Cross-Chapter PCAP Walkthrough: Fundamentals

2,958 words · ~13 min read 2026-05-07 v1

Course companion for: NET-101 (primary) + NET-201 + ADV-101 + RE-101 (secondary cross-references) **Scope:** pcap walkthrough; pairs with the curated catalog manifest **Pairs with manifest:** `pcap-tools/fixtures/curated-12-manifest.json` (entries `net-101-fund-*`) **Catalog surface:** `https://virtuscyberacademy.org/pcap-tools/` **Version:** 2026-05-07 v1

Overview

This handout is the companion walkthrough for six academy-original fundamentals PCAPs synthesized via scapy and added to the curated catalog under the NET-101 track. The pedagogical goal is **vocabulary-fluency for protocol byte-shape recognition before students touch malformed-record CVE captures**: a NET-101 student opening the canonical DNS query capture, walking the layer stack down from Ethernet to UDP to DNS, and naming the bytes at each layer is then prepared to read the Wireshark RCE Quartet's malformed TLS / SBC / RDP / Profile-import captures with the right vocabulary already installed. The handout assumes no prior packet-analysis experience; it does assume FND-102 Python fluency for students who want to extend the synthesis-via-scapy pattern beyond the academy-shipped corpus.

The six captures are academy-original, synthesized on the operator laptop via scapy 2.7.0 with explicit byte-shape control at every layer. License is `academy-original`. No live-capture noise; no PII risk; no third-party-traffic concerns. Each capture passed a tshark display-filter sanity check before commit (verification command set documented in §3 per capture). The captures pair with the catalog-page selector at `https://virtuscyberacademy.org/pcap-tools/` and render as click-to-load entries against the in-browser Wireshark / tshark Wasm workbench.

`--authorized-by` discipline applies even though these captures are academy-synthesized. The captures and the analysis discipline they teach extend to academy-owned, intentionally-vulnerable lab harnesses inside the `fwlab` container or equivalent.

Production network captures, third-party traffic, and any traffic carrying real-user PII are out of scope for the discipline this handout teaches; SOC analysts inheriting the academy's pattern apply it under their own organizational authorization.

---

§1: What this handout covers

Six academy-original fundamentals PCAPs covering the canonical traffic samples a NET-101 graduate must recognize at byte-shape register: DNS query/response; HTTP GET full cycle with TCP teardown; bare TCP three-way handshake; ICMP echo request and reply; ARP request and reply; DHCP four-step handshake. Each capture is small (under 2 KB), every byte is intentional, and the canonical Wireshark display filter that finds the distinctive packets is documented per capture.

The pedagogical thesis is that **vocabulary-fluency precedes anomaly recognition**. A student who has walked the layer stack of a known-clean DNS query has the byte-shape vocabulary to read a malformed DNS query as a deviation from the clean baseline; a student who has seen the canonical TCP three-way handshake recognizes a half-open scan or a SYN-flood as a deviation from the baseline. The Wireshark CVE Quartet's malformed-record captures (TLS dissector ECH integer truncation, SBC Codec frame-count overflow, RDP ZGFX missing bounds-check, Profile-import zip-slip) are best read after the baseline; this handout supplies the baseline.

The captures deliberately use IPv4 + classical protocols (DNS, HTTP/1.1 plain, TCP, ICMP, ARP, DHCP) rather than IPv6 + modern protocols (HTTP/2, HTTP/3, QUIC, TLS 1.3 + ECH, mDNS, SSDP). The classical-baseline scope reflects NET-101's pedagogical commitment to teach the substrate that every modern protocol layers on; IPv6 fundamentals, HTTP/2 fundamentals, and TLS 1.3 fundamentals are named as future-related topics in §6.3.

---

§2: The catalog at a glance

File	Protocols	Size	Canonical display filter	Pedagogical anchor
`fundamentals-dns-query.pcap`	DNS over UDP/53	225 B	`dns.qry.name == "example.com"`	Resolver request + response pair; DNS message format
`fundamentals-http-get.pcap`	HTTP/1.1 over TCP/80	958 B	`http.request.method == "GET"`	Full HTTP cycle with TCP three-way and FIN/ACK teardown
`fundamentals-tcp-3way.pcap`	TCP/22 (SSH port; no payload)	242 B	`tcp.flags.syn == 1`	TCP connection establishment in isolation
`fundamentals-icmp-ping.pcap`	ICMP echo	216 B	`icmp.type == 8` (request); `icmp.type == 0` (reply)	Simplest IP-layer probe
`fundamentals-arp-request-reply.pcap`	ARP on faux Ethernet	140 B	`arp.opcode == 1` (request); `arp.opcode == 2` (reply)	Layer-2 address resolution
`fundamentals-dhcp-handshake.pcap`	DHCP / BOOTP over UDP/67-68	1299 B	`dhcp.option.dhcp`	Four-step lease acquisition

Total corpus size: 3080 bytes across six captures. Cumulative is well under the 100 KB ceiling stated in the brief; each individual capture is under 2 KB.

The captures use `02:00:00:*` MAC addresses (locally-administered unicast range; no real-vendor collision) and `192.0.2.0/24` IPv4 addresses (TEST-NET-1; reserved for documentation per RFC 5737). The DNS query targets `example.com` (canonical IETF-blessed documentation domain) and the response carries the legitimate IPv4 address `93.184.216.34`. No real-user PII appears in any capture; no real-network traffic is recorded; no live-capture fingerprints are present.

---

§3: Per-capture walkthrough

§3.1 fundamentals-dns-query.pcap

§3.1.1 What the capture contains. Two packets: a single DNS A-record query for `example.com.` from a documentation client at `192.0.2.10` to a documentation resolver at `198.51.100.53`, and the matching A-record response. Transaction ID is `0x1234`; the response carries the legitimate `example.com` IPv4 address `93.184.216.34`.

§3.1.2 Why it matters. DNS is the first protocol most students learn end-to-end because it is small, common, and architecturally illustrative. A NET-101 graduate who walks this capture's two packets understands the DNS message format (transaction ID + flags + counts + question section + answer section), the request-response pairing pattern (matching transaction ID), and the canonical resolver behavior. The same byte shapes appear in every DNS capture the student will read for the rest of their career; this two-packet capture installs the vocabulary.

§3.1.3 Canonical display filter. `dns.qry.name == "example.com"` matches both packets (verified via `tshark -r fundamentals-dns-query.pcap -Y 'dns.qry.name == "example.com"' | wc -l` returning 2). The filter teaches Wireshark's protocol-field-comparison syntax; the same shape generalizes to any DNS field (`dns.flags.response`, `dns.qry.type`, `dns.resp.ttl`, `dns.a`).

§3.1.4 Layer-stack walkthrough. Ethernet header (14 bytes; locally-administered MACs at both ends) → IPv4 header (20 bytes; TEST-NET-1 source; 198.51.100.53 destination; protocol field `0x11` = UDP) → UDP header (8 bytes; ephemeral source port 51234; destination port 53) → DNS payload (query: 12-byte header + question section; response: 12-byte header + question section + answer section with 4-byte A-record rdata).

§3.1.5 What to look at next. `fundamentals-tcp-3way.pcap` for the connection-oriented analog (DNS over UDP is connectionless; HTTP over TCP requires a three-way handshake first). The Wireshark CVE Quartet's TLS dissector capture (`cve-2026-5402-tls-ech.pcap` per the parent quartet handout) reads against this baseline as a malformed-record extension of the same layer stack.

§3.2 fundamentals-http-get.pcap

§3.2.1 What the capture contains. Ten packets: the full HTTP/1.1 GET cycle from documentation client `192.0.2.10:49152` to documentation server `93.184.216.34:80`. The cycle covers the TCP three-way handshake (SYN, SYN-ACK, ACK), the GET request with canonical headers (`Host`, `User-Agent: virtus-academy-fundamentals/1.0`, `Accept`,

`Connection: close`), the server's ACK of the request, the 200 OK response with a small text body, the client's ACK of the response, the server's FIN/ACK, the client's FIN/ACK, and the server's final ACK. Sequence and acknowledgment numbers form a valid TCP exchange throughout.

§3.2.2 Why it matters. HTTP/1.1 is the protocol every web-related capture references implicitly; HTTP/2 + HTTP/3 + QUIC all build on the conceptual model HTTP/1.1 establishes (request + response + headers + body). A NET-101 graduate who walks this capture has the application-layer vocabulary (request line, status line, headers, body, content-length, connection control) plus the transport-layer vocabulary (sequence numbers, acknowledgments, window sizes, FIN/ACK teardown) installed. The walkthrough is also the academy's first exposure to **TCP sequence-number arithmetic** at depth: the SYN consumes one sequence number; subsequent ACKs name `prev_seq + payload_length`; the FIN consumes one sequence number on each side of the teardown.

§3.2.3 Canonical display filter. `http.request.method == "GET"` matches exactly one packet (the GET request itself). Complementary filters: `http.response.code == 200` (matches the response packet); `tcp.flags.fin == 1` (matches both FIN-bearing packets). The HTTP filters work because Wireshark's HTTP dissector parses the application layer; students should know that the dissector requires the underlying TCP stream to reassemble cleanly, which the canonical capture guarantees.

§3.2.4 Layer-stack walkthrough. Ethernet → IPv4 → TCP (with timestamp options omitted for clarity; window scaling option absent for the same reason) → HTTP. The HTTP layer carries CR-LF terminated lines and a blank line separating headers from body. A student inspecting the capture in Wireshark should expand the TCP header to see the SEQ / ACK arithmetic and expand the HTTP header to see the request method, URI, version, headers, and body.

§3.2.5 What to look at next. `fundamentals-tcp-3way.pcap` to see the three-way handshake in isolation (without HTTP application payload obscuring the TCP semantics). The cross-chapter NET-101 anchor reading guide (`cross-chapter-net-101-anchor-reading-guide.md`) carries Stevens *TCP/IP Illustrated Vol 1* Chapter 7 as the canonical TCP-three-way-handshake reading.

§3.3 fundamentals-tcp-3way.pcap

§3.3.1 What the capture contains. Three packets: SYN (client `192.0.2.10:49200` → server `192.0.2.20:22`), SYN-ACK (server → client), ACK (client → server). No application payload; no teardown; the connection is established and the capture ends. Initial client sequence number is `0xa1b2c3d4`; initial server sequence number is `0x55667788`; both increment by 1 on the SYN-and-ACK exchange.

§3.3.2 Why it matters. The TCP three-way handshake is the canonical connection-establishment pattern that every reliable-stream protocol on top of TCP inherits. Isolating the handshake (no payload) lets students see the TCP semantics in their pure form: SYN flag bits, MSS option negotiation in SYN and SYN-ACK, sequence-number selection at each end, the ACK number convention (`prev_peer_seq + 1`). The same shape recurs in every TCP capture; this three-packet capture installs the vocabulary cleanly.

§3.3.3 Canonical display filter. `tcp.flags.syn == 1` matches two packets (SYN and SYN-ACK; both have the SYN flag set). Complementary: `tcp.flags.ack == 1 and tcp.flags.syn == 0` matches the third packet (the bare ACK that completes the three-way). Students should know that Wireshark's TCP dissector exposes individual flag bits as separate filter fields, which lets the analyst combine flag matches with logical operators.

§3.3.4 Layer-stack walkthrough. Ethernet (14 bytes; client and server MACs in the documentation range) → IPv4 (20 bytes; TEST-NET-1 source and destination) → TCP (20-byte header + 4-byte MSS option in SYN and SYN-ACK; no options in the bare ACK). The TCP option field carries the Maximum Segment Size negotiation; both ends advertise 1460 bytes, which is the canonical Ethernet-over-IPv4 MSS (1500-byte MTU minus 20-byte IP header minus 20-byte TCP header).

§3.3.5 What to look at next. `fundamentals-http-get.pcap` for the same three-way handshake followed by HTTP application payload and FIN/ACK teardown. NET-101 Lab 4 reproduces this capture against a lab harness using `tcpdump -i lo port 22 -w lab-tcp.pcap` followed by an `nc -lp 22` and `nc 127.0.0.1 22` pair on the same host.

§3.4 fundamentals-icmp-ping.pcap

§3.4.1 What the capture contains. Two packets: an ICMP echo request (type 8) from `192.0.2.10` to `192.0.2.1`, and the matching echo reply (type 0). Identifier `0xbeef` and sequence number 1 are shared between the request and the reply, which is how ICMP

pairs request with reply. The payload `academy-fundamentals-icmp-ping-payload` is identical in both packets, mirroring the canonical ping behavior where the requester sends a payload and the responder echoes it back.

§3.4.2 Why it matters. ICMP is the simplest IP-layer probe and the canonical "is the host alive and routable" diagnostic. A NET-101 graduate walking this capture sees the IP layer in its purest form (no transport layer; ICMP rides directly on IP) and learns the request-response pairing pattern at a layer below TCP. The same byte shape appears in every traceroute capture, every ping diagnostic, and every ICMP-based covert-channel research write-up.

§3.4.3 Canonical display filter. `icmp.type == 8` matches the echo request; `icmp.type == 0` matches the echo reply. Complementary: `icmp.ident == 0xbeef` matches both packets via the shared identifier (which is how a pcap of mixed ping conversations distinguishes which reply pairs with which request). The filter teaches that ICMP types are numeric and well-known; the canonical Wireshark display shows the type symbolically (`Echo (ping) request`, `Echo (ping) reply`) but the underlying byte is the integer.

§3.4.4 Layer-stack walkthrough. Ethernet → IPv4 → ICMP. The ICMP header is 8 bytes (type, code, checksum, identifier, sequence) followed by the payload. There is no transport layer; the IPv4 protocol field carries `0x01` (ICMP) directly. The payload is arbitrary bytes that the responder echoes; this capture's payload is a deliberate ASCII string for legibility.

§3.4.5 What to look at next. `fundamentals-tcp-3way.pcap` for the contrast between ICMP's stateless request-reply and TCP's stateful connection establishment. NET-201 Module 6 covers ICMP-based diagnostics at deeper register including traceroute mechanics and the asymmetric-path detection patterns that operational network engineers use daily.

§3.5 fundamentals-arp-request-reply.pcap

§3.5.1 What the capture contains. Two packets at layer 2 (Ethernet) without IP layer: an ARP request from `02:00:00:00:00:01` (broadcasting "who has `192.0.2.1`?") and the ARP reply from `02:00:00:00:00:fe` (claiming the IP belongs to its own MAC). The reply unicasts back to the requester; the request broadcasts to `ff:ff:ff:ff:ff:ff`. ARP opcodes are `1` for request and `2` for reply.

§3.5.2 Why it matters. ARP is the protocol that binds layer-3 IPv4 addresses to layer-2 Ethernet MAC addresses. Every IPv4-over-Ethernet conversation requires an ARP exchange to resolve the destination MAC for the next-hop IP address; without ARP, the

packet has nowhere to go on the local segment. A NET-101 graduate who walks this capture understands the protocol's stateless-broadcast-and-cached-reply pattern and recognizes that the ARP cache (visible via `ip neighbor` on Linux or `arp -a` on Windows) is per-host state populated by exactly this protocol.

§3.5.3 Canonical display filter. `arp.opcode == 1` matches the request; `arp.opcode == 2` matches the reply. Complementary: `arp.src.hw_mac == 02:00:00:00:00:01` matches the request (filtered by source MAC); `arp.dst.proto_ipv4 == 192.0.2.1` matches the request (filtered by target IPv4). ARP attack analysis (cache poisoning, gratuitous ARP, ARP spoofing) builds on these primitives; PEN-101 Week 9's lateral-movement module exercises the cache-poisoning pattern against an academy lab harness under `--authorized-by` discipline.

§3.5.4 Layer-stack walkthrough. Ethernet (14 bytes; broadcast destination on the request, unicast destination on the reply; EtherType `0x0806` = ARP) → ARP (28 bytes for IPv4-over-Ethernet; hardware type, protocol type, hardware address length, protocol address length, opcode, sender hardware address, sender protocol address, target hardware address, target protocol address). There is no IP layer; ARP rides directly on Ethernet.

§3.5.5 What to look at next. `fundamentals-dhcp-handshake.pcap` for the layer-2-broadcast pattern at higher protocol depth (DHCP also broadcasts on the local segment, but at UDP/IP layer rather than at ARP layer). The cross-chapter `cross-chapter-cve-class-vocabulary-reference.md` does not currently cover ARP-cache-poisoning as a CVE class; the class is operationally important and earns a future related topic per §6.3.

§3.6 fundamentals-dhcp-handshake.pcap

§3.6.1 What the capture contains. Four packets covering the canonical DHCP four-step lease acquisition: DISCOVER (client `0.0.0.0` broadcasting from UDP/68 to UDP/67), OFFER (server `192.0.2.1` broadcasting back with `192.0.2.50` offered), REQUEST (client formally requesting the offered IP), ACK (server confirming the lease). Transaction ID `0x12345678` is shared across all four packets; the BOOTP layer carries the chaddr (client hardware address) and the DHCP options layer carries `message-type` plus the canonical lease parameters (server identifier, lease time, subnet mask, router).

§3.6.2 Why it matters. DHCP is the protocol that lets a host with no IP configuration (just a MAC address) discover an IP address, subnet mask, gateway, and DNS resolver from a server on the local segment. Every laptop, phone, and IoT device on every network the student will ever touch acquired its IP via DHCP. The four-step pattern

(DISCOVER, OFFER, REQUEST, ACK) is canonical; understanding it lets the analyst diagnose connectivity failures (no DHCP server present; server offering lease but client not requesting; ACK not arriving) at the protocol layer.

§3.6.3 Canonical display filter. `dhcp.option.dhcp` matches all four packets (each carries the DHCP message-type option). Complementary, per-step: `dhcp.option.dhcp_message_type == 1` (DISCOVER), `== 2` (OFFER), `== 3` (REQUEST), `== 5` (ACK). The numeric message-type values are part of the DHCP options registry; students should recognize the canonical four (1, 2, 3, 5) plus the operationally-relevant NAK (6) and RELEASE (7).

§3.6.4 Layer-stack walkthrough. Ethernet (broadcast destination on every packet because the client has no IP yet on DISCOVER and REQUEST; broadcast destination on OFFER and ACK because the broadcast-flag in BOOTP forces broadcast even when the server could unicast) → IPv4 (`0.0.0.0` source on client packets; `255.255.255.255` destination on every packet) → UDP (port 67 server-side; port 68 client-side) → BOOTP (236-byte fixed header carrying client hardware address) → DHCP (variable-length options field; `dhcp_message_type` is option 53; `server_id` is option 54; `lease_time` is option 51; `subnet_mask` is option 1; `router` is option 3; `end` is option 255).

§3.6.5 What to look at next. `fundamentals-dns-query.pcap` is what typically happens immediately after DHCP completes (the newly-configured client's first action is usually a DNS query). NET-201's network-architecture module covers DHCP relay agents, DHCP option 82, and DHCP-snooping defensive patterns at deeper register.

---

§4: Cross-track linkage

Course	Pickup module / week	Treatment of the fundamentals corpus
NET-101	Lab 4 (Wireshark capture and analysis); Lab 5 (Vulnerability identification at protocol layer)	Primary use; students walk every capture in Lab 4 with the catalog selector + the Wasm Wireshark workbench; Lab 5 contrasts the malformed-record CVE Quartet captures against these baselines
NET-201	Network architecture module; NSM-lite Suricata + Zeek authoring	Baseline-anchor for "what does normal look like" discussions; the captures appear in NSM-rule-tuning labs where false-positive avoidance against legitimate baseline traffic is the discipline
ADV-101	CVE-to-tool work where the target is a network protocol	The fundamentals corpus is the contrast set for the CVE captures; ADV-101 students reproduce the malformed-record patterns against academy lab harnesses and validate against the baseline via display-filter comparison
RE-101	Module 7 (CVE class reading list); SB6141 firmware-extraction context	The fundamentals corpus is the byte-shape reference for protocol parsers students reverse-engineer in extracted firmware; the SB6141 lab target's web-management UI captures (when synthesized for a future round) read against these baselines

The cross-track interleave reflects the academy's discipline that canonical baseline captures appear at multiple registers: NET-101 vocabulary, NET-201 NSM-rule-tuning register, ADV-101 contrast-set register, RE-101 byte-shape-reference register. The same six captures, four registers, four pedagogical purposes; the academy's coordinated curriculum makes the multi-register treatment tractable without each course curating its own baseline corpus.

---

§5: `--authorized-by` discipline reminder

These six PCAPs are academy-synthesized via scapy on the operator laptop. Every byte was produced under the academy's authorization and carries the `academy-original` license. No live network capture; no third-party traffic; no PII. The captures are safe to redistribute, embed in lab harnesses, and use as training material; the academy's catalog-page deployment serves them as click-to-load entries against the in-browser Wasm Wireshark workbench.

The discipline that travels with the captures is the same `--authorized-by` discipline that PEN-101 Lab 1 establishes and that ADV-101 capstone work enforces. Students learning to capture their own traffic via `tcpdump` or `wireshark` operate against academy-owned, intentionally-vulnerable lab harnesses inside the `fwlab` container or equivalent; production network captures, third-party traffic, and any traffic carrying real-user PII are out of scope for the discipline. Synthesizing additional fundamentals captures via scapy (the related topics in §6.3) is encouraged for cohorts that want to extend the corpus; the synthesis pattern established here generalizes across protocols, and the operator's laptop running scapy is the canonical academy-authorized synthesis surface.
