
```

Nmap – einzelnen TCP-Port prüfen:

```
[TEST] nmap -sT -p 443 server.example.internal
```

Mehrere festgelegte TCP-Ports prüfen:

```
[TEST] nmap -sT -p 22,80,443 server.example.internal
```

Nmap nur im freigegebenen Umfang einsetzen. Ein vollständiger Portscan ist für die Prüfung eines bekannten Dienstes normalerweise nicht erforderlich.

---

9. Nmap-Portzustände richtig interpretieren

Zustand	Typische Bedeutung
`open`	Eine Anwendung nimmt Verbindungen oder Datagramme auf diesem Port an
`closed`	Ziel ist erreichbar, aber auf dem Port lauscht kein Dienst
`filtered`	Nmap kann wegen Paketfilterung nicht sicher feststellen, ob der Port geöffnet ist
`unfiltered`	Port ist erreichbar, aber der konkrete Offen-/Geschlossen-Zustand wurde mit der verwendeten Scanmethode nicht bestimmt
`open|filtered`	Nmap kann nicht zwischen geöffnet und gefiltert unterscheiden
`closed|filtered`	Nmap kann nicht zwischen geschlossen und gefiltert unterscheiden

Die Bewertung hängt von Scanart, Berechtigungen, Zielsystem und den empfangenen Antworten ab.

---

10. UDP-Dienste sinnvoll prüfen

Ein allgemeiner UDP-Porttest ist nur eingeschränkt aussagekräftig. Nach Möglichkeit sollte immer das eigentliche Anwendungsprotokoll geprüft werden.

Dienst	Sinnvoller Funktionstest
DNS	`nslookup`, `Resolve-DnsName` oder `dig`

Dienst	Sinnvoller Funktionstest
NTP	`w32tm`, `chronyc`, `ntpq` oder `sntp`, abhängig vom System
DHCP	DHCP-Ablauf und Paketmitschnitt analysieren
Syslog über UDP	Empfang auf dem Syslog-Server und Paketmitschnitt prüfen
SNMP	Autorisierte SNMP-Abfrage mit gültigen Parametern durchführen

DNS gezielt über UDP testen:

Windows:

```
[TEST] Resolve-DnsName example.org -Server 192.0.2.53 -Type A -DnsOnly
```

Linux und macOS, sofern `dig` installiert ist:

```
[TEST] dig @192.0.2.53 example.org A
```

DNS gezielt über TCP testen:

Windows:

```
[TEST] Resolve-DnsName example.org -Server 192.0.2.53 -Type A -DnsOnly -TcpOnly
```

Linux und macOS:

```
[TEST] dig +tcp @192.0.2.53 example.org A
```

Netcat-UDP-Test:

```
[TEST] nc -vzu -w 3 192.0.2.53 53
```

Dieser Test ist nicht mit einem erfolgreichen DNS-Funktionstest gleichzusetzen. Eine scheinbare Erfolgsmeldung kann lediglich bedeuten, dass lokal kein unmittelbarer Fehler festgestellt wurde.

Nmap-UDP-Test:

```
[TEST][PRIV] sudo nmap -sU -p 53 192.0.2.53
```

UDP-Scans können langsam und mehrdeutig sein. Ein Ergebnis wie `open|filtered` ist bei UDP häufig, wenn weder eine Protokollantwort noch eine eindeutige ICMP-Fehlermeldung empfangen wurde.

---

11. Nicht nur den Port, sondern die Anwendung testen

Ein Porttest prüft die Transportschicht. Danach sollte ein Test mit dem tatsächlichen Anwendungsprotokoll folgen.

Anwendung	Beispiel
HTTP	`[TEST] curl -v http://server.example.internal/`
HTTPS	`[TEST] curl -vk https://server.example.internal/`
HTTPS mit regulärer Zertifikatsprüfung	`[TEST] curl -v https://server.example.internal/`
TLS-Handshake	`[TEST] openssl s_client -connect server.example.internal:443 -servername server.example.internal`
DNS über UDP	`[TEST] dig @192.0.2.53 example.org A`
DNS über TCP	`[TEST] dig +tcp @192.0.2.53 example.org A`
SSH-Protokolltest	`[TEST] ssh -vvv user@server.example.internal`

HTTP-Header abrufen:

```
[TEST] curl -I https://server.example.internal/
```

Ausführliche HTTPS-Diagnose:

```
[TEST][SENS] curl -v https://server.example.internal/
```

Bei `curl -v` können Header, Cookies, interne Hostnamen und weitere sensible Informationen sichtbar werden. Zugangsdaten oder Sitzungstoken dürfen nicht ungeprüft dokumentiert werden.

TLS-Verbindung untersuchen:

```
[TEST][SENS] openssl s_client \
    -connect server.example.internal:443 \
    -servername server.example.internal
```

`-servername` übermittelt den Servernamen per SNI. Das ist wichtig, wenn mehrere TLS-Websites dieselbe IP-Adresse verwenden.

Hinweis zu `curl -k`:

```
[TEST] curl -vk https://server.example.internal/
```

`-k` deaktiviert die Zertifikatsprüfung. Das kann zur Eingrenzung eines Zertifikatsfehlers verwendet werden, darf aber nicht als dauerhafte Lösung betrachtet werden.

---

12. IPv4 und IPv6 getrennt prüfen

Ein Hostname kann gleichzeitig eine IPv4- und eine IPv6-Adresse besitzen. Dadurch kann derselbe Porttest je nach ausgewählter Adresse unterschiedlich ausfallen.

Namensauflösung kontrollieren:

Windows:

```
[RO] Resolve-DnsName server.example.internal
```

Linux:

```
[RO] getent ahosts server.example.internal
```

macOS:

```
[RO] dscacheutil -q host -a name server.example.internal
```

HTTP gezielt über IPv4 testen:

```
[TEST] curl -4 -v https://server.example.internal/
```

HTTP gezielt über IPv6 testen:

```
[TEST] curl -6 -v https://server.example.internal/
```

Netcat gezielt über IPv4 oder IPv6:

```
[TEST] nc -4 -vz -w 5 server.example.internal 443
[TEST] nc -6 -vz -w 5 server.example.internal 443
```

Windows – aufgelöste Zieladresse beachten:

```
[TEST] Test-NetConnection server.example.internal -Port 443 -InformationLevel Detailed
```

In der Ausgabe muss `RemoteAddress` kontrolliert werden. Ein erfolgreicher IPv4-Test beweist nicht, dass IPv6 funktioniert – und umgekehrt.

---

13. TCP-Zustände beurteilen

TCP-Zustand	Bedeutung	Diagnosehinweis
`LISTEN`	Socket wartet auf eingehende Verbindungen	Dienst ist lokal grundsätzlich gebunden
`SYN-SENT`	Client hat SYN gesendet und wartet	Viele dauerhafte Einträge können auf fehlende Antworten hindeuten
`SYN-RECEIVED`	SYN wurde empfangen, SYN-ACK gesendet	Viele Einträge können auf fehlende abschließende ACKs hinweisen
`ESTABLISHED`	TCP-Verbindung besteht	Transportverbindung funktioniert grundsätzlich
`FIN-WAIT-1`	Aktives Schließen wurde begonnen	Kurzzeitig normal
`FIN-WAIT-2`	Eigenes FIN wurde bestätigt	Dauerhafte Häufung kann auf Probleme der Gegenstelle hinweisen
`CLOSE-WAIT`	Gegenstelle hat geschlossen, lokale Anwendung noch nicht	Viele dauerhafte Einträge deuten häufig auf ein Anwendungsproblem hin
`LAST-ACK`	Lokale Seite wartet auf Bestätigung ihres FIN	Kurzzeitig normal
`TIME-WAIT`	Verbindung bleibt vorübergehend gespeichert	Viele Einträge können bei hoher Verbindungsrate normal sein

TCP-Zustand	Bedeutung	Diagnosehinweis
`CLOSED`	Keine Verbindung vorhanden	Normaler Endzustand

Windows – bestehende TCP-Verbindungen:

```
[RO] Get-NetTCPConnection -State Established
```

Windows – Verbindungen zu einem bestimmten Zielport:

```
[RO] Get-NetTCPConnection -RemotePort 443
```

Linux – bestehende TCP-Verbindungen:

```
[RO] ss -tn state established
```

Linux – Verbindungen mit Zustand und Timern:

```
[RO] ss -tano
```

macOS – TCP-Zustände:

```
[RO] netstat -anv -p tcp
```

Ein einzelner Zustand ist nur eine Momentaufnahme. Bei sporadischen Problemen sind wiederholte Beobachtungen, Anwendungsprotokolle und gegebenenfalls ein Paketmitschnitt erforderlich.

---

14. TCP-Paketmuster interpretieren

Beobachtung im Mitschnitt	Typische Interpretation
`SYN → SYN/ACK → ACK`	TCP-Verbindung wurde aufgebaut
Wiederholte SYN-Pakete ohne Antwort	Paketverlust, Filterung, falsches Routing, ausgefallenes Ziel oder fehlerhafter Rückweg
`SYN → RST/ACK`	Ziel ist erreichbar, aber Port ist geschlossen oder wird aktiv abgelehnt

Beobachtung im Mitschnitt	Typische Interpretation
SYN-ACK kommt an, abschließendes ACK fehlt	Problem auf der Clientseite oder beim Rückweg des ACK möglich
Verbindung wird aufgebaut und sofort mit RST beendet	Anwendung oder Sicherheitskomponente bricht Verbindung ab
Viele Retransmissions	Paketverlust, Überlastung, fehlerhafte Verbindung oder asymmetrische Erfassung möglich
TCP-Verbindung erfolgreich, HTTP-Fehler folgt	Transportschicht funktioniert; Fehler liegt wahrscheinlich auf Anwendungsebene

Ein Paketmitschnitt sollte möglichst gleichzeitig auf Client und Server erfolgen. Dadurch lässt sich feststellen, an welcher Stelle Pakete verloren gehen oder verändert werden.

---

15. Paketmitschnitt für einen Port erstellen

Paketmitschnitte können IP-Adressen, Hostnamen, Nutzdaten, Cookies und andere vertrauliche Informationen enthalten. Speicherung und Weitergabe müssen den betrieblichen Datenschutz- und Sicherheitsvorgaben entsprechen.

Linux – TCP-Port 443 mitschneiden:

```
[TEST][PRIV][FILE][SENS] sudo tcpdump -i any -nn \
    'host 192.0.2.20 and tcp port 443' \
    -w port-443.pcap
```

Linux – UDP-Port 53 mitschneiden:

```
[TEST][PRIV][FILE][SENS] sudo tcpdump -i any -nn \
    'host 192.0.2.53 and udp port 53' \
    -w dns-udp.pcap
```

macOS – vorher Schnittstellen ermitteln:

```
[RO] networksetup -listallhardwareports
```

Anschließend beispielsweise:

```
[TEST][PRIV][FILE][SENS] sudo tcpdump -i en0 -nn \
    'host 192.0.2.20 and tcp port 443' \
    -w port-443.pcap
```

Windows – verfügbare Schnittstellen mit Dumpcap anzeigen:

```
[RO] dumpcap -D
```

Windows – Mitschnitt auf Schnittstelle 1:

```
[TEST][PRIV][FILE][SENS] dumpcap -i 1 `
    -f "host 192.0.2.20 and tcp port 443" `
    -w port-443.pcapng
```

Die Schnittstellennummer `1` ist nur ein Beispiel und muss vorher mit `dumpcap -D` ermittelt werden.

---

16. Nützliche Wireshark-Anzeigefilter

Aufgabe	Wireshark-Anzeigefilter
TCP-Port 443	`tcp.port == 443`
UDP-Port 53	`udp.port == 53`
Verkehr zu oder von einer IP-Adresse	`ip.addr == 192.0.2.20`
IPv6-Adresse	`ipv6.addr == 2001:db8::20`
IP-Adresse und TCP-Port	`ip.addr == 192.0.2.20 && tcp.port == 443`
TCP-SYN-Pakete	`tcp.flags.syn == 1`
Nur erste SYN-Pakete ohne gesetztes ACK	`tcp.flags.syn == 1 && tcp.flags.ack == 0`
TCP-Reset	`tcp.flags.reset == 1`
Vermutete TCP-Wiederholungen	`tcp.analysis.retransmission`
Doppelte ACKs	`tcp.analysis.duplicate_ack`
TCP-Verbindungsaufbau	`tcp.connection.syn
ICMP-Meldungen	`icmp`
ICMPv6-Meldungen	`icmpv6`

Wireshark-Analysehinweise wie `tcp.analysis.retransmission` sind Interpretationen anhand des vorhandenen Mitschnitts. Fehlende Pakete am Beginn oder während der Aufzeichnung können zu irreführenden Markierungen führen.

---

17. Container und veröffentlichte Ports prüfen

Bei Containerdiensten müssen mehrere Ebenen unterschieden werden:

```
Client
  ↓
Host-IP und veröffentlichter Hostport
  ↓
Portweiterleitung oder Proxy
  ↓
Container-IP und Containerport
  ↓
Anwendungsprozess im Container
```

Laufende Container und Portzuordnungen anzeigen:

```
[RO] docker ps --format 'table {{.Names}}\t{{.Ports}}'
```

Portzuordnung eines Containers anzeigen:

```
[RO] docker port webserver
```

Container detailliert untersuchen:

```
[RO][SENS] docker inspect webserver
```

Zu prüfen sind:

- Ist der Container gestartet?
- Ist der benötigte Containerport veröffentlicht?
- Wurde der richtige Hostport verwendet?
- Ist der Hostport nur an `127.0.0.1` gebunden?
- Lauscht die Anwendung innerhalb des Containers?

- Verwendet die Anwendung im Container die richtige Bind-Adresse?
- Existiert eine vorgeschaltete Firewall oder ein Reverse Proxy?
- Ist der Zugriff nur innerhalb eines Container-Netzwerks vorgesehen?

Eine Docker-Ausgabe wie diese:

```
127.0.0.1:8080->80/tcp
```

bedeutet, dass der veröffentlichte Hostport normalerweise nur über die Loopback-Adresse des Docker-Hosts erreichbar ist.

Eine Ausgabe wie:

```
0.0.0.0:8080->80/tcp
```

zeigt eine Veröffentlichung über die passenden IPv4-Adressen des Hosts. Ob ein entfernter Zugriff erlaubt ist, hängt zusätzlich von Firewall, Routing und Netzwerkrichtlinien ab.

---

18. Typische Fehlerbilder systematisch eingrenzen

Fall A – Auf dem Server existiert kein Listener

```
Porttest vom Client schlägt fehl
↓
Auf dem Server ist kein LISTEN-Socket vorhanden
↓
Dienststatus, Dienstkonfiguration und Protokolldateien prüfen
```

Mögliche Ursachen:

- Dienst wurde nicht gestartet.
- Dienst ist abgestürzt.
- Falscher Port wurde konfiguriert.
- Dienst konnte den Port nicht binden.
- Ein anderer Prozess verwendet den Port.
- Dienst lauscht nur auf einer anderen Adresse oder einem anderen Protokoll.

Fall B – Dienst lauscht nur auf Loopback

Lokaler Test erfolgreich

Entfernter Test nicht erfolgreich

Listener: 127.0.0.1:PORT oder ::1:PORT

Wahrscheinliche Ursache:

- Der Dienst ist ausschließlich für lokale Verbindungen konfiguriert.

Fall C – Listener vorhanden, entfernter Test läuft in einen Timeout

Mögliche Ursachen:

- Host-Firewall filtert den Port.
- Netzwerk-Firewall oder Access Control List filtert den Port.
- Falsches VLAN oder fehlerhaftes Routing.
- Rückweg zum Client fehlt.
- NAT- oder Portweiterleitungsregel fehlt.
- Es wird die falsche IP-Adresse getestet.
- Dienst lauscht nur auf IPv4 oder nur auf IPv6.
- Zielsystem ist über einen anderen Netzwerkpfad erreichbar als erwartet.

Fall D – `Connection refused`

Mögliche Ursachen:

- Kein Prozess lauscht auf dem Zielport.
- Dienst lauscht nur auf einer anderen IP-Adresse.
- Aktive Firewall-Ablehnung.
- Portweiterleitung zeigt auf ein Ziel ohne Listener.
- Anwendung wurde während des Tests beendet.

Fall E – TCP-Test erfolgreich, Anwendung funktioniert nicht

Die Netz- und Transportschicht funktionieren zumindest grundsätzlich. Anschließend prüfen:

- TLS-Handshake und Zertifikatskette
- Server Name Indication
- HTTP-Statuscode
- Authentifizierung
- Benutzerberechtigung
- Reverse Proxy
- Backend-Erreichbarkeit

- Anwendungskonfiguration
- Datenbankverbindung
- Anwendungsprotokolle

Fall F – Nur manche Clients sind betroffen

Vergleich zwischen funktionierendem und betroffenem Client:

- aufgelöste Ziel-IP-Adresse,
- IPv4 oder IPv6,
- Quell-IP-Adresse,
- VLAN,
- Standardgateway,
- Routingtabelle,
- Proxykonfiguration,
- lokale Firewall,
- VPN-Verbindung,
- DNS-Suffix und DNS-Server,
- Zeitpunkt des Tests.

Fall G – UDP-Test liefert kein eindeutiges Ergebnis

Vorgehen:

1. Gültige Anfrage des Anwendungsprotokolls senden.
2. Gleichzeitig auf Client und Server mitschneiden.
3. Prüfen, ob die Anfrage den Server erreicht.
4. Prüfen, ob der Server eine Antwort erzeugt.
5. Prüfen, ob die Antwort den Client erreicht.
6. ICMP- beziehungsweise ICMPv6-Fehlermeldungen beachten.

---

19. Empfohlener Diagnoseablauf

Schritt	Prüfung	Ergebnisfrage
1	Zielhost, Zieladresse, Transportprotokoll und Port bestimmen	Was soll genau erreicht werden?
2	Namensauflösung kontrollieren	Wird die erwartete IP-Adresse verwendet?
3	Routing und Quelladresse kontrollieren	Wird der erwartete Netzwerkpfad verwendet?

Schritt	Prüfung	Ergebnisfrage
4	Dienststatus auf dem Server prüfen	Läuft die Anwendung?
5	Lokalen Listener oder UDP-Endpunkt prüfen	Lauscht der Dienst auf Port und Adresse?
6	Prozess dem Socket zuordnen	Gehört der Port zur erwarteten Anwendung?
7	Lokal auf dem Server testen	Funktioniert der Dienst lokal?
8	Vom betroffenen Clientsegment testen	Ist der Transportweg funktionsfähig?
9	IPv4 und IPv6 getrennt testen	Ist nur eine Adressfamilie betroffen?
10	Anwendungsprotokoll testen	Antwortet der eigentliche Dienst korrekt?
11	Firewall, ACL, NAT und Proxy prüfen	Wird der Verkehr unterwegs beeinflusst?
12	Paketmitschnitt erstellen	Wo endet der erfolgreiche Paketfluss?
13	Vergleich mit funktionierendem System	Welche relevante Abweichung besteht?
14	Ergebnis dokumentieren	Ist die Diagnose reproduzierbar?

Grundregel:

Listener vorhanden

≠

Port aus jedem Netz erreichbar

≠

Anwendung funktionsfähig

≠

Benutzer kann den Dienst erfolgreich verwenden

Jede dieser Aussagen muss getrennt geprüft werden.

---

20. Kompakte Befehlstabelle

Aufgabe	Windows	Linux	macOS
TCP-Listener anzeigen	`[RO] Get-NetTCPConnection -State Listen`	`[RO] ss -ltn`	`[RO] lsof -nP -iTCP -sTCP:LISTEN`

Aufgabe	Windows	Linux	macOS
UDP-Endpunkte anzeigen	`[RO] Get-NetUDPEndpoint`	`[RO] ss -lun`	`[RO] lsof -nP -iUDP`
Listener mit Prozess	`[RO] Get-NetTCPConnection -State Listen`	`[RO][PRIV] sudo ss -ltnp`	`[RO][PRIV] sudo lsof -nP -iTCP -sTCP:LISTEN`
TCP-Port 443 lokal prüfen	`[RO] Get-NetTCPConnection -State Listen -LocalPort 443`	`[RO] ss -ltn 'sport = :443'`	`[RO] lsof -nP -iTCP:443 -sTCP:LISTEN`
TCP-Port entfernt testen	`[TEST] Test-NetConnection HOST -Port 443`	`[TEST] nc -vz -w 5 HOST 443`	`[TEST] nc -vz -w 5 HOST 443`
HTTP testen	`[TEST] curl.exe -v http://HOST/`	`[TEST] curl -v http://HOST/`	`[TEST] curl -v http://HOST/`
HTTPS testen	`[TEST] curl.exe -v https://HOST/`	`[TEST] curl -v https://HOST/`	`[TEST] curl -v https://HOST/`
TLS-Handshake	Falls OpenSSL installiert: `[TEST] openssl s_client -connect HOST:443 -servername HOST`	`[TEST] openssl s_client -connect HOST:443 -servername HOST`	`[TEST] openssl s_client -connect HOST:443 -servername HOST`
TCP-Port mit Nmap	`[TEST] nmap -sT -p 443 HOST`	`[TEST] nmap -sT -p 443 HOST`	`[TEST] nmap -sT -p 443 HOST`
UDP-Port mit Nmap	Administrator-Konsole: `[TEST][PRIV] nmap -sU -p 53 HOST`	`[TEST][PRIV] sudo nmap -sU -p 53 HOST`	`[TEST][PRIV] sudo nmap -sU -p 53 HOST`
Bestehende TCP-Verbindungen	`[RO] Get-NetTCPConnection -State Established`	`[RO] ss -tn state established`	`[RO] netstat -anv -p tcp`
Prozess über PID suchen	`[RO] Get-Process -Id PID`	`[RO] ps -fp PID`	`[RO] ps -p PID -o pid,ppid,user,command`

`HOST`, `PID`, IP-Adressen und Ports müssen durch die Werte des konkreten Störungsfalls ersetzt werden.

---

21. Änderungen erst nach gesicherter Diagnose durchführen

Nicht vorschnell:

- die Firewall vollständig deaktivieren,
- beliebige Ports dauerhaft freigeben,
- Dienste unkontrolliert neu starten,
- Sicherheitssoftware abschalten,
- Listener an alle Adressen binden,
- Ports ohne Freigabe scannen,
- NAT- oder Firewallregeln ohne Dokumentation verändern.

Stattdessen:

1. Fehlerzustand dokumentieren.

2. Erwarteten Sollzustand bestimmen.
3. Ursache möglichst eindeutig nachweisen.
4. Änderung genehmigen lassen.
5. Nur die erforderliche Änderung durchführen.
6. Funktion und Sicherheit anschließend erneut prüfen.
7. Rückfallmöglichkeit und Ergebnis dokumentieren.

Ein temporär deaktivierter Paketfilter kann zwar eine Hypothese bestätigen, erzeugt aber ein Sicherheitsrisiko und kann den ursprünglichen Zustand verändern. Besser ist eine gezielte Auswertung von Regeln, Protokollen und Paketmitschnitten.

---

22. Dokumentationsvorlage für Port- und Transportfehler

```
Störung:
Zeitpunkt:
Betroffener Benutzer beziehungsweise Standort:
Clientname:
Client-IP-Adresse:
Client-VLAN:
Client-Betriebssystem:

Zielhostname:
Aufgelöste Zieladresse:
Verwendete Adressfamilie: IPv4 / IPv6
Transportprotokoll: TCP / UDP
Zielport:
Erwarteter Dienst:

Lokaler Listener vorhanden: Ja / Nein
Bind-Adresse:
Zugehöriger Prozess:
Prozess-ID:
Dienststatus:

Lokaler Funktionstest:
Entfernter Porttest:
Anwendungsprotokolltest:
TCP- beziehungsweise UDP-Ergebnis:
```

Beobachtete Fehlermeldung:

Firewall- oder ACL-Prüfung:

Paketmitschnitt vorhanden:

Beobachtetes Paketmuster:

Vergleich mit funktionierendem System:

Festgestellte Ursache:

Durchgeführte Änderung:

Änderung genehmigt durch:

Ergebnis der Nachprüfung:

Offene Punkte:

Vor der Ablage müssen Passwörter, Sitzungstoken, personenbezogene Daten und andere vertrauliche Inhalte entfernt oder geschützt werden.

---

23. Kontrollfragen nach der Diagnose

- Wurde wirklich der richtige Host getestet?
- Wurde die tatsächlich aufgelöste IP-Adresse dokumentiert?
- Wurde zwischen TCP und UDP unterschieden?
- Lauscht der Dienst auf dem erwarteten Port?
- Gehört der Listener zur erwarteten Anwendung?
- Lauscht der Dienst an der richtigen IP-Adresse?
- Wurden IPv4 und IPv6 getrennt betrachtet?
- Wurde vom tatsächlich betroffenen Netzsegment getestet?
- Wurde ein Anwendungstest zusätzlich zum Porttest durchgeführt?
- Wurde bei UDP eine gültige Protokollanfrage verwendet?
- Wurden Rückweg und Firewallregeln berücksichtigt?
- Wurde ein Timeout nicht vorschnell als „Port geschlossen“ bezeichnet?
- Wurde ein erfolgreicher TCP-Test nicht mit einer funktionsfähigen Anwendung gleichgesetzt?
- Ist die Ursache durch Messergebnisse belegt?
- Wurde der Zustand nach einer Änderung erneut geprüft?

---

24. Quellen und weiterführende Dokumentation

- Microsoft Learn – Get-NetTCPConnection: https://learn.microsoft.com/powershell/module/nettcpip/get-nettcpconnection

- Microsoft Learn – Get-NetUDPEndpoint: https://learn.microsoft.com/powershell/module/nettcpip/get-netudpendpoint
- Microsoft Learn – Test-NetConnection: https://learn.microsoft.com/powershell/module/nettcpip/test-netconnection
- Linux-Handbuch – `ss(8)`: https://man7.org/linux/man-pages/man8/ss.8.html
- RFC 9293 – Transmission Control Protocol: https://www.rfc-editor.org/rfc/rfc9293.html
- RFC 768 – User Datagram Protocol: https://www.rfc-editor.org/rfc/rfc768.html
- IANA – Service Name and Transport Protocol Port Number Registry: https://www.iana.org/assignments/service-names-port-numbers/
- Nmap – Port Scanning Basics: https://nmap.org/book/man-port-scanning-basics.html
- Nmap – Port Scanning Techniques: https://nmap.org/book/man-port-scanning-techniques.html
- Wireshark – TCP Display Filter Reference: https://www.wireshark.org/docs/dfref/t/tcp.html
- Wireshark – UDP Display Filter Reference: https://www.wireshark.org/docs/dfref/u/udp.html
- Docker Docs – Publishing and exposing ports: https://docs.docker.com/get-started/docker-concepts/running-containers/publishing-ports/
