2단원. 패킷 분석 실습

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무(네트워크·클라우드·시스템·보안) 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

> **Q1. Wireshark와 tcpdump의 차이와 각각을 언제 사용하는지 설명하세요.** ★★★

답안 **tcpdump**는 CLI 기반의 경량 패킷 캡처 도구로, GUI가 없는 서버 환경(리눅스 프로덕션 서버 등)에서 원격 접속만으로 패킷을 캡처할 때 주로 사용하며 리소스 소모가 적습니다. **Wireshark**는 GUI 기반으로 프로토콜 디코딩, 컬러링, 스트림 재조합, 통계 분석 등 심층 분석 기능이 풍부해 캡처된 패킷을 사람이 보기 쉽게 분석하는 데 강점이 있습니다. 실무 흐름은 보통 서버에서 `tcpdump`로 `.pcap` 파일을 캡처한 뒤, 이를 로컬로 옮겨 **Wireshark**로 열어 상세 분석하는 방식이 일반적입니다. 두 도구 모두 내부적으로 **libpcap**을 사용하기 때문에 캡처 필터 문법이 유사합니다.

관련 개념 libpcap, pcap 파일, BPF(Berkeley Packet Filter), 원격 캡처, 프로토콜 디코딩

> **Q2. 캡처 필터(Capture Filter)와 디스플레이 필터(Display Filter)의 차이를 설명하세요.** ★★★

답안 캡처 필터는 패킷을 캡처하는 시점에 조건에 맞는 패킷만 저장하도록 걸러내는 필터로, tcpdump/Wireshark 모두 **BPF(Berkeley Packet Filter)** 문법을 사용

합니다(예: `tcp port 80`). 캡처 이후에는 필터에 걸리지 않은 패킷은 복구할 수 없다는 특징이 있습니다. 반면 디스플레이 필터는 이미 캡처된 전체 패킷 중 화면에 보여줄 패킷만 골라내는 필터로 Wireshark 고유의 문법을 사용하며(예: `http.request.method == "GET"`), 캡처된 데이터는 그대로 유지되고 필터를 자유롭게 바꿔가며 분석할 수 있습니다. 실무에서는 캡처 단계에서는 최소한의 조건(예: 특정 호스트/포트)으로 데이터량을 줄이고, 분석 단계에서 디스플레이 필터로 세밀하게 좁혀가는 방식을 씁니다.

관련 개념 BPF 문법, Wireshark 필터 문법, 캡처 후 손실 불가, 프로토콜 필드 필터링, 캡처 성능

> **Q3. TCP 3-way Handshake**와 4-**way Termination**을 패킷 관점에서 설명하세요. ★★★

답안 TCP 연결 수립은 **3-way Handshake**로 이루어집니다. 클라이언트가 **SYN**(시퀀스 번호 포함)을 보내면, 서버가 **SYN+ACK**로 응답하며 자신의 시퀀스 번호와 상대 시퀀스+1의 확인 번호를 담고, 클라이언트가 다시 **ACK**를 보내 연결이 확립됩니다. 연결 종료는 **4-way Termination**으로, 한쪽이 **FIN**을 보내면 상대가 **ACK**로 확인하고(이 시점 반이중 상태), 이후 상대도 자신의 **FIN**을 보내면 최초 종료를 요청한 쪽이 **ACK**로 마무리합니다. Wireshark에서는 `tcp.flags.syn==1`, `tcp.flags.fin==1` 같은 필터로 이 과정을 쉽게 추적할 수 있으며, 핸드셰이크가 완료되지 않고 SYN만 반복되면 방화벽 차단이나 서비스 다운을 의심할 수 있습니다.

관련 개념 SYN/ACK/FIN 플래그, 시퀀스/확인 번호, TIME_WAIT, half-close, 3-way/4-way

---

> **Q4. Wireshark**에서 **TCP** 재전송(**Retransmission**)이 관찰될 때 이를 어떻게 해석하고 원인을 진단하나요? ★★★

답안 Wireshark는 동일한 시퀀스 번호의 패킷이 다시 나타나면 **[TCP Retransmission]** 또는 **[TCP Spurious Retransmission]**로 자동 표시해줍니다. 재전송이 발생하는 근본 원인은 송신 측이 **ACK**를 제때 받지 못해 재전송 타이머(**RTO**)가 만료되었거나, 수신 측에서 중복 **ACK(Duplicate ACK)**를 3개 이상 받아 빠른 재전송이 트리거되었기 때문입니다. 실무에서는 재전송이 다수 관찰되면 회선 품질 저하(패킷 유실), 네트워크 혼잡, 방화벽/보안장비의 비정상 드롭, **MTU** 불일치등을 의심하고, `tcp.analysis.retransmission` 필터로 재전송 패킷만 걸러 발생 구간과 빈도를 확인합니다. 재전송이 지속되면 애플리케이션 응답 지연이나 타임아웃으로 이어지므로 서비스 장애의 주요 원인 중 하나로 다룹니다.

관련 개념 RTO(재전송 타임아웃), 중복 ACK, Fast Retransmit, 패킷 유실, tcp.analysis.retransmission

---

> **Q5. TCP** 윈도우 크기(**Window Size**)와 윈도우 축소(**Zero Window**)의 의미를 설명하세요. ★★

답안 **TCP** 윈도우 크기는 수신 측이 **ACK** 없이 한 번에 받을 수 있는 최대 데이터량을 송신 측에 알려주는 값으로, 흐름 제어(Flow Control)의 핵심 메커니즘입니다. 수신 측 애플리케이션이 데이터를 빠르게 처리하지 못해 수신 버퍼가 가득 차면 윈도우 크기를 점점 줄이다가 결국 **0(Zero Window)**을 보내 송신을 일시 중단시키며, 이후 버퍼에 여유가 생기면 **Window Update** 패킷으로 다시 알려줍니다. Wireshark에서 Zero Window가 자주 관찰되면 수신 서버의 처리 지연(**CPU/**애플리케이션 병목)이나 애플리케이션이 소켓 버퍼를 제때 비우지 못하는 문제를 의심할 수 있습니다. 대용량 전송 환경에서는 윈도우 스케일링(**Window Scaling**) 옵션으로 기본 16비트 한계(65535바이트)를 넘는 큰 윈도우를 사용할 수 있습니다.

관련 개념 흐름 제어, Zero Window, Window Update, 윈도우 스케일링, 수신 버퍼

> **Q6.** 패킷 분석에서 지연(**Latency**)의 원인을 어떻게 구분해서 진단하나요? ★★

답안 지연은 크게 네트워크 구간 지연과 서버(애플리케이션) 처리 지연으로 나눠 봐야 합니다. Wireshark에서 TCP 핸드셰이크의 SYN과 SYN+ACK 사이 시간 차는 순수 네트워크 왕복 지연(**RTT**)을 나타내고, 반면 요청 패킷(HTTP GET 등)과 응답 패킷 사이의 시간 차가 크면 서버 애플리케이션의 처리 지연을 의심합니다. Wireshark의 **Time** 컬럼 델타(**Delta time**)나 `tcp.time_delta` 필드를 활용하면 특정 구간의 응답 지연을 정량적으로 확인할 수 있습니다. 실무에서는 이 둘을 구분해야 네트워크 팀과 애플리케이션 팀 중 어느 쪽에서 원인을 찾아야 하는지 빠르게 판단할 수 있습니다.

관련 개념 RTT, tcp.time_delta, 서버 처리 지연 vs 네트워크 지연, Follow TCP Stream, 응답 시간 분석

> **Q7. DNS** 쿼리 응답 지연이나 실패를 패킷 캡처로 진단하는 방법을 설명하세요. ★★

답안 DNS 문제 진단 시에는 `udp.port==53` 또는 `dns` 디스플레이 필터로 쿼리와 응답 쌍을 확인합니다. 정상이라면 **DNS Query**에 대해 곧바로 **DNS Response**가 오지만, 응답이 없거나 지연되면 DNS 서버 장애, 방화벽 차단, 또는 재귀 질의 타임아웃을 의심할 수 있습니다. 응답 코드가 **NXDOMAIN**이면 존재하지 않는 도메인, **SERVFAIL**이면 DNS 서버 측 처리 오류를 의미합니다. 또한 응답 크기가 512바이트를 넘으면 UDP에서 TCP로 전환(**TCP Fallback**)되는 경우가 있어 방화벽에서 DNS TCP(53) 포트를 막아둔 경우 조회 실패의 원인이 되기도 합니다. 실무에서는 nslookup/dig 결과와 패킷 캡처를 함께 대조해 원인을 좁힙니다.

관련 개념 NXDOMAIN/SERVFAIL, DNS TCP Fallback, 재귀/반복 질의, dig/nslookup, DNS 캐시

> **Q8.** 스니핑 공격의 원리와 스위치 환경에서 패킷을 캡처하기 위한 방법(포트 미러링 등)을 설명하세요. ★★

답안 허브 환경에서는 모든 트래픽이 모든 포트로 브로드캐스트되어 스니핑이 쉽지만, 스위치 환경에서는 목적지 MAC 주소에 해당하는 포트로만 프레임을 전달하므로 다른 포트의 트래픽을 그냥 캡처할 수 없습니다. 이 때문에 공격자는 **ARP** 스푸핑이나 **MAC** 플러딩으로 스위치의 정상 동작을 방해해 트래픽을 가로채려 시도하기도 합니다. 반대로 정상적인 분석 목적으로는 스위치의 포트 미러링(**SPAN, Switched Port Analyzer)** 기능을 사용해 특정 포트나 VLAN의 트래픽을 분석용 포트로 복사해 캡처합니다. 실무 트러블슈팅이나 보안 모니터링(IDS 센서 연결 등)에서는 SPAN 포트나 네트워크 탭(**TAP**) 장비를 통해 정상적으로 트래픽을 확보하는 것이 표준적인 방법입니다.

관련 개념 SPAN(포트 미러링), 네트워크 TAP, MAC 플러딩, ARP 스푸핑, 스니핑 방어

> **Q9.** 이상 트래픽(비정상 트래픽) 탐지 시 패킷 캡처에서 주로 확인하는 지표들을 설명하세요. ★★

답안 이상 트래픽 탐지 시에는 몇 가지 패턴을 중점적으로 봅니다. 첫째, 짧은 시간 내 다수의 **SYN** 패킷만 있고 완료된 핸드셰이크가 적은 경우 SYN Flood 같은 서비스 거부 공격을 의심합니다. 둘째, 특정 목적지 포트에 대한 순차적인 접속 시도(포트 스캔)가 짧은 시간에 여러 포트로 발생하면 정찰 행위로 판단합니다. 셋째, 비정상적으로 큰 아웃바운드 트래픽이나 알려지지 않은 외부 **IP·**비표준 포트로의 지속적인 통신은 데이터 유출(Exfiltration)이나 C2(명령제어) 통신 가능성을 시사합니다. Wireshark의 통계(**Statistics**) 메뉴 - **Conversations, IO Graph** 기능을 활용하면 트래픽 양과 패턴을 시각적으로 파악해 이런 이상 징후를 빠르게 포착할 수 있습니다.

관련 개념 SYN Flood, 포트 스캔 패턴, C2 통신, IO Graph, Conversations 통계

> **Q10. Follow TCP Stream** 기능은 무엇이고 실무에서 언제 유용한가요? ★

답안 **Follow TCP Stream**은 Wireshark에서 특정 TCP 연결에 속한 모든 패킷의 페이로드를 시간 순서대로 재조합해 하나의 연속된 대화 형태로 보여주는 기능입니다. 개별 패킷 단위로는 파악하기 어려운 **HTTP** 요청/응답 전체 내용, 평문 프로토콜의 로그인 정보, 애플리케이션 레벨 오류 메시지 등을 한눈에 확인할 수 있어 장애 원인 분석이나 보안 점검에 유용합니다. 다만 TLS로 암호화된 트래픽은 세션 키 없이는 내용을 복호화해 볼 수 없으며, 이 경우 서버/클라이언트의 **SSL** 키 로그 파일을 Wireshark에 등록해야 복호화된 스트림을 확인할 수 있습니다. 실무에서는 평문 프로토콜(HTTP, FTP, Telnet 등) 트러블슈팅 시 이 기능을 자주 활용합니다.

관련 개념 TCP 스트림 재조합, TLS 복호화(SSL 키 로그), 평문 프로토콜, 페이로드 분석, HTTP 요청/응답 확인
