

Lab Guide

Four exercises: Snort alert triage, pcap summary analysis, baseline-vs-anomaly comparison, and rule writing — turn network data into detections with AI.

Authorized Use Only

All pcap summaries and alerts are **synthetic**. Do not replay or resend any traffic from these files to real hosts.

Prerequisites

- Completed the [Theory](#) reading
- Access to an AI assistant (Claude, ChatGPT, or local LLM)
- Downloaded all four files into a folder named `nids-lab/`
- Familiarity with: TCP/IP, common ports, Snort/Suricata rule syntax, tshark output

Lab Scenario

AcmeCorp's Suricata sensor flagged a burst of alerts overnight. You have the alert dump, a tshark summary of the surrounding pcap, a known-good baseline of the network, and a list of anomaly traffic samples. Your job: separate true positives from noise, rebuild the attack path, and harden detection.

Download Lab Files

Save all 4 files into a folder named `nids-lab/` before starting the exercises.

↓ [anomaly-traffic.txt](#) ~1 KB

↓ [network-baseline.txt](#) ~1 KB

↓ [pcap-summary.txt](#) ~16 KB

↓ [snort-alerts.txt](#) ~14 KB

Exercise 1 — Snort Alert Triage

Open [snort-alerts.txt](#). NIDS output is 90% noise — you need to find the 10% that matters.

Task 1.1 — Separate signal from noise

Below are Snort/Suricata alerts. For each unique SID, determine:

1. Is this a known noisy rule (policy/info) or a high-fidelity attack signature?
2. Source / destination / port / protocol — do the directions make sense?
3. Is the destination an asset that SHOULD receive this traffic?
4. Cluster alerts by source IP and goal (scan / exploit / C2).
5. Flag any alert that correlates with known-bad IOCs (pastebin, Tor, known C2 ranges).
6. Rank by: confidence the alert is real x business impact.

Output:

- Top 5 true positives with severity and MITRE technique
- Noise list (rules to tune down)
- One rule to ADD based on coverage gaps

[paste snort-alerts.txt]

✓ **Self-check: Your top-5 list should have concrete IPs and a confidence rating, not just rule names.**

Exercise 2 — PCAP Summary Analysis

Open `pcap-summary.txt`. This is a tshark/zeek summary of the packet capture around the alerts.

Task 2.1 — Reconstruct the sessions

Below is a pcap summary (flows, HTTP requests, DNS queries, TLS handshakes). Analyze:

1. Top talkers (by bytes, by flow count).
2. Unusual flows: internal -> external to high ports, small heartbeat-sized flows, DNS to non-corporate resolvers.
3. HTTP findings: suspicious user-agents, beaconing intervals, POST to IP-literal URLs.
4. DNS findings: DGA-like names, long/random subdomains (exfil tunnel), TXT lookups, NXDOMAIN bursts.
5. TLS findings: self-signed certs, mismatched SNI vs. cert CN, JA3 fingerprints matching known tools.
6. Any plaintext protocols (Telnet, FTP, HTTP with creds).

For each finding map to MITRE ATT&CK (Command & Control, Exfiltration, or Discovery).

[paste pcap-summary.txt]

✓ **Self-check: You should find at least one C2 candidate with a specific beacon interval or DNS pattern.**

Exercise 3 — Baseline vs Anomaly

Open `network-baseline.txt` and `anomaly-traffic.txt`. The baseline is "normal". The anomaly file is the delta.

Task 3.1 — Statistical anomaly detection

I have a network baseline (normal traffic per hour) and an anomaly sample (current hour). Compare and identify:

1. Ports that appeared in anomaly but not baseline.
2. Destinations with >3x normal volume.
3. New external IPs/ASNs never seen before.
4. Protocols that shifted ratio (e.g., DNS jumped 10x).
5. Time-of-day violations (heavy traffic in off-hours).

Explain each delta: is it benign (backup job, patch Tuesday) or malicious (exfil, scan, C2)?

Output: a ranked anomaly list with confidence and suggested containment.

[baseline]

[paste network-baseline.txt]

[anomaly]

[paste anomaly-traffic.txt]

✓ **Self-check: At least one anomaly should be confidently attributed to a specific attacker behavior.**

Exercise 4 — Write the Detection Rule

Task 4.1 — Snort / Suricata rule authoring

Based on the attack path you reconstructed in exercises 1-3, write ONE high-fidelity Suricata rule that would detect this attack pattern in the future.

Requirements:

- Include: sid, rev, msg, classtype, priority, metadata (mitre_technique_id)
- Use flow, content matches, pcre only if necessary
- Avoid false positives on the baseline traffic
- Include a comment explaining the logic
- Estimate the FP rate per day on a 10k endpoint network

Then write a corresponding Sigma rule (source: network) so the logic can be ported to other NIDS/SIEMs.

✓ Self-check: Your rule should reference specific content bytes, not just "any any any". A weak rule is worse than no rule.

Final Exercise — Coverage Gap Analysis

Task 5.1 — What the NIDS missed

Given the attack you reconstructed:

1. Which stage of the kill chain was DETECTED by NIDS?
2. Which stage was MISSED?
3. What sensor / rule / log source would have caught the missed stage?
4. Write a prioritized coverage-improvement list (3 items).

Format as a 1-page detection engineering report.

✓ Self-check: You should be able to name at least one kill-chain stage that NIDS cannot see (e.g., encrypted C2, endpoint-local actions).

What's Next

Lesson 03 moves from network to **malware analysis** — sandbox reports, hashes, strings, and YARA.