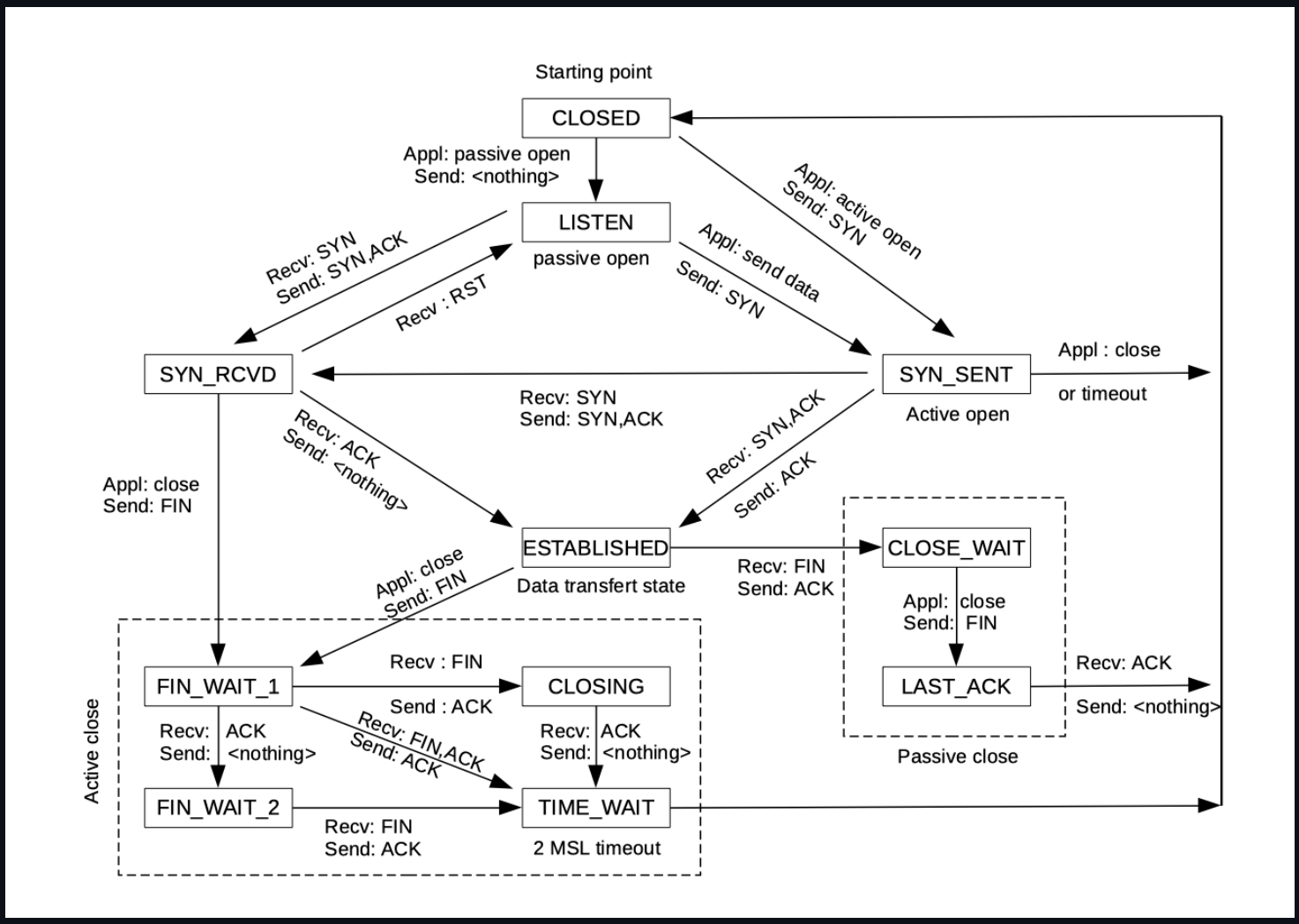
TCP/IP 관련 개념과 명령들

Flowchart of TCP



TCP 연결은 다음 상태중 하나이다.

- CLOSED : TCP 연결이 비활성화되어 데이터를 보낼 수 없음.
- LISTEN : TCP 서버는 클라이언트가 새 연결을 열기를 기다리고 있는 상태
- SYN_RECVD : 3-way 핸드셰이크에서 클라이언트로부터 첫 번째 TCP 메세지를 수신한 상태
- SYN_SENT : TCP 클라이언트가 3-way 핸드셰이크에서 첫 번째 메세지를 보냈다. 이 메세지에는 SYN 비트가 설정되어 있다.
- ESTABLISHED : 연결이 되어 데이터를 보내고 받을 수 있다.
- FIN_WAIT_1 : TCP 연결의 한 쪽에서 FIN 비트가 설정된 메세지를 전송하여 종료하고 연결의 반대 쪽에서 FIN을 기다리는 상태
- FIN_WAIT_2 : TCP 연결의 한 쪽에서 FIN을 보내고 반대 쪽에서부터 ACK를 받았다.
- CLOSE_WAIT : TCP 연결의 한 쪽에서 FIN 비트가 설정된 메세지를 수신하여 FIN을 승인하는 ACK 비트가 설정된 메세지를 보낸다.
- CLOSING : FIN_WAIT_1 상태에 있는 연결의 한쪽은 FIN 메세지를 수신하여 종료되고 해당 FIN에 대한 ACK를 보내고 반대 쪽에서 최종 ACK를 보낼 때까지 기다린다.

TCP 플래그 타입

- Synchronization (SYN) : 두 호스트간의 3-way 핸드셰이크 프로세스의 첫 번째 단계이다.
  오로지 송신자(sender)와 수신자(receiver)의 첫 번째 패킷만 이 플래그를 설정해야 한다.
  이것은 시퀀스 번호를 동기화하는 데 사용된다. 즉, 수락해야 하는 시퀀스 번호를 상대방에게 알리는 데 사용된다.
- Acknowledgement (ACK) : 호스트가 성공적으로 수신한 패킷을 승인하는 데 사용된다.
  ACK 필드에 유효한 ACK 번호가 포함되어 있으면 플래그가 설정된다.
  수신자는 연결 설정의 두 번째 단계에서 ACK = 1과 SYN = 1을 보내 송신자에게 초기 패킷을 수신했음을 알린다.
- Finish (FIN) : 연결 종료를 요청하는 데 사용된다. 즉, 송신자로부터 더 이상 받을 데이터가 없을 때 연결 종료를 요청한다.
  송신자가 보낸 마지막 패킷이다. 예약된 리소스를 해제하고 정상적으로 연결을 종료한다.
- Reset (RST) : RST 송신자가 해당 TCP 연결에 문제가 있거나 대화가 존재하지 않아야 한다고 느끼는 경우 연결을 종료하는 데 사용된다.
  패킷이 예상하지 못한 특정 호스트로 전송될 때 수신자 측에서 전송될 수 있다.
- Urgent (URG) : 패킷에 포함된 데이터에 우선 순위를 부여하고 수신자가 긴급하게 처리해야 함을 나타낸다.
  이 플래그는 긴급한 데이터가 패킷의 어디에 있는지 식별할 수 있게 Urgent Pointer field와 함께 사용된다.
  URG = 1 플래그가 있는 세그먼트 내부의 데이터는 응용 계층에 더 많은 데이터가 주어져도 즉시 응용 계층으로 전달된다.
- Push (PSH) : 전송 계층은 PSH = 1로 설정하고 응용 계층에서 시그널을 받는 즉시 세그먼트를 네트워크 계층으로 보낸다.
  수신자의 전송 계층은 PSH = 1을 확인하는 즉시 데이터를 응용 계층으로 전달한다.

일반적으로 수신자에게 이러한 패킷을 버퍼링(buffering) 없이 받는 즉시 처리하도록 지시

- Window (WND) : 수신 윈도우의 크기를 송신자에게 알리는 데 사용된다.
  윈도우 크기는 수신 호스트가 주어진 시간에 받아들일 수 있는 데이터의 양이다.
  송신자는 윈도우의 크기에 따라 보내는 데이터의 양을 제한해야 한다.
- Checksum (CHK) : 전송 중 TCP 세그먼트의 무결성을 확인하는데 사용된다.
  CHK는 헤더 및 데이터 필드를 포함하여 전체 세그먼트에 대해 계산되며 네트워크 경로를 따라 각 홉(Hop)에서 다시 계산된다.
- Sequence Number (SEQ) : 수신자가 패킷을 수신해야 하는 순서를 식별하기 위해 송신자가 각 세그먼트에 할당하는 고유 번호이다.
  시퀀스 번호는 승인 번호(ACK)와 함께 사용되어 안정적인 데이터 전송을 보장하고 중복 패킷을 방지한다.
- Acknowledgement Number (ACK) : TCP 세그먼트 수신을 확인하고 다음 예상 시퀀스 번호를 송신자에게 전달하는 데 사용된다.
  승인 번호 필드(Acknowledgement number field)에는 마지막으로 수신한 세그먼트의 번호가 아닌 다음 예상 세그먼트의 시퀀스 번호가 포함된다.

TCPDUMP 명령으로 패킷 캡쳐

```
tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]
[ -c count ] [ --count ] [ -C file_size ]
[ -E spi@ipaddr algo:secret,... ]
[ -F file ] [ -G rotate_seconds ] [ -i interface ]
[ --immediate-mode ] [ -j tstamp_type ] [ -m module ]
[ -M secret ] [ --number ] [ --print ]
[ --print-sampling nth ] [ -Q in|out|inout ] [ -r file ]
[ -s snaplen ] [ -T type ] [ --version ] [ -V file ]
[ -w file ] [ -W filecount ] [ -y datalinktype ]
[ -z postrotate-command ] [ -Z user ]
[ --time-stamp-precision=tstamp_precision ]
[ --micro ] [ --nano ]
[ expression ]
```

tcpdump는 부울(boolean) 식과 일치하는 네트워크 인터페이스의 패킷 내용을 인쇄한다.
-w 플래그로 패킷 데이터를 파일에 저장할 수 있고, -r 플래그로 네트워크 인터페이스 대신 저장된 패킷 파일에서 읽을 수도 있다.
tcpdump는 -c 플래그로 캡처할 패킷의 수를 지정하지 않으면 SIGINT 또는 SIGTERM 시그널에 의해 중단될 때까지 패킷 캡쳐를 계속한다.

tcpdump의 옵션들

옵션	*예제 / 설명*
-A	*각 패킷을 ASCII로 인쇄한다.*
-B buffer_size	*운영체제 캡쳐 버퍼 크기를 Kib(1024 바이트)단위로 buffer_size로 설정한다.*
-c count	*지정된 카운트 만큼 패킷을 수신하고 종료한다.*
--count	*패킷을 캡쳐한 패킷 파일로부터 읽을 때 출력할 패킷의 수를 지정한다.*
-C file_size	*raw packet을 파일에 저장하기 전에 지정된 파일 사이즈보다 큰지 확인하고 더 크면 현재 저장 파일을 닫고 새 파일을 연다. 첫 번째 저장파일은 -w 플래그로 지정된 이름을 가지며 그 다음 파일부터는 뒤에 1부터 시작하는 숫자가 붙는다. file_size의 기본 단위는 백만 바이트(1,048,576바이트가 아닌 1,000,000바이트)이다.*
-d	*컴파일된 패킷 일치 코드를 사람이 읽을 수 있는 형식으로 표준 출력에 덤프하고 중단한다. 코드 컴파일은 항상 DLT-specific 이라는 것을 명심하세요. 일반적으로 덤프에 사용할 DLT(Diagnostic Log and Trace)를 지정하는 것은 불가능하고 불필요하다. 왜냐하면 tcpdump는 -r로 지정한 입력 pcap 파일의 DLT, -i로 지정된 네트워크 인터페이스의 기본 DLT 또는 각각 -y 및 -i로 지정된 네트워크 인터페이스의 특정 DLT를 사용하기 때문이다.*
-dd	*패킷 일치 코드를 C프로그램 조각으로 덤프한다.*
-ddd	*패킷 일치 코드를 10진수로 덤프한다.*
-e	*각 덤프 라인에 링크 레벨(link-level) 헤더를 인쇄한다. 예를 들어 이더넷 및 IEEE 802.11과 같은 프로토콜의 MAC 계층 주소를 인쇄하는 데 사용할 수 있다.*
-E	*주소가 addr이고, SPI(Security Parameter Index) 값이 spi를 포함하는 패킷을 해독하려면 spi@ipaddr algo:secret 를 사용한다. 이 조합은 콤마(,) 또는 줄바꿈으로 구분해서 여러개 지정할 수 있다. 패킷을 해독한는 알고리즘은 des-cbc, 3des-cbc, blowfish-cbc, rc3-cbc, cast128-cbc이거나 없음(none)일 수 있다. 기본값은 des-cbc 패킷을 해독하는 기능은 tcpdump가 암호화가 활성화된 상태로 컴파일된 경우에만 존재 secret은 ESP 시크릿 키의 ASCII 텍스트이다.*
-i	*인터페이스를 지정하는 옵션이다. 생략하면 시스템 인터페이스 목록에서 가장 낮은 번호로 구성된 인터페이스(loopback 제외)를 검색하여 가장 먼저 일치하는 항목을 선택한다. 2.2 이상의 커널이 있는 Linux 시스템에서 "any" 인수값을 사용해 모든 인터페이스에서 패킷을 캡쳐할 수 있다.*
-n	*호스트 주소를 이름으로 변환하지 않는다.*
-nn	*프로토콜 및 포트 번호 등도 이름으로 변환하지 않는다.*
-r	*-w 옵션으로 생성된 패킷을 읽는다.*
-S	*상대적이 아닌 절대적인 TCP 시퀀스 번호를 인쇄한다.*
-s snaplen 또는 --snapshot-length=snaplen	*Snap length(캡쳐할 패킷의 크기)을 지정한다. (기본값은 262144 bytes) 제한된 스냅샷으로 인해 잘린 패킷은 "[|proto]"로 출력에 표시된다. 여기서 proto는 잘림이 발생항 프로토콜 레벨의 이름이다. 스냅샷 크기를 기본값 이하로 줄여야 할 때 관심 있는 프로토콜 정보를 캡쳐할 수 있는 가장 작은 숫자로 snaplen을 제한해야 한다. snaplen을 0으로 설정하면 tcpdump의 최신 이전 버전과의 호환성을 위해 기본값인 262144로 설정된다.*
-t	*각 덤프 라인에 타임스탬프를 인쇄하지 않는다.*
-tt	*각 덤프 라인에 형식화되지 않은(unformatted) 타임스탬프를 인쇄한다.*
-ttt	*각 덤프 라인의 현재 라인과 이전 라인 사이의 델타(micro-second resolution)를 인쇄한다.*

옵션	예제 / 설명
-tttt	각 덤프 라인의 날짜별로 진행되는 기본 포맷으로 타임스탬프를 인쇄한다.
-ttttt	각 덤프 라인의 현재 라인과 첫 번째 라인 사이의 델타(micro-second resolution)를 인쇄한다.
-v	구문 분석 및 인쇄 시 약간 더 장황한 출력 생성 예를 들어, TTL(Time to Live), ID(identification), 전체 길이, IP패킷의 옵션 등을 보여준다. 또한 IP 및 ICMP 헤더 체크섬 확인과 같은 추가 패킷 무결성 검사를 가능하게 한다. -w 옵션을 사용하여 파일에 쓸 때 10초마다 캡쳐된 패킷의 수를 보고한다.
-vv	훨씬 더 자세한 출력 생성 예를 들어 NFS 응답 패킷에서 추가 필드가 인쇄되고, SMB 패킷이 완전히 디코딩된다.
-vvv	훨씬 더 자세한 출력 생성 예를 들어 telnet SB ... SE 옵션은 전체적으로 인쇄된다. -X를 사용하면 telnet 옵션도 16진수로 인쇄된다.
-V file	파일에서 파일 이름 목록을 읽는다. 파일이 '-'이면 표준 입력이 사용된다.
-w file	원시 패킷(raw packets)을 구문 분석하고 인쇄하는 대신 파일에 쓴다. 나중에 -r 옵션으로 인쇄 가능하다. 파일이 '-'이면 표준 출력이 사용된다. 이 출력은 파일이나 파이프에 기록되면 버퍼링되므로 파일이나 파이프에서 읽는 프로그램은 패킷을 받은 후 임의의 시간 동안 패킷을 볼 수 없다. -U 플래그와 같이 쓰면 출력 버퍼가 채워질 때만 기록되는 것이 아니라 출력 파일에 기록된다.(패킷이 수신되는 즉시 기록)
-x	구문 분석 및 인쇄 시 각 패킷의 헤더를 인쇄하는 것 외에도 각 패킷의 데이터를 16진수로 인쇄한다. (link level의 헤더는 제외) 전체 패킷 또는 snaplen 바이트 중 더 작은 것이 인쇄된다. 이것은 전체 데이터 링크 계층(link-layer) 패킷이므로 패딩하는 링크 계층(예: Ethernet)의 경우 상위 계층의 패킷이 필요한 패딩보다 짧을 때 패딩 바이트도 안쇄된다.
-xx	구문 분석 및 인쇄 시 각 패킷의 헤더를 인쇄하는 것 외에도 link level의 헤더를 포함하여 각 패킷의 데이터를 16진수로 인쇄한다.
-X	구문 분석 및 인쇄 시 각 패킷의 헤더를 인쇄하는 것 외에도 각 패킷의 데이터를 16진수 및 ASCII로 인쇄한다. (link level의 헤더는 제외)
-XX	구문 분석 및 인쇄 시 각 패킷의 헤더를 인쇄하는 것 외에도 link level의 헤더를 포함하여 각 패킷의 데이터를 16진수 및 ASCII로 인쇄한다.
-z postrotate-command	-C 또는 -G 옵션과 함께 사용하면 tcpdump가 "postrotate-command file" 을 실행하게 된다. 여기서 file은 각 로테이션 후 닫히는 저장파일이다. 예를 들어 -z gzip 또는 -z bzip2를 지정하면 각 저장 파일이 압축된다. tcpdump는 캡처 프로세스를 방해하지 않도록 가장 낮은 우선 순위를 사용하여 캡처하면서 병렬로 다른 명령도 실행한다.
expression(표현식)	덤프할 패킷을 식으로 표현한다. 없으면 네트워크의 모든 패킷이 덤프된다. 표현식 구문은 pcap-filter 이고 식이 true인 패킷만 덤프된다.

tcpdump의 expression(패킷 필터 구문)

필터 표현식은 하나 이상의 프리미티브(primitives)로 구성된다. 프리미티브는 id(이름 또는 번호)로 구성되고 세 가지 종류가 있다.

- type : id 또는 이름이나 번호가 어떤 종류의 것을 가리키는지 알려준다. 가능한 type은 **host**, **net**, **port** and **portrange** 이다.
  예를 들어 `host foo', `net 128.3', `port 20', `portrange 6000-6008'
  지정하지 않으면 host가 기본값
- dir : dir은 특정 전송 방향을 지정한다. 가능한 dir은 **src**, **dst**, **src or dst**, **src and dst**, **ra**, **ta**, **addr1**, **addr2**, **addr3**, **addr4** 이다.
  예를 들어 `src foo`, `dst net 128.3`, `src or dst port ftp-data`
  지정하지 않으면 기본값은 src or dst 이다
- proto : proto는 패킷을 일치시킬 특정 프로토콜을 제한한다. 가능한 프로토콜은 **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **sctp**, **tcp**, **udp** 이다.
  proto를 지정하지 않으면 유형과 일치하는 모든 프로토콜을 매칭한다. 예를 들어, `src foo`는 `(ip6 or ip or arp or rarp) src foo` 를, `net bar`는 `(ip or arp or rarp) net bar`를, `port 53`은 `(tcp or udp or sctp) port 53`을 의미한다.

위의 항목 외에도 패턴을 따르지 않는 필터 표현식에 사용되는 몇가지 프리미티브가 있다.
gateway, **broadcast**, **less**, **greater** 및 같이 사용하는 산술표현식은 아래에 설명되어 있다.

보다 복잡한 필터 표현식은 단어 **and, or, not(각각 '&&', '||', '!' 와 같다)**을 사용하여 프리미티브를 결합하여 구성된다.
예를 들어 `**host** foo **and not port** ftp **and not port** ftp-data` 이런식으로 사용할 수 있다. 그리고 사용자의 입력을 줄이기 위해 동일한 한정자 리스트를 생략할 수도 있다.
예를 들어 `**tcp dst port** ftp **or** ftp-data **or** domain` 는 `**tcp dst port** ftp **or tcp dst port** tfp-data **or tcp dst port** domain`과 정확히 동일하다.

tcpdump의 허용되는 프리미티브(primitives)

프리미티브	설명
dst host hostnameaddr	IPv4/v6 패킷의 목적지(destination) 필드가 지정한 hostnameaddr(주소 또는 이름)과 일치할때 True
src host hostnameaddr	IPv4/v6 패킷의 출발지(source) 필드가 지정한 hostnameaddr(주소 또는 이름)과 일치할때 True
host hostnameaddr	IPv4/v6 패킷의 목적지 또는 출발지가 hostnameaddr과 일치할때 True
ether dst ethernameaddr	이더넷 목적지 주소가 지정한 ethernameaddr 이면 True ethernameaddr은 /etc/ethers의 이름이거나 다음과 같은 형식의 숫자 MAC 주소일 수 있다. 예1) XX:XX:XX:XX:XX:XX 예2) XX.XX.XX.XX.XX.XX 예3) XX-XX-XX-XX-XX-XX 예4) XXXXXXXXXXXX 여기서 각 X는 16진수(0-9,a-f,A-F로 이루어진 숫자)이고 ':', '.', '-' 의 다양한 혼합으로 사용가능하다
ether src ethernameaddr	이더넷 출발지 주소가 지정한 ethernameaddr 이면 True
ether host ethernameaddr	이더넷 출발지(source) 또는 목적지(destination) 주소가 지정한 ethernameaddr 이면 True
gateway host	패킷이 호스트를 게이트웨이로 사용한 경우 True 즉, 이더넷에서 source와 destination은 직접연결이라 host주소가 되지만 IP의 source와 destination은 직접연결이 아니라 둘

프리미티브	설명
	다 호스트가 아니다. 호스트는 이름이어야 하며, 머신의 호스트 이름에서 IP 주소로의 해석 메커니즘(host name file, DNS, NIS 등)과 머신의 호스트 이름에서 이더넷 주소로의 해석 메커니즘(/etc/ethers 등)으로 찾아야 한다. 문서내용 => Host must be a name and must be found both by the machine's host-name-to-IP-address resolution mechanisms (host name file, DNS, NIS, etc.) and by the machine's host-name-to-Ethernet-address resolution mechanism (/etc/ethers, etc.) 동등한 표현식은 다음과 같다. ether host *ethernameaddr* and not host *hostnameaddr*
dst net netnameaddr	패킷의 IPv4/v6 목적지 주소에 지정한 netnameaddr의 네크워크 번호가 있으면 True. netnameaddr은 네트워크 데이터베이스(/etc/networks 등)의 이름이거나 네트워크 번호일 수 있다.
src net netnameaddr	패킷의 IPv4/v6 출발지 주소에 지정한 netnameaddr의 네크워크 번호가 있으면 True.
net netnameaddr	패킷의 IPv4/v6 출발지 또는 목적지 주소에 지정한 netnameaddr의 네트워크 번호가 있으면 True.
net netaddr mask netmask	IPv4의 주소가 지정한 netaddr 주소와 일치시킬 때 지정한 넷마스크 netmask를 가지고 비교해서 일치하면 True.
dst port portnamenum	패킷이 IPv4/v6 TCP, UDP 또는 SCTP이고 목적지 포트번호가 지정한 portnamenum인 경우 True. portnamenum은 숫자이거나 /etc/services에서 사용되는 이름일 수 있다.
src port portnamenum	패킷의 출발지 포트가 지정한 portnamenum인 경우 True.
port portnamenum	패킷의 출발지 또는 목적지 포트가 지정한 portnamenum인 경우 True.
dst portrange portnamenum1-portnamenum2	패킷이 IPv4/v6 TCP, UDP 또는 SCTP이고 목적지 포트가 지정한 portnamenum1과 portnamenum2 사이에 있으면 True.
src portrange portnamenum1-portnamenum2	패킷의 출발지 포트가 지정한 portnamenum1과 portnamenum2 사이에 있으면 True.
portrange portnamenum1-portnamenum2	패킷의 출발지 또는 목적지 포트가 지정한 portnamenum1과 portnamenum2 사이에 있으면 True.
less length	패킷의 길이가 지정한 length보다 작거나 같으면 True.
greater length	패킷의 길이가 지정한 length보다 크거나 같으면 True.
위에 설명된 프리미티브 외 엄청 많은 프리미티브가 있다... 너무많아서 생략...	

tcpdump의 output 분석

```
00:00:04.611982 IP (tos 0x0, ttl 64, id 36884, offset 0, flags [DF], proto TCP(6), length 270)
    10.10.200.211.45074 > 12.123.11.223.6600: Flags [P.], cksum 0xed53 (incorrect -> 0x7c59), seq 3684485638:3684485822, ack 1769536644, win 14158, options[
        0x0000: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0010: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0020: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0030: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0040: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0050: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0060: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0070: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0080: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0090: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00a0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00b0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00c0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00d0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00e0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x00f0: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0100: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0110: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        0x0120: [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수] [4자리 16진수]
        ..............
```

TCP 프로토콜 라인의 일반적인 형식은 다음과 같다.

src > dst: Flags [tcpflags], seq data-seqno, ack ackno, win window, urg urgent, options [opts], length len

22:24:18.910372 IP (tos 0x10, ttl 64, id 9792, offset 0, flags [DF], proto TCP (6), length 88)
78.47.105.76.ssh > 82.132.219.219.55495: Flags [P.], cksum 0xcb29 (correct), seq 497880562:497880610(48), ack 1593322765, win 379, length 48

- 위 형식에서 src와 dst는 각각 출발지, 목적지 IP주소와 포트이다.
- tcpflags는 S(SYN), F(FIN), P(PSH), R(RST), U(URG), W(CWR), E(ECE) 또는 '.'(ACK) 의 조합이다. 플래그가 설정되지 않은 경우 'none'이다.
- data-seqno는 TCP 패킷의 시작 및 끝 시퀀스 번호이다. 전송되는 데이터의 양을 나타낸다.
예를 들어 seq 497880562:497880610 이면 시작과 끝의 차이는 48이고 전송되는 데이터의 Byte 이다. 이 값은 뒤에 length 필드와 일치해야 한다.
- Ackno는 이 커넥션에서 반대 방향의 예상되는 다음 데이터의 시퀀스 번호이다.
- window는 이 커넥션에서 반대 방향으로 사용 가능한 수신 버퍼 공간의 바이트 수이다. (즉 source 호스트의 window)
- opts는 TCP 옵션이다. (예를 들어 mss 1024 -> max-segment-size 가 1024 바이트)
- len은 페이로드(payload) 데이터의 길이이다.(헤더를 제외한 TCP 패킷 길이이고 Byte 단위이다.)
위 예시에서 모든 헤더를 포함한 IP 패킷의 길이가 88 Byte 이다. 이는 결합된 IP 헤더와 TCP 헤더의 합이 40 Byte 임을 뜻한다.

다음은 호스트 rtsg에서 호스트 csam으로의 rlogin 시작 부분이다

```
IP rtsg.1023 > csam.login: Flags [S], seq 768512:768512, win 4096, opts [mss 1024]
IP csam.login > rtsg.1023: Flags [S.], seq, 947648:947648, ack 768513, win 4096, opts [mss 1024]
IP rtsg.1023 > csam.login: Flags [.], ack 1, win 4096
IP rtsg.1023 > csam.login: Flags [P.], seq 1:2, ack 1, win 4096, length 1
IP csam.login > rtsg.1023: Flags [.], ack 2, win 4096
```

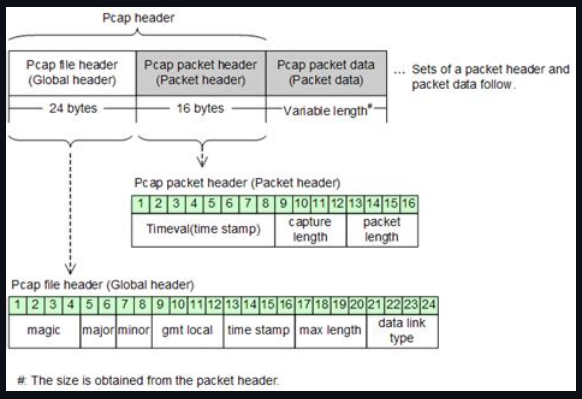
```
IP rtsg.1023 > csam.login: Flags [P.], seq 2:21, ack 1, win 4096, length 19
IP csam.login > rtsg.1023: Flags [P.], seq 1:2, ack 21, win 4077, length 1
IP csam.login > rtsg.1023: Flags [P.], seq 2:3, ack 21, win 4077, urg 1, length 1
IP csam.login > rtsg.1023: Flags [P.], seq 3:4, ack 21, win 4077, urg 1, length 1
```

- 첫 번째 라인은 rtsg의 TCP 포트 1023이 csam의 포트 login으로 패킷을 보냈다고 말한다. S는 SYN 플래그가 설정되었음을 나타내고, 패킷 시퀀스 번호는 168512이며 데이터가 없다.
(여기서 표기법 'first:last'는 첫 번째 부터 마지막까지 시퀀스 번호를 의미한다.)
piggy-backed ACK가 없고 사용 가능한 수신 window 크기는 4096 바이트였고 1024 바이트의 MSS를 요청하는 옵션이 있었다.
- csam은 rtsg의 SYN에 대한 piggy-backed ACK를 포함한다는 점을 제외하면 유사한 패킷으로 응답한다.
- 그런 다름 rtsg는 csam의 SYN을 ACK합니다. 여기서 '.'은 ACK 플래그가 설정되었음을 의미한다.
패킷에 데이터가 없으므로 데이터 시퀀스 번호 또는 length가 없다. 그리고 ACK 시퀀스 번호는 정수 '1' 이다.
tcpdump가 대화(conversation)를 처음 볼 때 패킷의 시퀀스 번호를 출력한다. 그리고 대화의 후속 패킷에는 현재 패킷의 시퀀스 번호와 맨처음 시퀀스 번호 간의 차이를 출력한다.
이 말은 첫 번째 이후의 시퀀스 번호들은 대화의 데이터 스트림에서 상대적인 바이트 위치로 해석될 수 있음을 의미한다.
(각 방향의 첫 번째 데이터 바이트는 1이다. => 호스트A, 호스트B가 TCP 통신을 할 때 A가 B에게 보내는 첫 번째 데이터, 반대로 B가 A에게 보내는 첫 번째 데이터도 1바이트이다. '-S' 옵션은 이 규칙을 무시하고 원래 시퀀스 번호가 출력되게 한다)

6번째 라인에서 rtsg는 csam에게 19 Byte의 데이터를 보낸다.(바이트 2부터 20까지 보낸다) 그리고 PSH 플래그가 설정되었다.
7번째 라인에서 csam은 rtsg가 전송한 데이터 21바이트를 포함하지 않고 2부터 20까지 받았다고 말한다.
수신한 데이터의 대부분은 분명히 소켓 버퍼에 있다. 그리고 19 Byte의 데이터를 받았기 때문에 수신 윈도우의 크기가 4096 Byte에서 4077 Byte로 19 Byte 만큼 작아졌다.
8번째, 9번째 라인에서 csam은 rtsg에게 긴급 푸시 데이터를 보낸다.

PCAP 파일 분석

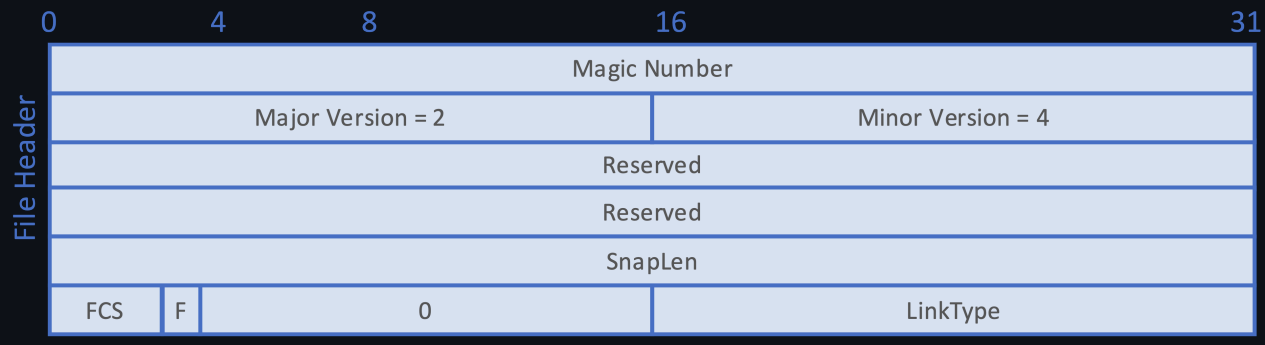
PCAP 파일의 구조



캡처 파일은 파일 헤더로 시작하여 0개 이상의 패킷 레코드(패킷당 하나씩)가 이어진다.

파일 헤더 및 패킷 레코드의 모든 필드는 항상 캡처 장비의 특성(little endian / big endian)에 따라 저장된다. 이것은 파일에 숫자로 저장되고 두 개 이상의 옥텟(octet = 8비트)에 걸쳐 있는 모든 필드를 나타낸다.
패킷은 전통적인 IETF(Internet Engineering Task Force) 다이어그램으로 표시되며 비트 단위로 왼쪽에서 오른쪽으로 번호가 매겨진다.
IETF 프로토콜은 전통적으로 빅 엔디안(big-endian) 네트워크 바이트 순서이므로 비트에 번호를 매기기(bit numbering)는 이진 값 위치를 반영하지 않습니다.
따라서 가장 중요한 비트는 파일이 빅 엔디안 시스템에 저장되는 것처럼 이 다이어그램의 왼쪽에 있다.

PCAP 파일 헤더



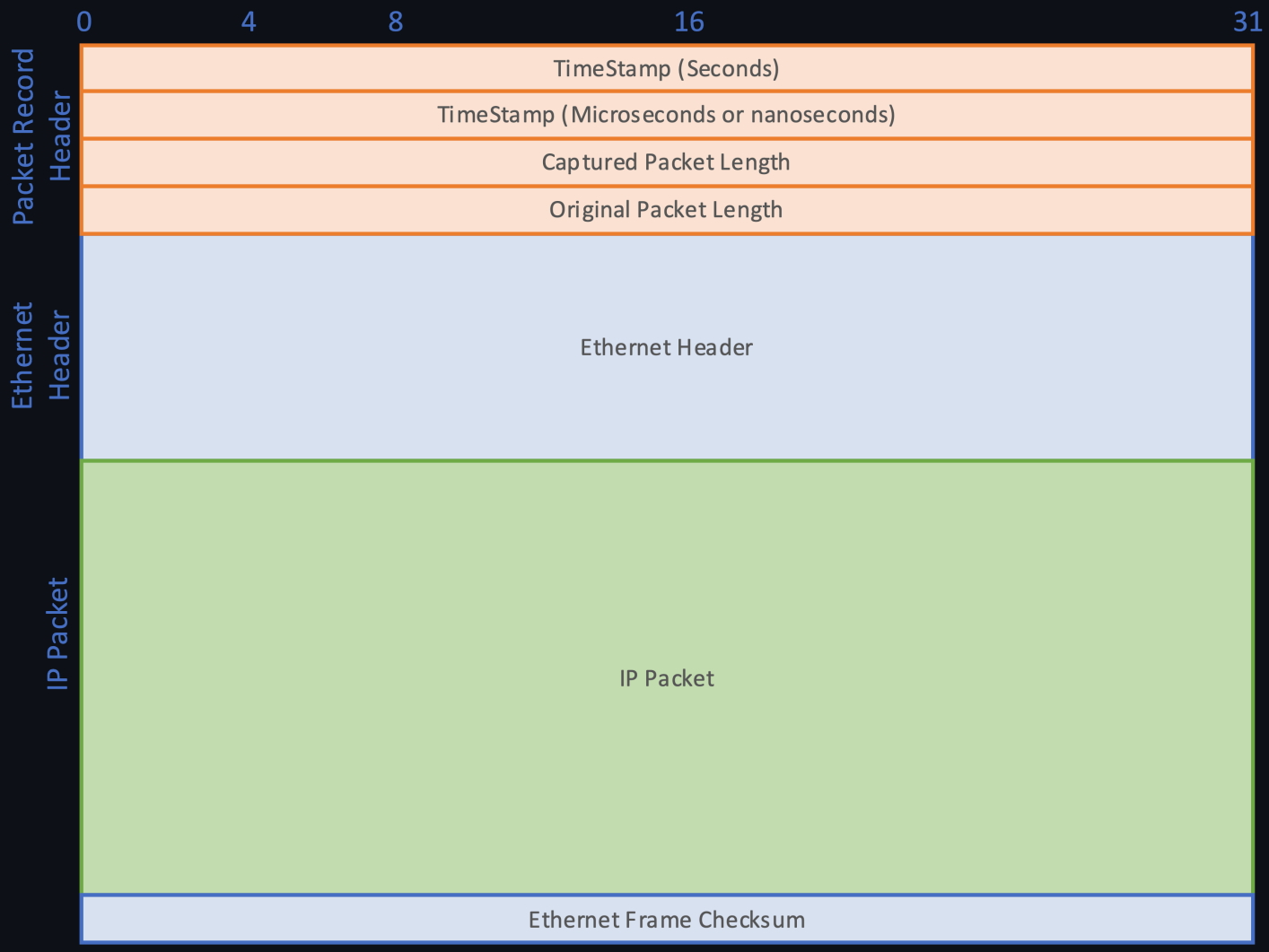
PCAP 파일 헤더 길이는 24 Byte 이다.
파일 헤더에 있는 각 필드의 의미는 다음과 같다.

- Magic Number (32 bits): 부호 없는(unsigned) 16진수이고 그 값은 0xA1B2C3D4 또는 0xA1B23C4D 이다.
  값이 0xA1B2C3D4 이면 패킷 레코드의 타임스탬프는 초 및 마이크로초(seconds and microseconds) 단위이고, 값이 0xA1B23C4D 이면 패킷 레코드의 타임스탬프는 초 및 나노초(seconds and nanoseconds) 단위이다.
  이 숫자를 사용하여 little-endian 시스템에 저장된 섹션과 big-endian 시스템에 저장된 섹션을 구분하고 경험적으로 pcap 파일을 식별할 수 있다.
- Major Version (16 bits): 현재 major version
- Minor Version (16 bits): 현재 minor version
- Reserved1 (32 bits): 현재 사용하지 않고 pcap 파일 작성자는 이 필드를 0으로 채워야 하며 pcap 읽는자는 이 부분은 무시해야 한다.
- Reserved2 (32 bits): 현재 사용하지 않고 pcap 파일 작성자는 이 필드를 0으로 채워야 하며 pcap 읽는자는 이 부분은 무시해야 한다.

- SnapLen (32 bits): 각 패킷에서 캡처된 옥텟(octets)의 최대값. 이 값을 초과하는 각 패킷 부분은 파일에 저장되지 않는다. 이 값은 당연히 0이 아니어야 하며 제한이 지정되지 않은 경우 가장 큰 패킷의 길이보다 크거나 같은 숫자여야 한다.
- LinkType (32 bits): 하위 28비트에서 파일에 있는 패킷의 링크 계층 유형을 정의하는 값이다. 예를 들어 값이 1이면 IEEE 802.3 이더넷이다.
- FCS and F : Frame Check Sequence 만약 'F' 비트가 설정되어 있으면(true), FCS 필드의 3비트는 각 패킷에 추가되는 FCS의 16비트(2바이트) 단어 수를 제공한다.

PCAP 파일 패킷 레코드

PCAP 파일에서 파일 헤더 뒤에는 일련의 패킷 레코드가 온다.
각 패킷 레코드는 타임스탬프, 패킷 길이 및 해당 패킷에서 캡처된 길이와 함께 네트워크에서 캡처된 싱글 패킷을 나타낸다.



패킷 헤더 길이는 16 Byte 이다.
패킷 레코드의 각 필드의 의미는 다음과 같다.

- Timestamp (Seconds), Timestamp (Microseconds or nanoseconds): 초 값은 1970-01-01 00:00:00 UTC 이후 몇 초가 지났는지 나타내는 32비트 부호 없는 정수이다. 그리고 마이크로초 또는 나노초 값은 앞서 말한 해당 시간 이후 경과된 마이크로초 또는 나노초의 수를 의미한다.
  값이 마이크로초를 나타내는지 나노초를 나타내는지는 파일 헤더의 매직 넘버로 지정된다.
- Captured Packet Length (32 bits): 패킷에서 캡처된 옥텟(octets)수. 패킷이 잘린 경우 이 값은 잘린 패킷의 길이이다.
- Original Packet Length (32 bits):
- 커밍순...
- 커밍순...