1. pcap-wireshark.c 코드 설명

pcap-wireshark-김유석.c 파일의 코드를 보자. Main함수의 가장 첫 부분에 다음과 같은 코드가 있다.

```
/* Retrieve the device list */
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
    fprintf(stderr,"Error in pcap_findalldevs: %s\n", errbuf);
    exit(1);
}

/* Print the dev list */
for(d=alldevs; d; d=d->next)
{
    printf("%d. %s", ++ndNum, d->name);
    if (d->description)
        printf(" (%s)\n", d->description);
    else
        printf(" (No description available)\n");

}

/* error ? */
if(ndNum==0)
{
    printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
    return -1;
}

/* select device for online packet capture application */
printf("Enter the interface number (1-%d):",ndNum);
scanf("%d", &devNum);
```

해당 코드에서는 먼저 pcap_findalldevs()함수를 통해 pcap_if_t 포인터 변수인 alldevs에 network interface list를 저장한다. 그리고 아래에서는 인터페이스 리스트를 출력하고, 사용자에게 사용하려는 interface를 번호로 입력하게 한다.

```
    /* Jump to the selected adapter */
    for(d=alldevs, i=0; i< devNum-1 ;d=d->next, i++);

    /* Open the adapter */
    if ( (adhandle= pcap_open_live(d->name,     // name of the device
                            SNAPLEN,            // captured packet size
                            1,                  // promiscous mode
                            1000,               // read timeout
                            errbuf              // error buffer
                            ) ) == NULL)
    {
        fprintf(stderr,"\nUnable to open the adapter. %s is not supported..\n",d->nam
e);
        /* Free the device list */
        pcap_freealldevs(alldevs);
        return -1;
    }

    printf("\nselected device %s is available\n\n", d->description);

    /* At this point, we don't need any more the device list. Free it */
    pcap_freealldevs(alldevs);
```

alldevs는 가장 첫 번째 인터페이스를 가리키는 포인터이므로 for문을 통해서 사용자가 입력한

인터페이스를 d에 할당하도록 한다. 이후 pcap_open_live()를 통해 어댑터를 열어준다. 마지막으로 이제 원하는 어댑터를 open하였으므로 device list는 더이상 필요하지 않다. 따라서 pcap_freealldevs()를 통해 free 시켜준다.

```
printf("Enter the file name for storing captured packets : ");
scanf("%s", fileName);
strcat(fileName, ".cap");

/* open file for storing captured packets */
pd = pcap_dump_open( adhandle, fileName);

/* start the capture */
pkNum=byteNum=ipNum=tcpNum=udpNum=0;
pcap_loop(adhandle, -1, cappkt_save, (u_char *)pd) ;

return 0;
```

이제 저장할 파일 이름을 사용자에게 입력하게 한다. pcap_dump_open을 통해 캡쳐 파일을 쓰게 된다. pcap_loop를 통해 위에서 입력한 어댑터로 패킷이 전송되면 cappkt_save함수가 콜백 함수로서 호출된다. 콜백 함수는 아래에서 설명하도록 하겠다.

```
// call back function
void cappkt_save (u_char *user, const struct pcap_pkthdr *h, const u_char *p);
```

위에서 설명하였듯이 이 함수는 패킷이 adhandle에 해당하는 어댑터로 전송되었을 때 실행되는 콜백 함수이다. 여기서는 캡쳐된 패킷의 헤더를 파악하여 어떤 패킷이 캡쳐되었는지 결과로 보여주기 위한 코드가 들어 있다.

```
// IP datagram
if ( (type=p[12]<<8 | p[13]) == 0x0800 ) {
    ipNum++;

    printf("(");
    // source IP address
    for(i = 0; i < 4; i++)
        printf("%02d%s",p[i+26], i==3?"->":".");

    // destination IP address
    for(i = 0; i < 4; i++)
        printf("%d%s",p[i+30], i==3?"):".");

    if(p[23] == IP_PROTO_UDP)
    {
        udpNum++;
    }
    if(p[23] == IP_PROTO_TCP)
    {
        tcpNum++;
    }

    // upper layer protocol
    printf(" Protocol : %s", p[23]==IP_PROTO_TCP? "TCP":
            p[23]==IP_PROTO_UDP? "UDP":"OTH");
} else
```

위의 코드는 캡쳐된 패킷이 IPv4이면 src, dst IP 주소를 출력하고, TCP, UDP인지 파악하여 해당 패킷이 어떤 프로토콜을 사용하였는지 출력해준다. 또한 헤더의 유무에 따라 개수를 count한다.

```
    printf(" clen=%d len=%d", h->caplen, h->len);

    // store captured packet with WinPcap packet header
    pcap_dump(user, h, p);
```

이후 pcap_dump함수를 통해 해당 패킷 정보를 파일에 쓴다.

```
// check termination
if ( cpkNum >= MAXPKT ) {
    printf("\n\n %d-packets were captured.\n", cpkNum);
    printf(" %d-ip datagrams were captured.\n", ipNum);
    printf(" %d-tcp packet were captured.\n", tcpNum);
    printf(" %d-udp packets were captured.\n", udpNum);
    /* close all devices and files */
    pcap_close(adhandle);
    pcap_dump_close(pd);
    exit(0);
}
```

MAXPKT은 DEFINE된 전처리문으로, 캡쳐할 패킷의 수이다. 만약 캡쳐한 패킷의 수가 MAXPKT과 같거나 더 많아지면 현재까지 캡쳐한 패킷의 수, ip, tcp, udp의 수를 출력한다. 그리고 pcap_close() 를 통해 해당 캡처 핸들을 닫아준다. 그리고 pcap_dump_close()를 통해 최종으로 파일을 닫고 저장하게 된다.

2. 동작 결과 설명

```
➜ NetSW11-expgm ./oncap_lab
1. en0 (No description available)
2. bridge0 (No description available)
3. p2p0 (No description available)
4. awdl0 (No description available)
5. utun0 (No description available)
6. vboxnet0 (No description available)
7. en1 (No description available)
8. utun1 (No description available)
9. en2 (No description available)
10. en4 (No description available)
11. lo0 (No description available)
12. gif0 (No description available)
13. stf0 (No description available)
Enter the interface number (1-13):1

selected device (null) is available

Enter the file name for storing captured packets : test
```

컴파일 후 실행시키면 가장 먼저 network interface 리스트를 출력하고, 캡쳐를 위한 인터페이스를 선택한다. 필자는 '1'을 선택하였다. 다음으로, 패킷 캡쳐 후 저장될 파일 명을 입력해야 한다. 필자는 'test'를 입력하였다. 이 파일은 test.cap으로 저장될 것이다.

```
No : 23 Time : 1.1930 (192.168.00.103->64.71.168.217) Protocol : TCP clen=66 len=66
No : 24 Time : 1.2964 (64.71.168.217->192.168.0.103) Protocol : TCP clen=66 len=66
No : 25 Time : 1.2977 (64.71.168.217->192.168.0.103) Protocol : TCP clen=68 len=216
No : 26 Time : 1.2978 (192.168.00.103->64.71.168.217) Protocol : TCP clen=66 len=66
No : 27 Time : 1.2979 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=117
No : 28 Time : 1.2986 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=119
No : 29 Time : 1.2986 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=122
No : 30 Time : 1.2986 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=108
No : 31 Time : 1.2988 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=1384
No : 32 Time : 1.2988 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=317
No : 33 Time : 1.7028 (64.71.168.217->192.168.0.103) Protocol : TCP clen=66 len=66
No : 34 Time : 1.7028 (64.71.168.217->192.168.0.103) Protocol : TCP clen=68 len=144
No : 35 Time : 1.7028 (64.71.168.217->192.168.0.103) Protocol : TCP clen=66 len=66
No : 36 Time : 1.7028 (64.71.168.217->192.168.0.103) Protocol : TCP clen=68 len=108
No : 37 Time : 1.7028 (64.71.168.217->192.168.0.103) Protocol : TCP clen=68 len=553
No : 38 Time : 1.7029 (192.168.00.103->64.71.168.217) Protocol : TCP clen=66 len=66
No : 39 Time : 1.7029 (192.168.00.103->64.71.168.217) Protocol : TCP clen=66 len=66
No : 40 Time : 1.7029 (192.168.00.103->64.71.168.217) Protocol : TCP clen=66 len=66
No : 41 Time : 1.7038 (192.168.00.103->64.71.168.217) Protocol : TCP clen=68 len=104
No : 42 Time : 1.7652 (192.168.00.103->222.239.118.8) Protocol : TCP clen=54 len=54
No : 43 Time : 1.7749 (222.239.118.08->192.168.0.103) Protocol : TCP clen=66 len=66
No : 44 Time : 1.8969 (192.168.00.103->216.58.197.138) Protocol : TCP clen=54 len=54
No : 45 Time : 1.9394 (216.58.197.138->192.168.0.103) Protocol : TCP clen=66 len=66
No : 46 Time : 2.0242 (64.71.168.217->192.168.0.103) Protocol : TCP clen=66 len=66
No : 47 Time : 5.7728 (192.168.00.103->192.168.0.1) Protocol : UDP clen=68 len=91
No : 48 Time : 5.7994 (192.168.00.01->192.168.0.103) Protocol : UDP clen=68 len=107
No : 49 Time : 5.8002 (192.168.00.103->17.248.153.170) Protocol : TCP clen=68 len=78

50-packets were captured.
50-ip datagrams were captured.
46-tcp packet were captured.
4-udp packets were captured.
```

파일 명을 입력하면 패킷 캡처가 시작되며 패킷 캡쳐가 시작된 시점을 기준으로 몇 번째 패킷인지, 몇 초 후의 패킷인지, src, dst IP주소, procotol, packet length 등을 보여준다. 패킷 캡처가 완료

되면 총 몇 개의 패킷이 캡쳐되었는지, 그 중 ip, tcp, udp는 각각 몇 개인지 출력하는 것을 확인할 수 있다. 다음으로, 캡쳐된 파일을 분석하는 코드를 실행시켜보았다.

```
➜  NetSW11-expgm ./capFileAnalysis test.cap
Frame 1:    54 bytes on wire,    54 bytes captured.
ETHERNET 2  src : f4:5c:89:b1:af:e7  dst : 90:8d:78:64:a5:0
Internet Protocol Version 4:  src : 192.168.0.103  dst : 58.123.220.114
Transmission Control Protocol: Src port : 64457  Dst port : 443

Frame 2:    54 bytes on wire,    54 bytes captured.
ETHERNET 2  src : f4:5c:89:b1:af:e7  dst : 90:8d:78:64:a5:0
Internet Protocol Version 4:  src : 192.168.0.103  dst : 58.123.220.114
Transmission Control Protocol: Src port : 64456  Dst port : 443

Frame 3:    66 bytes on wire,    66 bytes captured.
ETHERNET 2  src : 90:8d:78:64:a5:0  dst : f4:5c:89:b1:af:e7
Internet Protocol Version 4:  src : 58.123.220.114  dst : 192.168.0.103
Transmission Control Protocol: Src port : 443  Dst port : 64457

Frame 4:    66 bytes on wire,    66 bytes captured.
ETHERNET 2  src : 90:8d:78:64:a5:0  dst : f4:5c:89:b1:af:e7
Internet Protocol Version 4:  src : 58.123.220.114  dst : 192.168.0.103
Transmission Control Protocol: Src port : 443  Dst port : 64456
```

방금 저장한 test.cap 파일을 분석한 결과이다. 패킷마다 각 헤더 정보를 출력한다.

```
Frame 48:    91 bytes on wire,    68 bytes captured.
ETHERNET 2  src : f4:5c:89:b1:af:e7  dst : 90:8d:78:64:a5:0
Internet Protocol Version 4:  src : 192.168.0.103  dst : 192.168.0.1
User Datagram Protocol: Src port : 49197  Dst port : 53

Frame 49:   107 bytes on wire,    68 bytes captured.
ETHERNET 2  src : 90:8d:78:64:a5:0  dst : f4:5c:89:b1:af:e7
Internet Protocol Version 4:  src : 192.168.0.1  dst : 192.168.0.103
User Datagram Protocol: Src port : 53  Dst port : 49197

Frame 50:    78 bytes on wire,    68 bytes captured.
ETHERNET 2  src : f4:5c:89:b1:af:e7  dst : 90:8d:78:64:a5:0
Internet Protocol Version 4:  src : 192.168.0.103  dst : 17.248.153.170
Transmission Control Protocol: Src port : 64474  Dst port : 443

50-ip packets captured
4-udp packets captured
46-tcp packets captured
```

마지막 패킷까지 출력한 후 총 캡쳐된 패킷 수를 출력한다. 위의 캡처 프로그램과 같은 결과를 출력함을 확인할 수 있었다.