12.3 tcpdump, tshark und Paketmitschnitt auf der Kommandozeile

Neben Wireshark gibt es wichtige Werkzeuge für Paketmitschnitte auf der Kommandozeile.

Besonders wichtig sind:

```
- tcpdump
- tshark
- netstat / ss
- curl
- nc
- ping
- traceroute / tracert
- nslookup / dig
```

Diese Werkzeuge sind besonders nützlich, wenn man auf Servern, Firewalls, Routern oder per SSH arbeitet.

Merksatz:

```
Wireshark ist grafisch.
tcpdump und tshark arbeiten auf der Kommandozeile.
```

Warum Kommandozeilenwerkzeuge wichtig sind

In der Praxis hat man nicht immer eine grafische Oberfläche.

Beispiele:

```
Linux-Server per SSH
Firewall-Konsole
Container-Host
Cloud-Server
NAS-System
Router
Recovery-Umgebung
minimale Serverinstallation
```

Dort sind Kommandozeilenwerkzeuge oft schneller und direkter.

Merksatz:

```
Auf Servern ist die Kommandozeile oft das wichtigste Analysewerkzeug.
```

---

tcpdump

tcpdump ist ein Werkzeug, mit dem Netzwerkverkehr auf der Kommandozeile mitgeschnitten werden kann.

Es kann:

```
- Pakete live anzeigen
- Pakete nach Host filtern
- Pakete nach Port filtern
- Pakete nach Protokoll filtern
- Mitschnitte in Dateien speichern
- PCAP-Dateien erzeugen
```

Diese PCAP-Dateien kann man später mit Wireshark öffnen.

Merksatz:

```
tcpdump zeichnet Pakete auf und kann PCAP-Dateien erzeugen.
```

---

tcpdump-Grundidee

tcpdump lauscht auf einer Netzwerkschnittstelle.

Beispiele für Schnittstellen:

```
eth0
ens18
en0
wlan0
br0
docker0
vlan10
```

Wichtig:

```
Man muss die richtige Schnittstelle auswählen.
```

Wenn man auf der falschen Schnittstelle mitschneidet, sieht man den relevanten Verkehr nicht.

Merksatz:

```
Richtige Schnittstelle wählen,
sonst ist der Mitschnitt wertlos.
```

Schnittstellen anzeigen

Vor einem Mitschnitt prüft man, welche Schnittstellen vorhanden sind.

Typische Idee:

```
tcpdump -D
```

Damit werden verfügbare Interfaces angezeigt.

Danach wählt man die passende Schnittstelle aus.

Merksatz:

```
Erst Schnittstellen anzeigen,
dann Mitschnitt starten.
```

Einfacher tcpdump-Mitschnitt

Ein einfacher Mitschnitt auf einer Schnittstelle:

```
tcpdump -i eth0
```

Bedeutung:

```
-i eth0:
auf Schnittstelle eth0 mitschneiden
```

Dieser Mitschnitt zeigt viele Pakete live auf der Konsole.

Merksatz:

```
tcpdump -i Schnittstelle startet einen Live-Mitschnitt.
```

Namensauflösung vermeiden

tcpdump kann IP-Adressen und Ports in Namen auflösen.

Das kann die Ausgabe verlangsamen oder verwirren.

Daher nutzt man häufig:

```
-n
```

oder:

```
-nn
```

Bedeutung:

```
keine Namensauflösung
keine Portnamenauflösung
```

Beispiel:

```
tcpdump -nn -i eth0
```

Merksatz:

```
-nn zeigt IPs und Ports unverändert an.
```

Paketinhalt ausführlicher anzeigen

Mit mehr Ausführlichkeit sieht man mehr Details.

Typisch:

```
-v
-vv
-vvv
```

Beispiel:

```
tcpdump -nn -vv -i eth0
```

Aber:

```
mehr Details sind nicht immer übersichtlicher.
```

Merksatz:

```
Mehr Ausgabe hilft nur,
wenn man sie auch gezielt auswertet.
```

In Datei speichern

Für spätere Analyse speichert man Mitschnitte in eine Datei.

Beispiel:

```
tcpdump -nn -i eth0 -w capture.pcap
```

Bedeutung:

```
-w capture.pcap:
Mitschnitt in Datei schreiben
```

Die Datei kann später mit Wireshark geöffnet werden.

Merksatz:

```
Mit -w speichert tcpdump einen Mitschnitt als Datei.
```

Datei wieder lesen

Eine gespeicherte PCAP-Datei kann mit tcpdump gelesen werden.

Beispiel:

```
tcpdump -nn -r capture.pcap
```

Bedeutung:

```
-r:
Datei lesen
```

Merksatz:

```
Mit -r liest tcpdump gespeicherte Mitschnitte.
```

Nur bestimmten Host mitschneiden

Wenn nur ein bestimmter Host interessiert, filtert man nach IP-Adresse.

Beispiel:

```
tcpdump -nn -i eth0 host 192.168.10.20
```

Das zeigt Pakete, bei denen diese IP Quelle oder Ziel ist.

Merksatz:

```
host filtert Quelle oder Ziel.
```

Nur Quelle oder Ziel filtern

Quelle filtern:

```
tcpdump -nn -i eth0 src host 192.168.10.20
```

Ziel filtern:

```
tcpdump -nn -i eth0 dst host 192.168.10.20
```

Merksatz:

```
src = Quelle.
dst = Ziel.
```

Nach Netzwerk filtern

Man kann auch ganze Netze filtern.

Beispiel:

```
tcpdump -nn -i eth0 net 192.168.10.0/24
```

Das zeigt Verkehr, bei dem dieses Netz beteiligt ist.

Merksatz:

```
net filtert ein ganzes Subnetz.
```

Nach Port filtern

Beispiel:

```
tcpdump -nn -i eth0 port 443
```

Das zeigt Pakete, bei denen Port 443 als Quell- oder Zielport vorkommt.

Genauer:

```
tcpdump -nn -i eth0 dst port 443
```

zeigt Pakete, die zu Zielport 443 gehen.

Merksatz:

```
port zeigt Quell- oder Zielport.
dst port zeigt den Zielport.
```

TCP und UDP filtern

Man kann gezielt TCP oder UDP filtern.

Beispiel TCP:

```
tcpdump -nn -i eth0 tcp port 443
```

Beispiel UDP:

```
tcpdump -nn -i eth0 udp port 53
```

Wichtig:

```
TCP 53 und UDP 53 sind unterschiedlich.
```

Merksatz:

```
Protokoll und Port zusammen prüfen.
```

DNS mitschneiden

DNS nutzt häufig UDP 53, kann aber auch TCP 53 verwenden.

Beispiel:

```
tcpdump -nn -i eth0 port 53
```

Oder genauer:

```
tcpdump -nn -i eth0 udp port 53
```

Wichtig:

```
Bei großen Antworten oder Zonentransfers kann TCP 53 relevant sein.
```

Merksatz:

```
DNS nicht nur als UDP denken,
auch TCP 53 beachten.
```

HTTP und HTTPS mitschneiden

HTTP:

```
tcpdump -nn -i eth0 tcp port 80
```

HTTPS:

```
tcpdump -nn -i eth0 tcp port 443
```

Bei HTTPS sieht man den Inhalt nicht im Klartext, aber man sieht Verbindungsaufbau, IP-Adressen, Ports, TLS-Handshake und Fehler.

Merksatz:

```
HTTPS-Inhalte sind verschlüsselt,
aber Verbindungsdaten sind sichtbar.
```

SSH mitschneiden

SSH nutzt typischerweise TCP 22.

Beispiel:

```
tcpdump -nn -i eth0 tcp port 22
```

Man sieht:

```
Verbindungsaufbau
Pakete
```

```
Abbrüche
Timeouts
```

Man sieht nicht:

```
eingegebene Befehle im Klartext,
weil SSH verschlüsselt ist.
```

Merksatz:

```
SSH schützt Inhalte,
tcpdump sieht aber Verbindungsmetadaten.
```

ICMP mitschneiden

ICMP wird für Diagnose genutzt.

Beispiel:

```
tcpdump -nn -i eth0 icmp
```

Damit sieht man zum Beispiel:

```
Ping-Anfragen
Ping-Antworten
Destination Unreachable
Time Exceeded
```

Merksatz:

```
ICMP zeigt Diagnose- und Fehlermeldungen.
```

ARP mitschneiden

ARP ist wichtig im lokalen IPv4-Netz.

Beispiel:

```
tcpdump -nn -i eth0 arp
```

Damit sieht man ARP-Anfragen und ARP-Antworten.

Typische Frage:

```
Wer hat diese IP-Adresse?
```

Merksatz:

```
ARP hilft bei lokalen Erreichbarkeitsproblemen.
```

DHCP mitschneiden

DHCP nutzt UDP 67 und UDP 68.

Beispiel:

```
tcpdump -nn -i eth0 port 67 or port 68
```

Damit kann man den DORA-Ablauf sehen:

```
Discover
Offer
Request
Acknowledge
```

Merksatz:

```
DHCP-DORA lässt sich mit tcpdump gut prüfen.
```

Filter kombinieren

Filter können kombiniert werden.

Beispiel:

```
tcpdump -nn -i eth0 host 192.168.10.20 and tcp port 443
```

Bedeutung:

```
Zeige nur TCP-443-Verkehr mit Host 192.168.10.20.
```

Weitere logische Verknüpfungen:

```
and
or
not
```

Merksatz:

```
Mit and, or und not werden tcpdump-Filter gezielt.
```

Beispiel: Alles außer SSH anzeigen

Wenn man per SSH auf einem Server arbeitet, kann der eigene SSH-Verkehr die Anzeige stören.

Dann kann man SSH ausblenden:

```
tcpdump -nn -i eth0 not port 22
```

Merksatz:

```
Eigenen SSH-Verkehr bei Bedarf ausblenden.
```

Beispiel: Fehler reproduzieren

Gute Vorgehensweise:

```
1. Mitschnitt vorbereiten.
2. Mitschnitt starten.
3. Fehler gezielt auslösen.
4. Mitschnitt stoppen.
```

```
5. Datei sichern.
6. Mit Wireshark analysieren.
```

Merksatz:

```
Mitschnitt immer passend zum Fehlerzeitpunkt erstellen.
```

Mitschnitt begrenzen

Paketmitschnitte können schnell sehr groß werden.

Möglichkeiten:

```
nur bestimmte IP mitschneiden
nur bestimmten Port mitschneiden
nur kurze Zeit mitschneiden
Datei begrenzen
Rotation nutzen
nicht unnötig viele Daten erfassen
```

Merksatz:

```
So wenig wie möglich,
so viel wie nötig mitschneiden.
```

Warum PCAP-Dateien sensibel sind

PCAP-Dateien können enthalten:

```
- interne IP-Adressen
- MAC-Adressen
- Hostnamen
- Benutzernamen
- Cookies
- Tokens
- unverschlüsselte Passwörter
- E-Mail-Inhalte
- interne Protokolle
```

- Kundendaten

Deshalb:

sicher speichern
Zugriff begrenzen
nicht unnötig weitergeben
nach Zweck löschen

Merksatz:

PCAP-Dateien wie vertrauliche Daten behandeln.

---

tshark

tshark ist die Kommandozeilenversion von Wireshark.

Es nutzt ähnliche Analysefunktionen, aber ohne grafische Oberfläche.

tshark kann:

- live mitschneiden
- PCAP-Dateien lesen
- Display Filter verwenden
- Felder ausgeben
- automatisiert auswerten
- Protokolle analysieren

Merksatz:

tshark ist Wireshark für die Kommandozeile.

---

tshark für vorhandene Dateien

Eine vorhandene PCAP-Datei kann mit tshark gelesen werden.

Beispiel:

```
tshark -r capture.pcap
```

Mit Filtern kann man gezielt suchen.

Beispielidee:

```
nur DNS anzeigen
nur HTTP anzeigen
nur bestimmte IP anzeigen
```

Merksatz:

```
tshark eignet sich gut für schnelle PCAP-Auswertung.
```

---

tcpdump oder tshark?

Werkzeug	Stärke
tcpdump	schnell mitschneiden, auf vielen Systemen verfügbar
tshark	detailliertere Protokollauswertung auf Kommandozeile
Wireshark	grafische Analyse, sehr übersichtlich

Merksatz:

```
tcpdump zum Aufnehmen,
Wireshark oder tshark zum Analysieren.
```

---

ss

ss zeigt Netzwerk-Sockets auf Linux-Systemen.

Damit prüft man:

```
welche Dienste lauschen
welche Verbindungen bestehen
welche Ports offen sind
welche Prozesse Ports verwenden
```

Beispielidee:

```
Lauscht ein Dienst wirklich auf Port 443?
```

Merksatz:

```
ss zeigt lokale Ports und Verbindungen.
```

---

netstat

netstat ist ein älteres Werkzeug, das ähnliche Informationen wie ss anzeigen kann.

Es zeigt unter anderem:

```
- offene Ports
- aktive Verbindungen
- Routingtabelle
- Schnittstellenstatistiken
```

Auf modernen Linux-Systemen wird häufig ss bevorzugt.

Merksatz:

```
netstat ist älter,
ss ist auf Linux oft moderner.
```

Lauschende Dienste prüfen

Wenn ein Dienst nicht erreichbar ist, prüft man lokal:

```
Lauscht der Dienst überhaupt?
```

Beispielhafte Fragestellung:

```
Hört der Webserver auf TCP 80 oder TCP 443?
Hört SSH auf TCP 22?
Hört die Datenbank nur auf localhost?
```

```
Hört der Dienst auf der richtigen IP-Adresse?
```

Merksatz:

```
Dienst muss lokal lauschen,
bevor er extern erreichbar sein kann.
```

---

localhost-Falle

Ein Dienst kann nur auf localhost lauschen.

Beispiel:

```
127.0.0.1:8080
```

Dann ist er nur lokal auf dem System erreichbar, aber nicht von anderen Geräten im Netzwerk.

Für externe Erreichbarkeit muss er auf einer passenden Adresse lauschen.

Merksatz:

```
Dienst auf localhost ist nicht automatisch im Netzwerk erreichbar.
```

---

curl

curl ist ein Werkzeug zum Testen von HTTP, HTTPS und vielen anderen Protokollen.

Es kann zeigen:

```
- Statuscode
- Header
- Weiterleitungen
- TLS-Fehler
- Antwortinhalt
- API-Antworten
```

Merksatz:

```
curl testet Webdienste und APIs direkt.
```

---

curl und HTTP-Status

Mit curl kann man prüfen, welchen HTTP-Status ein Server zurückgibt.

Wichtige Fragen:

```
Kommt 200?
Kommt 301 oder 302?
Kommt 401?
Kommt 403?
Kommt 404?
Kommt 500?
Kommt 502?
```

Merksatz:

```
curl zeigt,
was ein Webserver tatsächlich antwortet.
```

---

curl und Header

HTTP-Header enthalten wichtige Informationen.

Beispiele:

```
Content-Type
Location
Server
Set-Cookie
Authorization
Cache-Control
Strict-Transport-Security
```

Mit curl kann man Header gezielt anzeigen.

Merksatz:

```
Header erklären oft,
warum ein Webdienst sich so verhält.
```

---

curl und TLS

curl kann TLS-Probleme sichtbar machen.

Typische Fehler:

```
Zertifikat abgelaufen
Name passt nicht
CA nicht vertrauenswürdig
TLS-Version nicht kompatibel
Zertifikatskette unvollständig
```

Merksatz:

```
curl hilft bei HTTPS- und Zertifikatsproblemen.
```

---

nc

nc steht für netcat.

Es kann verwendet werden, um einfache TCP- oder UDP-Verbindungen zu testen.

Typische Nutzung:

```
Ist ein TCP-Port erreichbar?
Antwortet ein einfacher Dienst?
Kann ich manuell Daten senden?
```

Beispielgedanke:

```
Porttest für TCP 443,
TCP 22
oder TCP 25.
```

Merksatz:

```
nc ist ein einfaches Werkzeug für Port- und Verbindungstests.
```

Porttest richtig einordnen

Ein erfolgreicher Porttest bedeutet:

```
TCP-Verbindung zum Port ist möglich.
```

Er bedeutet nicht automatisch:

```
Anwendung funktioniert korrekt.
Login funktioniert.
Zertifikat passt.
Berechtigung stimmt.
API antwortet richtig.
```

Merksatz:

```
Port offen ist nur ein Teil der Diagnose.
```

---

ping

ping nutzt ICMP Echo Request und Echo Reply.

Es prüft grob:

```
Ist ein Ziel per ICMP erreichbar?
```

Ping sagt nicht:

```
ob ein TCP-Port offen ist
ob DNS korrekt ist
ob HTTPS funktioniert
ob Anwendung antwortet
ob Anmeldung funktioniert
```

Merksatz:

```
Ping prüft nicht den Dienst.
```

---

traceroute und tracert

traceroute oder tracert zeigen den Weg zu einem Ziel.

Sie helfen bei:

```
Routingproblemen
Erreichbarkeitsproblemen
ungewöhnlichen Pfaden
Netzunterbrechungen
hoher Latenz
```

Wichtig:

```
Firewalls können traceroute-Ergebnisse beeinflussen.
```

Merksatz:

```
traceroute zeigt Hinweise auf den Weg,
aber nicht immer den vollständigen echten Pfad.
```

---

nslookup

nslookup prüft DNS-Auflösung.

Damit kann man fragen:

```
Welche IP liefert DNS für diesen Namen?
Welcher DNS-Server antwortet?
Gibt es überhaupt eine Antwort?
```

Gut für einfache DNS-Prüfung.

Merksatz:

```
nslookup prüft Namensauflösung.
```

---

dig

dig ist ein detaillierteres DNS-Werkzeug.

Es zeigt unter anderem:

```
- DNS-Antworten
- Record-Typen
- Antwortzeiten
- Nameserver
- TTL
- Autorität
- zusätzliche Antworten
```

dig ist besonders hilfreich, wenn DNS genauer analysiert werden soll.

Merksatz:

```
dig ist detaillierte DNS-Analyse.
```

---

openssl s_client

openssl s_client prüft TLS-Verbindungen.

Damit kann man sehen:

```
- Zertifikat
- Zertifikatskette
- TLS-Version
- Cipher
- Serverantwort
- SNI-Verhalten
```

Besonders nützlich bei:

```
HTTPS
LDAPS
IMAPS
SMTPS
POP3S
```

Merksatz:

```
openssl s_client hilft bei TLS- und Zertifikatsproblemen.
```

---

journalctl

journalctl zeigt Logs auf Systemen mit systemd.

Damit kann man prüfen:

```
Dienst gestartet?
Dienst abgestürzt?
Fehler beim Start?
Berechtigungsproblem?
Port bereits belegt?
Konfigurationsfehler?
```

Merksatz:

```
journalctl zeigt System- und Dienstlogs auf Linux.
```

Windows-Ereignisanzeige

Die Windows-Ereignisanzeige zeigt System- und Anwendungsereignisse.

Wichtige Bereiche:

```
Anwendung
Sicherheit
System
Setup
Dienstspezifische Protokolle
```

Hilfreich bei:

Anmeldefehlern
Dienstfehlern
RDP-Problemen
DNS- oder DHCP-Problemen
Zertifikatsproblemen

Merksatz:

Ereignisanzeige ist wichtig für Windows-Fehlersuche.

PowerShell Test-NetConnection

Unter Windows kann Test-NetConnection helfen.

Es prüft zum Beispiel:

Namensauflösung
TCP-Verbindung zu einem Port
Ziel-IP
grundlegende Erreichbarkeit

Beispielhafte Fragestellung:

Ist TCP 443 erreichbar?
Ist TCP 3389 erreichbar?

Merksatz:

Test-NetConnection ist ein Windows-Werkzeug für Porttests.

Werkzeuge richtig kombinieren

Kein Werkzeug beantwortet alles.

Beispiel Webdienst funktioniert nicht:

nslookup:
DNS prüfen

ping:
grobe Erreichbarkeit prüfen

Test-NetConnection oder nc:
Port prüfen

curl:
HTTP-Antwort prüfen

tcpdump:
Verkehr prüfen

Logs:
Dienstfehler prüfen

Merksatz:

Diagnose entsteht durch Kombination mehrerer Werkzeuge.

Beispiel: HTTPS funktioniert nicht

Sinnvolle Prüfung:

1. DNS mit nslookup oder dig prüfen.
2. Port TCP 443 mit nc oder Test-NetConnection prüfen.
3. HTTP-Antwort mit curl prüfen.
4. Zertifikat mit curl oder openssl s_client prüfen.
5. tcpdump oder Wireshark bei Verbindungsproblemen nutzen.
6. Webserver- oder Reverse-Proxy-Logs prüfen.

Merksatz:

HTTPS-Probleme mit DNS,
Port,
TLS,

```
Anwendung
und Logs prüfen.
```

Beispiel: SSH funktioniert nicht

Sinnvolle Prüfung:

```
1. Ziel-IP oder Hostname prüfen.
2. TCP 22 prüfen.
3. Lokalen SSH-Dienst prüfen.
4. Firewall und Host-Firewall prüfen.
5. Benutzer und Schlüssel prüfen.
6. SSH-Logs prüfen.
7. tcpdump bei Timeout oder unklarer Verbindung nutzen.
```

Merksatz:

```
SSH-Probleme mit Port,
Dienst,
Benutzer,
Schlüssel
und Logs prüfen.
```

Beispiel: DNS funktioniert nicht

Sinnvolle Prüfung:

```
1. Welcher DNS-Server ist eingetragen?
2. Ist DNS-Server erreichbar?
3. Antwortet UDP 53?
4. Antwortet bei Bedarf TCP 53?
5. Liefert DNS richtige Antwort?
6. Gibt es NXDOMAIN?
7. Gibt es falschen Cache?
8. tcpdump zeigt echte DNS-Anfrage?
```

Merksatz:

DNS-Fehler mit Server,

Port,

Antwort

und Cache prüfen.

Beispiel: DHCP funktioniert nicht

Sinnvolle Prüfung:

1. Client sendet Discover?
2. DHCP-Server antwortet mit Offer?
3. Client sendet Request?
4. Server sendet Acknowledge?
5. Scope hat freie Adressen?
6. DHCP-Relay korrekt?
7. VLAN korrekt?
8. Firewall blockiert UDP 67/68?

Merksatz:

DHCP mit DORA analysieren.

Beispiel: RDP funktioniert nicht

Sinnvolle Prüfung:

1. Hostname oder IP prüfen.
2. TCP 3389 prüfen.
3. Remote Desktop aktiviert?
4. Host-Firewall erlaubt?
5. Benutzer berechtigt?
6. VPN oder Gateway nötig?
7. Windows-Ereignisanzeige prüfen.
8. Nicht direkt aus Internet veröffentlichen.

Merksatz:

RDP-Probleme mit Port,
Dienst,
Berechtigung
und Sicherheit prüfen.

Typische Fehler bei Werkzeugnutzung

Häufige Fehler:

- nur ping testen und Dienstproblem übersehen
- DNS nicht prüfen
- TCP und UDP verwechseln
- falsche Schnittstelle mitschneiden
- falschen Port prüfen
- internen und externen Test verwechseln
- NAT nicht berücksichtigen
- Logs ignorieren
- verschlüsselte Inhalte im Klartext erwarten

Merksatz:

Werkzeugergebnisse immer fachlich einordnen.

Diagnose nach Schichten

Schicht	typische Werkzeuge
1	Link-Status, Kabeltest, Switchport
2	ARP, MAC-Tabelle, VLAN-Prüfung
3	ping, traceroute, Routingtabelle
4	nc, Test-NetConnection, ss, tcpdump
5–6	TLS-Prüfung, openssl s_client
7	curl, nslookup, dig, Logs, Anwendungstest

Merksatz:

Das Werkzeug richtet sich nach der vermuteten Schicht.

Was Kommandozeilenwerkzeuge nicht ersetzen

Sie ersetzen nicht:

- saubere Dokumentation
- Verständnis der Netzstruktur
- Firewall-Regelprüfung
- Berechtigungsprüfung
- Dienstkonfiguration
- Monitoring
- Sicherheitskonzept
- Datenschutz beim Mitschnitt

Merksatz:

Werkzeuge liefern Hinweise,
Fachwissen macht daraus eine Diagnose.

Typische IHK-Fragen

In AP1 und AP2 kann zum Beispiel gefragt werden:

- Wofür nutzt man tcpdump?
- Was ist eine PCAP-Datei?
- Wofür steht tshark?
- Was ist der Unterschied zwischen tcpdump und Wireshark?
- Warum muss man die richtige Schnittstelle wählen?
- Wie filtert man nach Host, Port oder Protokoll?
- Warum ist tcpdump mit -w nützlich?
- Warum sind Mitschnittdateien sensibel?
- Wofür nutzt man curl?
- Wofür nutzt man nc?
- Was prüft ping?
- Warum reicht ping nicht aus?
- Wofür nutzt man nslookup oder dig?
- Wofür nutzt man openssl s_client?
- Wie kombiniert man Werkzeuge bei der Fehlersuche?

Typische Prüfungsfallen

tcpdump ist Kommandozeile.

Wireshark ist grafisch.

tshark ist Wireshark auf Kommandozeile.

tcpdump kann PCAP-Dateien speichern.

PCAP-Dateien können sensible Daten enthalten.

Falsche Schnittstelle bedeutet falsche Sicht.

-nn verhindert Namensauflösung.

host filtert Quelle oder Ziel.

src ist Quelle.

dst ist Ziel.

port filtert Quell- oder Zielport.

TCP und UDP getrennt betrachten.

DNS kann UDP und TCP 53 nutzen.

DHCP nutzt UDP 67 und 68.

Ping prüft ICMP,
nicht den Dienst.

Porttest beweist nicht,
dass Anwendung funktioniert.

curl prüft Webdienste besser als ping.

openssl s_client hilft bei TLS.

Logs und Mitschnitt ergänzen sich.

Werkzeuge ersetzen keine fachliche Analyse.

Wichtige Begriffe kurz erklärt

Begriff	Kurze Erklärung
tcpdump	Kommandozeilenwerkzeug für Paketmitschnitte
tshark	Wireshark-Funktion auf Kommandozeile
PCAP	Datei mit aufgezeichneten Paketen
Interface	Netzwerkschnittstelle
Capture	Aufnahme von Paketen
-i	Schnittstelle auswählen
-w	Mitschnitt in Datei schreiben
-r	Mitschnitt aus Datei lesen
-n / -nn	Namensauflösung abschalten
host	Quelle oder Ziel filtern
src	Quelle
dst	Ziel
port	Port filtern
ss	lokale Sockets und Ports anzeigen
netstat	älteres Werkzeug für Netzwerkstatus
curl	Web- und API-Anfragen testen
nc	einfache Port- und Verbindungstests
ping	ICMP-Erreichbarkeit prüfen
traceroute	Weg zum Ziel prüfen
nslookup	DNS einfach prüfen
dig	DNS detailliert prüfen
openssl s_client	TLS-Verbindungen prüfen
journalctl	Linux-Systemlogs anzeigen
Ereignisanzeige	Windows-Logs anzeigen
Test-NetConnection	Windows-Porttest

IHK-sichere Kurzformulierung

tcpdump ist ein Kommandozeilenwerkzeug zum Mitschneiden von Netzwerkverkehr und kann Mitschnitte als PCAP-Datei speichern, die später mit Wireshark analysiert werden kann. tshark ist die Kommandozeilenversion von Wireshark und eignet sich für detaillierte Auswertungen ohne grafische Oberfläche. Bei der Analyse muss die richtige Netzwerkschnittstelle gewählt und gezielt nach Host, Port, Protokoll oder Netz gefiltert werden. Ergänzende Werkzeuge wie curl, nc, ping, traceroute, nslookup, dig, openssl s_client, ss, netstat und Logdateien helfen, DNS, Ports, TLS, Dienste und Verbindungen systematisch zu prüfen. Ein einzelnes Werkzeug reicht selten aus; eine zuverlässige Diagnose entsteht durch die Kombination mehrerer Hinweise.

Merksätze

tcpdump = Paketmitschnitt auf Kommandozeile.

tshark = Wireshark auf Kommandozeile.

Wireshark = grafische Analyse.

PCAP = gespeicherter Paketmitschnitt.

Richtige Schnittstelle ist entscheidend.

-nn verhindert Namensauflösung.

-w schreibt Mitschnitt in Datei.

-r liest Mitschnitt aus Datei.

host filtert Quelle oder Ziel.

src filtert Quelle.

dst filtert Ziel.

port filtert Quell- oder Zielport.

TCP und UDP nicht verwechseln.

DNS kann UDP und TCP nutzen.

DHCP nutzt UDP 67 und 68.

SSH nutzt typischerweise TCP 22.

HTTPS nutzt typischerweise TCP 443.

Ping prüft ICMP,
nicht Anwendung.

Porttest prüft Erreichbarkeit,
nicht Anwendungserfolg.

curl prüft HTTP und HTTPS.

nc testet einfache Verbindungen.

nslookup und dig prüfen DNS.

openssl s_client prüft TLS.

ss zeigt lokale Ports.

Logs zeigen Dienstfehler.

Mitschnitt und Logs gemeinsam auswerten.

PCAP-Dateien vertraulich behandeln.

Werkzeuge liefern Hinweise,
Fachwissen liefert die Diagnose.