

Lab Guide

Four exercises across four traffic sources: tcpdump capture analysis, DNS exfiltration discovery, HTTP session triage, and NetFlow beacon hunting — all AI-assisted.

Authorized Use Only

All capture files are **synthetic**. Any IPs, domains, and ports shown are fictional. The detection patterns you learn here apply only to traffic you are authorized to capture and analyze.

Prerequisites

- Completed the [Theory](#) reading
- Access to an AI assistant (Claude, ChatGPT, or local LLM)
- Downloaded all four files from the [Theory page resources](#) into a folder named `traffic-lab/`
- Familiarity with: TCP/IP basics, DNS query types, HTTP methods, BPF filter syntax
- Optional: `tshark`, `scapy`, `jq` if you want to verify AI output against tooling

Lab Scenario

The AcmeCorp blue team has captured 10 minutes of network traffic from an internal subnet they suspect is compromised. They have shared the captures with you in text form. Your goal: confirm or deny the compromise, identify the C2 channel, and reconstruct the attacker's actions — all from the wire data alone.

Download Lab Files

Save all four files into a folder named `traffic-lab/` before starting the exercises.

↓ [tcpdump-capture.txt](#) ~9 KB

↓ [dns-queries.txt](#) ~6 KB

↓ [http-sessions.txt](#) ~5 KB

↓ [netflow-data.txt](#) ~7 KB

Exercise 1 — tcpdump Triage

Open [tcpdump-capture.txt](#). This is the human-readable output of `tcpdump -nn -tttt` covering several protocols.

Task 1.1 — First-pass protocol breakdown

Below is a tcpdump capture in human-readable format. Produce:

1. Protocol breakdown: count of packets per protocol (TCP, UDP, ICMP, ARP, DNS, HTTP, HTTPS, etc.).
2. Top 10 source-destination IP pairs by packet count.
3. List every unique destination port seen and identify the probable service (use IANA assignments + common red team ports).
4. Any "interesting" combinations:
 - Outbound traffic to unusual countries/ASNs (you can guess by IP)
 - Connections to non-standard high ports
 - ICMP volumes that suggest tunneling
 - ARP storms or spoofing patterns
5. The single most suspicious flow and why.

[paste tcpdump-capture.txt]

✓ **Self-check:** You should have a one-paragraph summary of "what is normal" and "what is not" within 5 minutes of analysis.

Task 1.2 — Build the BPF filters you would use next

Continuing in the same conversation, ask:

"Based on the suspicious flows above, write 5 BPF filters I would use against a live capture to (a) capture only that traffic, and (b) explicitly exclude noise. For each filter, explain in one sentence what it isolates."

This is how the AI converts analysis into action.

Exercise 2 — DNS Exfiltration Hunting

Open `dns-queries.txt`. This is a DNS resolver query log.

Task 2.1 — Find the tunnel

Below is a DNS query log. Identify:

1. Normal DNS activity baseline (top legitimate queried domains, typical query types, query rate).
2. Any DGA (domain generation algorithm) patterns — long, random, alphanumeric subdomains.
3. Any DNS tunneling indicators:
 - Subdomains that look like base32/base64 encoded data
 - High query rate to a single second-level domain
 - TXT, NULL, or unusual record types being queried
 - Very long query names (>50 chars)
4. Any DNS-over-HTTPS bypass attempts.
5. Internal hosts that are talking to suspicious external resolvers.
6. Decode any base32/base64 subdomain content if possible.

For each finding, give:

- Source IP
- Destination domain
- Query rate or count
- What the attacker is likely doing
- A defender detection rule (Sigma or Suricata syntax)

[paste dns-queries.txt]

✓ **Self-check:** If the data contains base64-encoded exfiltration, the AI should decode it for you. Verify the decoded content makes sense (it should look like commands or data, not random bytes).

Exercise 3 — HTTP Session Reconstruction

Open `http-sessions.txt`. This is a reconstruction of HTTP request/response pairs from the capture.

Task 3.1 — Find the credential leak and the C2 channel

Below is a series of HTTP request/response pairs. Analyze them as a red team operator. Identify:

1. Any credentials transmitted in cleartext (Basic auth, form POSTs, query parameters, custom headers).
2. Any session tokens or cookies that could be reused.
3. Any uploaded files (look for multipart/form-data, large POSTs).
4. Any "beacon" pattern: same client polling the same URL on a regular interval (e.g., every 60 seconds with the same POST body shape).
5. Any unusual HTTP methods (PUT, DELETE, PROPFIND, PATCH).
6. Any User-Agent values that suggest tooling (curl, python, Cobalt Strike, Sliver, Empire, custom strings).
7. Any data exfiltration indicators (large outbound POSTs to unusual destinations).

For the highest-value finding, write a one-paragraph "how to reproduce" that an attacker could use, and a Sigma detection rule a defender could deploy.

[paste http-sessions.txt]

✓ **Self-check:** HTTP captures usually contain at least one credential. You should have at least one usable credential or session token by the end of this exercise.

Exercise 4 — NetFlow Beacon Detection

Open `netflow-data.txt`. NetFlow does not show packet contents — only the metadata of every flow. Beaconsing C2 stands out clearly here.

Task 4.1 — Hunt the beacon

Below is NetFlow data (src_ip, dst_ip, src_port, dst_port, proto, bytes, packets, start, end). Analyze for:

1. Beaconsing: any (src,dst,port) tuple that has a regular interval between flows (e.g., one connection every 60 ± 5 seconds). Calculate the jitter percentage. Real malware uses jitter to evade detection — under 30% jitter is suspicious.
2. Long-lived connections that may indicate interactive C2 or reverse shells.
3. Asymmetric data transfer ratios (lots of bytes one direction, few the other).
4. "Low and slow" exfiltration: small flows summing to large total bytes over hours.
5. Connections to known bad ports (common C2: 4444, 8080, 8443, 1337, 31337, also legitimate-looking 443/80).
6. Internal lateral-movement flows (RDP/SMB/WinRM/SSH between hosts that should not be talking).

Output a ranked list of suspicious flows with:

- IP pair
- Port
- Periodicity / jitter
- Total bytes
- Verdict (likely C2, exfil, lateral, benign)

[paste netflow-data.txt]

✓ Self-check: You should be able to point to one specific beacon and say "this is the C2, the interval is X, the jitter is Y%."

Final Exercise — Build the Attack Story

Task 5.1 — Cross-source correlation

In a fresh AI conversation, paste the outputs of Exercises 1–4 and ask:

Here are four analyses of four traffic captures from the same 10-minute window. Reconstruct the full attack:

1. Initial access (where, when, how).
2. Command-and-control channel (protocol, beacon interval, C2 IP).
3. Lateral movement (which internal hosts, which protocol).
4. Data exfiltration (channel, volume, destination).
5. The compromised host(s) and the attacker IP(s).

Then produce TWO outputs:

- Red team perspective: "what to do next to maintain access without the patterns we just used"
- Blue team perspective: "5 detection rules that would have caught this earlier in the kill chain"

Format as markdown, max 1000 words.

✓ **Self-check: Your final report should connect at least 3 of the 4 sources. If it can't, go back and ask the AI which finding from each source supports the same conclusion — that's how you build evidence chains.**

Reflection

- Which traffic source gave the strongest single signal? Which was hardest to interpret?
- How did the AI handle base64/base32 decoding? Did it give you raw bytes when it should have decoded?
- What would change if you had a real binary `.pcap` file instead of a text dump? (Hint: think about TLS, encryption, and file size limits in the AI context window.)
- If you were the attacker, which of your activities was hardest to hide in this capture?

What's Next

Lesson 04 moves from observation to **code auditing** — using AI to read source code and find vulnerabilities at the application layer instead of the network layer.