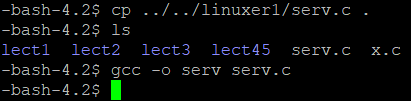
**HW Lect4, 5 sniffing**

**정보보호론 002분반**

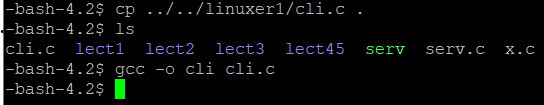
**정보통신공학과 12161719 김진호**

**[Lect4]**

**1) Copy serv.c.**

****

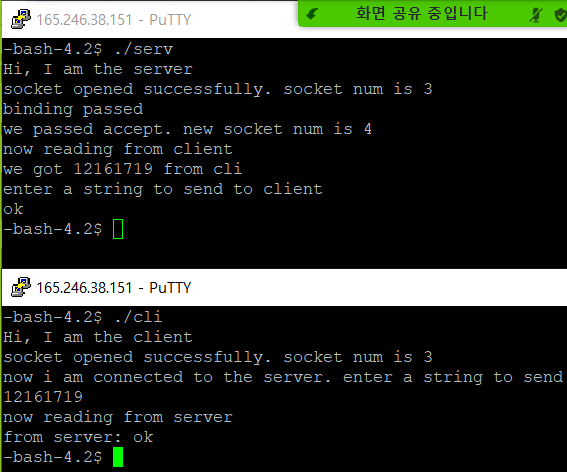
**2) Copy cli.c**

****

**3) Adjust port number both server and client. Recompile both and run the server first and run the client next. The client should talk first and then the server.**

****

서버와 클라이언트의 포트 번호를 위와 같이 통일하였다

****

포트 변경 후 다시 컴파일하여 서버와 클라이언트를 실행하고 통신한 결과는 위와 같다. 위의 사진 중 위 부분이 서버, 아래 부분이 클라이언트이며 서로에게 입력한 문자열을 잘 전송하고 있다.

**[Lect5]**

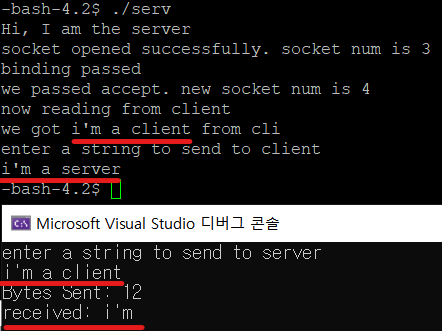
**Follow the steps below to display the contents of the packets between the server and the client. Submit explanation of each field in those packets.**

**1) Make a client in your PC as follows.**

****

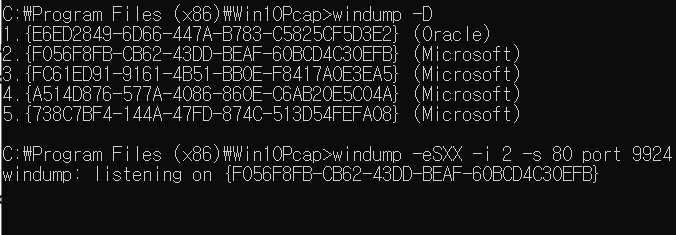
wincli.cpp를 강의 노트에서 복사해서 visual studio에서 작성하였다. 이 때 포트와 서버 주소를 위와 같이 조정하였다. (포트 번호는 1번 문제 이후로는 10000~60000 사이의 번호로 변경하여 사용하였다.)

매니페스트 포함 여부와 추가 종속성 설정 이후 putty에서 서버를, VS에서 클라이언트를 차례로 실행하였다.

****

클라이언트에서는 서버로 문자열이 온전히 잘 전송되는데, 서버에서 클라이언트로는 전송되는 문자열이 공백을 포함하지 못함을 확인할 수 있다. 이는 serv.c에서 문자열을 입력 받는 코드를 변경하면 손 쉽게 수정할 수 있다.

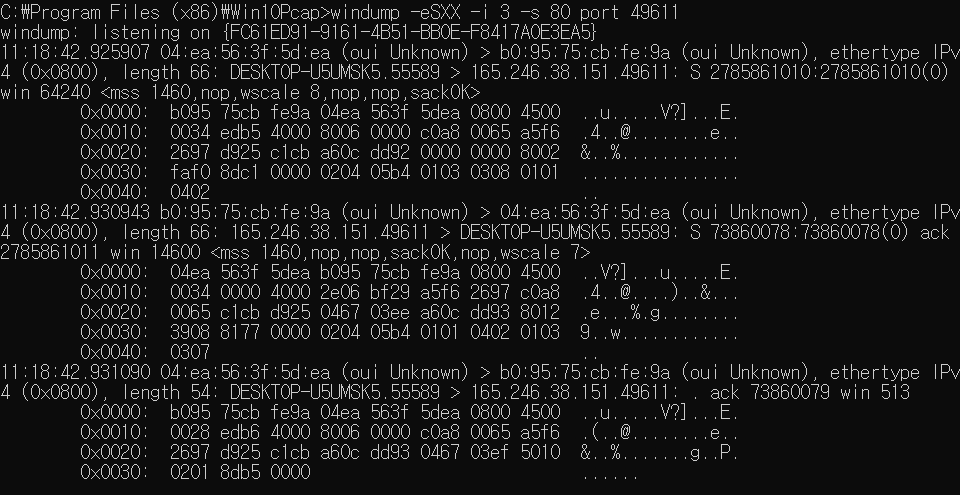
**2) Download win10pcap from iClass and install. Download windump.exe from iClass and run (in the command window; open command window as admin if needed) to monitor packets for specified port.**

****

windump가 위치한 폴더(디렉토리)에서 cmd를 실행한 후 window -D와 window -eSXX -i 2 -s 80 port 9924를 입력한 결과이다. window -D는 현재 사용 중인 네트워크 인터페이스를 출력하고, window -eSXX -i 2 -s 80 port OOOO는 특정 포트 OOOO에서의 데이터 패킷 정보를 출력하는 명령어이다.

i 뒤의 숫자는 인터페이스 번호를 의미하고, 80은 최대 몇 바이트까지 패킷 정보를 출력할 것인지를 지정한다.

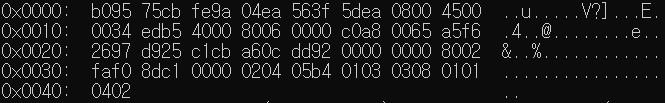
**3) Run your server again. Run the client in your PC.**

****

windump -eSXX -i 3 -s 80 port 49611을 cmd 창에 입력하고, 서버와 클라이언트를 실행한 결과 위와 같이 패킷 정보가 출력된 것을 확인할 수 있다. 이 때 인터페이스 3번을 이용하였다.

**4) Find the first packet which is a SYN packet sent by the client to the server in the windump window. Extract all packet header information. Refer TCP packet structure in Section 6 below.**

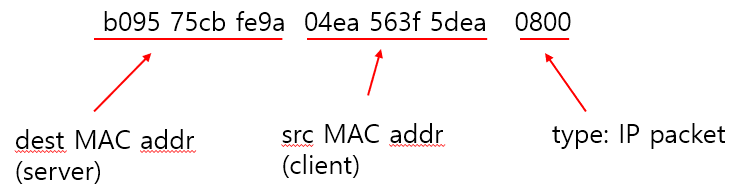
클라이언트와 서버를 실행시키면 가장 먼저, 클라이언트가 서버로 SYN 패킷을 보낸다. 따라서 windump에서 가장 먼저 출력된 데이터 패킷 정보가 SYN 패킷일 것이며, 이를 확인한 결과가 다음과 같다.



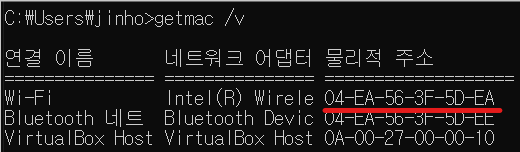
SYN 패킷에서는 Ethernet, IP, TCP 헤더 정보를 확인할 수 있는데 위 사진의 정보에서 각각의 의미는 다음과 같다.

**[Ethernet Header]**





첫 14 bytes가 Ethernet Header이다. 그 중에서도 첫 6 bytes는 destination의 MAC의 주소를, 7~12 bytes는 source의 MAC 주소를 의미하는데, SYN 패킷에서 source는 클라이언트, destination은 서버이다. 현재 클라이언트는 본인의 개인 컴퓨터로 실행 중이므로 실제 컴퓨터의 MAC 주소를 다음과 같이 확인했다.

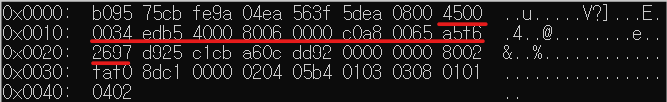


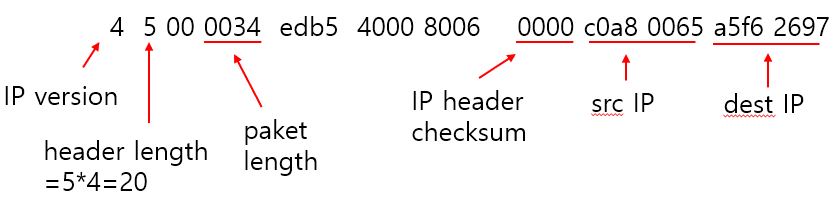
src MAC addr와 동일하다.

destination MAC addr은 현재 서버 컴퓨터의 MAC 주소로 예상된다.

그리고 마지막 13, 14번 바이트는 패킷의 타입이 IP 관련 패킷임을 알려주고 있다.

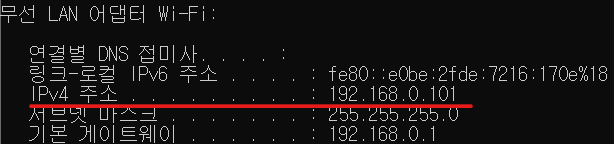
**[IP Header]**



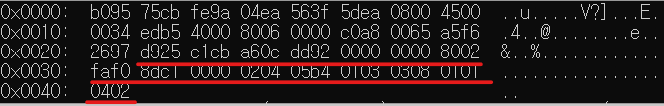


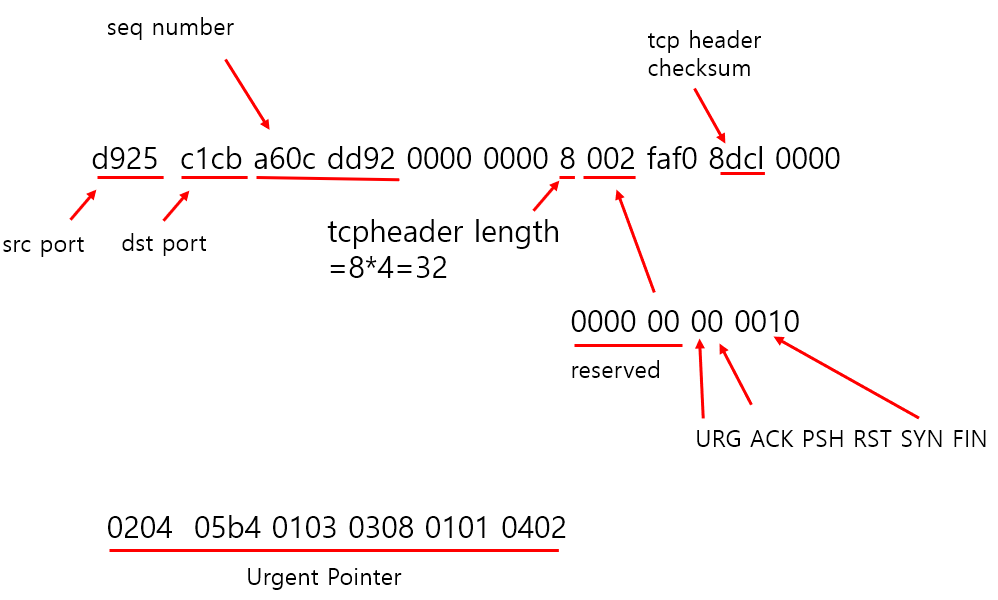
IP 헤더의 구성은 위와 같다. 가장 첫 바이트는 IP 버전이 4인지 6인지를 알려주고 그 다음으로 IP 헤더의 길이 정보를 담고 있다. 예를 들어 위에서는 헤더 길이로 5가 적혀있으며 이는 4를 곱하여 20 바이트로 해석할 수 있다. 그리고 IP헤더를 포함한 IP 패킷 전체 길이가 16\*3+4=52바이트임을 알 수 있다. 그리고 위의 사진에서 보이는 IP header checksum은 전송된 IP헤더에 오류가 있는지 파악할 때 쓰이는 지표이며, 가장 중요한 부분으로 마지막에는 source와 destination의 IP 정보가 존재한다. source의 IP 주소가 제대로 표현 되었는지 다음과 같이 확인하였다.

0xc0, 0xa8, 0x00, 0x65: 192. 168.0.101 이며 실제로 cmd에서 ipconfig를 통해 확인한 결과와 일치하였다.



**[TCP Header]**



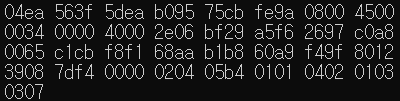


TCP 헤더의 해석은 위와 같다. source와 destination의 포트 번호 정보가 존재하고 seq number 즉, 이 tcp connection에서 현재까지 전송된 바이트의 수를 표시하고 있다. 이어서 “긴급 포인터가 있는지, 확인 응답 번호가 있는지, 데이터를 application layer로 즉각적으로 전달해야 하는지, 연결의 재설정이 필요한지, 연결 초기화를 위한 패킷인지, 데이터 전송이 종료인지”의 정보를 URG, ACK, PSH, RST, SYN, FIN의 플래그로 나타내고 있다. 그리고 이후의 정보들은 urgent pointer들의 위치이다.

**5) Analyze rest of the packets similarly.**

5번 숙제는 4번 숙제 이후 서버와 클라이언트를 다시 실행해서, TCP 헤더에서 확인할 수 있는 포트 번호가, 4번과 차이가 있음을 미리 알리는 바입니다.

**[SYN/ACK]**

****

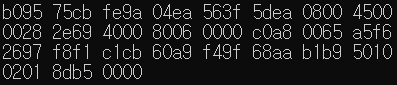
SYN 패킷의 Ethernet 헤더에서의 source와 destination MAC 주소가 뒤 바뀌었다. 이는 SYN 패킷에서는 클라이언트가 서버로 보내는 패킷이지만 SYN/ACK는 서버가 클라이언트에게 보내는 패킷이므로 자명하다. 그리고 여전히 패킷 타입은 0800으로 IP 패킷임을 알 수 있다.

4**5**00 **0034** 0000 4000 2e06 bf29 (a5f6 2697) (c0a8 0065) IP 헤더도 SYN 패킷에서의 내용과 본질적으로 다른 것은 없다. 여전히 IP 헤더의 길이가 4\*5=20 바이트이고 전체 IP 패킷 길이가 52바이트이며 단지 source와 destination의 IP 주소가 SYN 패킷과 비교해서 뒤바뀌었다는 것뿐이다.

c1cb f8f1 68aa bab8 60a9 f49f **8 012** 3908 7df4 0000 0204 05b4 0101 0402 0103 0307

TCP 헤더도 SYN 패킷의 그것과 큰 차이는 없다. TCP 헤더의 길이가 여전히 32바이트인 것이 동일한 점이며 0x012=0b000000 010010이므로 해당 패킷이 ACK인 동시에 SYN임을 알 수 있다.

**[final ACK]**

****

final ACK는 서버로부터 SYN/ACK 패킷을 받은 클라이언트가 다시 서버에게 보내는 최종 확인 패킷으로, Ethernet 헤더를 보면 SYN 패킷과 완전히 동일하다.

IP 헤더의 경우 IP 패킷 길이가 44로 줄은 것 말고는 source와 destination의 주소 구성도 SYN 패킷과 동일하다.

TCP 헤더는, f8f1 c1cb 60a9 f49f 68aa b1b0 **5 010** 0201 8db5 0000으로 이루어져 있다. 당연한 얘기이지만 현재 까지 전송된 바이트 수인 seq number는 변화하였고, TCP 헤더 길이도 4\*5=20바이트로 감소한 것을 확인할 수 있다. 0x010은 0b000000 010000이므로 final ACK의 패킷은 ACK 패킷임을 확인할 수 있다.

**6) Connect to www.inha.ac.kr and analyze SYN, S/ACK, ACK packets between the web browser and www.inha.ac.kr. You may need "-c num" option to capture the first num packets as below.**

windump –eSXX –c 20 –i 3 –s 80 host www.inha.ac.kr

위의 명령어를 입력하여 내 컴퓨터로 인하대학교 홈페이지 [www.inha.ac.kr](http://www.inha.ac.kr)에 접속할 때 송수신 되는 3way handshake 패킷을 확인하였다.

다만, 5번 문제 해결까지는 관찰하지 못했던 새로운 점이 하나 있었는데, 여러 개의 SYN, SYN/ACK, final ACK 패킷을 볼 수 있다는 것이었다. 이는 첫 SYN를 보내고 응답 패킷이 느린 경우에 이어서 SYN 패킷을 보내기 때문에 발생하는 현상이라고 한다. 물론 한번의 3way handshake가 완성되면 나머지 SYN 패킷과 그에 대한 응답들은 무시된다고 한다.