

Arbeitsblatt: Netzwerkverkehr beobachten

Pakete und TCP-Stack mit Wireshark, tcpdump, ss und netstat analysieren

Lernziele

Nach Bearbeitung dieses Arbeitsblattes können Sie:

- den Datenverkehr Ihrer Python-Programme aufzeichnen und im Detail analysieren
- zwischen UDP und TCP anhand der Paketstruktur unterscheiden
- den TCP-Verbindungsaufbau (3-Wege-Handshake) und -abbau im Mitschnitt identifizieren
- offene Sockets und ihren Zustand mit ss bzw. netstat anzeigen
- entscheiden, welches Werkzeug für welche Fragestellung geeignet ist

Voraussetzungen

- Linux-Mint-VM mit Internetzugriff und sudo-Rechten
- Ihre Python-Programme: udp_server.py, udp_client.py, tcp_server.py, tcp_client.py
- Mehrere geöffnete Terminals (Tipp: tmux oder mehrere Tabs)

Hinweis

Alle Werkzeuge benötigen für den Mitschnitt Root-Rechte, weil sie an die Netzwerkschnittstelle „herangehen“. ss und netstat funktionieren weitgehend auch ohne sudo.

Werkzeuge im Überblick

Werkzeug	Zweck	Ebene	Wann nutzen?
Wireshark	Grafischer Paket-Sniffer mit Filter und Dekodierung	L2–L7	Wenn Sie Pakete genau ansehen wollen
tcpdump	CLI-Paket-Sniffer, ideal über SSH und auf Servern	L2–L7	Schnell, ohne GUI, auch zum Aufzeichnen
ss	Zeigt Sockets und TCP-Zustände an	L4	Aktueller Zustand des TCP-Stacks
netstat	Klassiker; ähnlich wie ss, aber älter	L4	Wenn ss nicht verfügbar ist
lsof	Welcher Prozess hat welchen Socket offen?	OS	Zuordnung Socket ↔ Prozess

Teil A — Vorbereitung

Schritt A.1 Werkzeuge installieren

Öffnen Sie ein Terminal und installieren Sie die benötigten Pakete:

```
bash
sudo apt update
sudo apt install -y wireshark tcpdump iproute2 net-tools lsof
```

Während der Installation fragt Wireshark, ob Nicht-Superuser Pakete mitschneiden dürfen. Antworten Sie mit **Ja** und fügen Sie sich der Gruppe wireshark hinzu:

```
bash
sudo usermod -aG wireshark $USER
# Anschließend einmal ab- und wieder anmelden (oder: newgrp wireshark)
```

Schritt A.2 Netzwerkschnittstellen anzeigen

Finden Sie heraus, auf welcher Schnittstelle Sie mitschneiden müssen:

```
bash
ip -brief address
# oder
ip a
```

Bei einer VM ist die Schnittstelle meist `ens33`, `enp0s3` oder ähnlich. Für lokale Tests (alles auf einer VM) ist außerdem `lo` (Loopback) wichtig.

Ihre Schnittstelle(n) und IP-Adresse(n):

Achtung

Wenn Client und Server auf derselben VM laufen, fließt der Verkehr über das Loopback-Interface (lo, 127.0.0.1). Wireshark/tcpdump müssen dann auf lo mitlesen — sonst sehen Sie nichts!

Teil B — Wireshark

Wireshark ist der bekannteste grafische Paket-Sniffer. Er zeichnet Pakete auf, dekodiert sie und stellt sie übersichtlich dar.

Schritt B.1 Wireshark starten und Schnittstelle wählen

1. Starten Sie Wireshark über das Anwendungsmenü oder mit dem Befehl `wireshark`.
2. Wählen Sie im Startfenster die Schnittstelle, auf der Ihr Verkehr läuft (z. B. `lo` für lokale Tests).
3. Klicken Sie auf die blaue Haifischflosse (Start), um die Aufzeichnung zu beginnen.
4. Mit der roten Stop-Schaltfläche beenden Sie die Aufzeichnung.

Schritt B.2 Anzeigefilter nutzen

Das Filterfeld oberhalb der Paketliste schränkt die Anzeige ein (Pakete werden trotzdem aufgezeichnet). Wichtige Filter:

Filter	Bedeutung
<code>udp</code>	Nur UDP-Pakete
<code>tcp</code>	Nur TCP-Pakete
<code>udp.port == 5005</code>	UDP-Pakete auf Port 5005 (Quelle ODER Ziel)

tcp.port == 6000	TCP-Pakete auf Port 6000
ip.addr == 127.0.0.1	Pakete von oder zu 127.0.0.1
tcp.flags.syn == 1	Alle Pakete mit gesetztem SYN-Flag
tcp.flags.fin == 1	Alle Pakete mit gesetztem FIN-Flag
tcp.stream eq 0	Alle Pakete der ersten TCP-Verbindung

**Tipp**

Rechtsklick auf ein Paket → „Folgen“ → „TCP-Stream“ zeigt Ihnen den kompletten Datenaustausch einer Verbindung als Text.

Schritt B.3 UDP aufzeichnen

1. Wireshark auf Ihrer Schnittstelle starten. Filter: udp.port == <Ihr_port>
2. In einem zweiten Terminal den UDP-Server starten.
3. In einem dritten Terminal den UDP-Client starten und eine Nachricht senden.
4. Aufzeichnung stoppen und ein UDP-Paket anklicken.

**Aufgabe**

Klappen Sie im Detail-Bereich „User Datagram Protocol“ auf. Notieren Sie Source Port, Destination Port und Length. Vergleichen Sie mit Ihrem Python-Quelltext.

Wie viele Pakete sind insgesamt entstanden, wenn der Client EINE Nachricht sendet? Begründen Sie.

Notizen UDP:

Schritt B.4 TCP aufzeichnen — der 3-Wege-Handshake

Jetzt geht es ans Herzstück von TCP. Setzen Sie den Filter auf tcp.port == <Ihr_port> und führen Sie nacheinander aus:

1. Wireshark starten (Aufzeichnung läuft)
2. TCP-Server starten
3. TCP-Client starten, eine Nachricht senden, Verbindung schließen
4. Aufzeichnung stoppen

Sie sollten — vor dem Datenaustausch — drei Pakete mit folgenden Flags sehen:

#	Flags	Bedeutung	Wer sendet?
1	SYN	Verbindungswunsch	Client → Server
2	SYN, ACK	Server bestätigt und stellt eigene Anfrage	Server → Client

3	ACK	Client bestätigt — Verbindung steht	Client → Server
---	-----	-------------------------------------	-----------------

Den Verbindungsabbau erkennen Sie an den FIN- und ACK-Flags. Häufig wird der FIN mit dem letzten Daten-ACK kombiniert.

Aufgabe

Markieren Sie in Ihrer Aufzeichnung das SYN-, SYN/ACK- und ACK-Paket. Notieren Sie die Sequenznummern (Sequence Number, Acknowledgment Number).

Welche Zahl steht im SYN/ACK in „Acknowledgment Number“ im Verhältnis zur „Sequence Number“ aus dem SYN? Warum?

Notizen TCP-Handshake:

Tipp

Wireshark zeigt Ihnen relative Sequenznummern (Start bei 0). Über Edit → Preferences → Protocols → TCP können Sie „Relative sequence numbers“ deaktivieren und sehen die echten Werte.

Teil C — tcpdump

tcpdump ist das Kommandozeilen-Pendant zu Wireshark. Es ist auf nahezu jedem Linux-System verfügbar und perfekt, wenn Sie per SSH auf einem Server arbeiten.

Schritt C.1 Grundgerüst eines tcpdump-Aufrufs

```
bash
sudo tcpdump -i <interface> [filter-ausdruck]
# Beispiel:
sudo tcpdump -i lo udp port 5005
```

Wichtige Optionen:

Option	Wirkung
-i <if>	Schnittstelle (z. B. lo, ens33, any für alle)
-n	IP-Adressen nicht in Namen auflösen (schneller, klarer)
-nn	Zusätzlich Portnummern statt Dienstnamen anzeigen
-v / -vv	Mehr Detail im Output
-X	Paketinhalt zusätzlich als Hex und ASCII anzeigen

-c <n>	Nur n Pakete erfassen, dann beenden
-w datei	Aufzeichnung in pcap-Datei schreiben (z. B. für Wireshark)
-r datei	Aufzeichnung aus pcap-Datei lesen

Schritt C.2 Filterausdrücke (BPF-Syntax)

Achtung: Die tcpdump-Filter sind **nicht dieselben** wie die Wireshark-Anzeigefilter. Beispiele:

```
bash
# Nur UDP auf Port 5005
sudo tcpdump -i lo -nn udp port 5005

# Nur TCP zwischen 127.0.0.1 und einem bestimmten Port
sudo tcpdump -i lo -nn tcp and host 127.0.0.1 and port 6000

# Nur Pakete mit gesetztem SYN-Flag (Verbindungsaufbau)
sudo tcpdump -i lo -nn "tcp[tcpflags] & tcp-syn != 0"

# Inhalt mitschneiden (Hex + ASCII)
sudo tcpdump -i lo -nnX udp port 5005
```

Schritt C.3 Aufzeichnen für Wireshark

Eine sehr praktische Kombination: mit tcpdump aufzeichnen, in Wireshark analysieren:

```
bash
sudo tcpdump -i lo -w mein_mitschnitt.pcap tcp port 6000
# Strg+C zum Beenden, dann:
wireshark mein_mitschnitt.pcap
```

Aufgabe

Starten Sie Ihren TCP-Server und -Client und zeichnen Sie den Verkehr mit tcpdump in eine .pcap-Datei auf.

Öffnen Sie die Datei in Wireshark. Finden Sie denselben 3-Wege-Handshake?

Wie viele Bytes umfasst ein „leeres“ TCP-Paket (z. B. ein reines ACK ohne Nutzdaten)?

Notizen tcpdump:

Teil D — ss und netstat: Sockets und TCP-Zustände

Wireshark und tcpdump zeigen Ihnen, was über die Leitung geht. ss und netstat zeigen Ihnen, was der TCP-Stack des Betriebssystems gerade „im Kopf hat“ — also welche Sockets existieren und in welchem Zustand sie sind.

Schritt D.1 ss — der moderne Ersatz für netstat

Wichtige Optionen:

Option	Wirkung
-t	Nur TCP-Sockets
-u	Nur UDP-Sockets
-l	Nur lauschende (LISTEN) Sockets
-a	Alle Sockets (auch lauschende UND verbundene)
-n	Keine Namensauflösung (Zahlen statt Diensten)
-p	Zugehörigen Prozess anzeigen (sudo nötig)
-o	Timer-Informationen anzeigen
-s	Zusammenfassung / Statistik

```
bash
# Alle lauschenden TCP-Sockets mit Prozessinfo
sudo ss -tlnp

# Alle UDP-Sockets
sudo ss -ulnp

# Alle TCP-Verbindungen in jeglichem Zustand
ss -tan

# Statistik über alle Sockets des Systems
ss -s
```

Schritt D.2 TCP-Zustände erkennen

In der Spalte „State“ zeigt ss Ihnen den aktuellen TCP-Zustand. Die wichtigsten:

Zustand	Bedeutung
LISTEN	Socket wartet auf eingehende Verbindungen (typisch beim Server)
SYN-SENT	Client hat SYN geschickt, wartet auf SYN/ACK
SYN-RCV	Server hat SYN bekommen und SYN/ACK gesendet
ESTAB	Verbindung steht und überträgt Daten
FIN-WAIT-1/2	Wir haben FIN gesendet, warten auf Bestätigung des Partners
CLOSE-WAIT	Partner hat FIN gesendet, wir müssen noch close() aufrufen
TIME-WAIT	Verbindung geschlossen; OS wartet 2×MSL, falls noch Pakete eintrudeln
CLOSED	Endzustand — Socket existiert nicht mehr

Hinweis

Die kurzen Zustände wie SYN-SENT und SYN-RCV sehen Sie in der Praxis kaum — sie sind nur Bruchteile einer Sekunde aktiv. Lange ESTAB- oder TIME-WAIT-Zustände sind dagegen leicht zu beobachten.

Schritt D.3 netstat — der Klassiker

netstat liefert sehr ähnliche Informationen. Wichtige Optionen sind fast identisch:

```
bash
# Lauschende TCP-Sockets mit Prozess
sudo netstat -tlnp

# Alle Sockets mit Prozess
sudo netstat -anp

# Routing-Tabelle
netstat -rn

# Statistik
netstat -s
```

Unter modernen Distributionen ist ss bevorzugt, weil es schneller ist und besser mit großen Verbindungsmengen umgeht. netstat gehört zum Paket net-tools und muss eventuell nachinstalliert werden.

Aufgabe

Starten Sie Ihren TCP-Server. Finden Sie den lauschenden Socket mit ss -tlnp. Welchen Port verwendet er? Welcher Prozess hält ihn?

Starten Sie zusätzlich Ihren TCP-Client und beobachten Sie während er aktiv ist mit ss -tan, dass eine Verbindung im Zustand ESTAB existiert.

Beenden Sie den Client. Welchen Zustand sehen Sie direkt nach dem Beenden? Wie lange bleibt dieser Zustand bestehen?

Notizen ss / netstat:

Teil E — lsof (Bonus): Welcher Prozess hat den Port?

Wenn Sie wissen wollen, welcher Prozess gerade einen bestimmten Port belegt — z. B. weil Ihre bind()-Aufrufe mit „Address already in use“ fehlschlagen — hilft lsof:

```
bash
sudo lsof -i :5005          # Alles auf Port 5005
sudo lsof -i tcp:6000       # Nur TCP auf 6000
sudo lsof -i -P -n         # Alle Netz-Sockets, ohne Namensauflösung
```

Tipp

Mit ss geht das auch: sudo ss -tlnp | grep 6000 zeigt Ihnen den Server-Prozess.

Teil F — Synthese: UDP vs. TCP im Vergleich

Nutzen Sie die Werkzeuge, um die folgenden Beobachtungen zu sammeln und zu vergleichen.

Aufgabe

Schicken Sie mit Ihrem UDP-Client eine einzelne Nachricht und zeichnen Sie mit. Wie viele Pakete entstehen?

Schicken Sie dieselbe Nachricht mit Ihrem TCP-Client. Wie viele Pakete entstehen jetzt insgesamt (inklusive Handshake und Abbau)?

Schließen Sie den UDP-Server. Schickt der Client jetzt trotzdem ein Paket? Was zeigen Wireshark / ss?

Schließen Sie den TCP-Server. Was passiert beim TCP-Client beim Verbindungsversuch? Welche Pakete und welche Zustände sehen Sie?

Beobachtungen UDP:

Beobachtungen TCP:

Was sagt Ihnen der direkte Vergleich über die beiden Protokolle?

Teil G — Spickzettel

Aufzeichnen mit tcpdump

```
bash
sudo tcpdump -i lo -nn udp port 5005
sudo tcpdump -i lo -nn tcp port 6000
sudo tcpdump -i lo -nnX tcp port 6000      # mit Inhalt
sudo tcpdump -i lo -w out.pcap tcp port 6000 # in Datei
```

Wireshark-Anzeigefilter

```
udp.port == 5005
tcp.port == 6000
tcp.flags.syn == 1
tcp.flags.fin == 1
tcp.stream eq 0
ip.addr == 127.0.0.1
```

ss

```
bash
sudo ss -tlnp      # lauschende TCP-Sockets mit Prozess
sudo ss -ulnp      # lauschende UDP-Sockets mit Prozess
ss -tan           # alle TCP-Sockets in jedem Zustand
ss -s             # Statistik
```

netstat

```
bash
sudo netstat -tlnp # lauschende TCP-Sockets
sudo netstat -anp  # alle Sockets
netstat -s         # Statistik
```

lsof

```
bash
sudo lsof -i :<port>
sudo lsof -i tcp:<port>
```

Reflexion

Beantworten Sie die folgenden Fragen kurz und in eigenen Worten:

1. Welches Werkzeug würden Sie nutzen, um auf einem Server ohne grafische Oberfläche per SSH den Netzwerkverkehr zu analysieren? Warum?

2. Warum reicht ss/netstat nicht, wenn Sie verstehen wollen, welche Daten konkret übertragen wurden?

3. Welche zwei zentralen Unterschiede zwischen UDP und TCP konnten Sie in den Mitschnitten direkt sehen?
