

PCAP Programming

| [19반] 정민석_7000

요약

과제의 목적은 PCAP API를 활용하여, 아래의 PACKET 정보를 출력하는 것에 목적을 둡니다.

- Ethernet Header [sizeof(struct ethheader)]
 - src MAC
 - dst MAC
- IP Header [ip->iph_ihl * 4]
 - src IP
 - dst IP
- TCP Header [TH_OFF(tcp) * 4]
 - src PORT
 - dst PORT

위의 정보를 출력하는 코드를 구현하였습니다. 특히 각 Header의 길이를 정확히 계산하여, 정상적인 HTTP Message를 출력하는 데에 성공했습니다.

코드 구현

먼저, ifconfig 명령어를 통하여, 활성화된 어댑터를 확인하였습니다.

활성화된 어댑터와 sniff_improved.c, myheader.h 코드를 참고하여 다음의 코드를 작성하였습니다.

```
#include <stdlib.h>
#include <stdio.h>
#include <pcap.h>
#include <arpa/inet.h>
#include <string.h>

/* Ethernet header */
struct ethheader {
    u_char ether_dhost[6]; /* destination host address */
    u_char ether_shost[6]; /* source host address */
    u_short ether_type;    /* protocol type (IP, ARP, RARP, etc) */
};

/* IP Header */
struct ipheader {
    unsigned char iph_ihl:4, //IP header length
                 iph_ver:4; //IP version
    unsigned char iph_tos; //Type of service
    unsigned short int iph_len; //IP Packet length (data + header)
    unsigned short int iph_ident; //Identification
    unsigned short int iph_flag:3, //Fragmentation flags
                    iph_offset:13; //Flags offset
```

```

    unsigned char    iph_ttl; //Time to Live
    unsigned char    iph_protocol; //Protocol type
    unsigned short int iph_chksm; //IP datagram checksum
    struct in_addr    iph_sourceip; //Source IP address
    struct in_addr    iph_destip; //Destination IP address
};

/* TCP Header */
struct tcpheader {
    u_short tcp_sport;        /* source port */
    u_short tcp_dport;        /* destination port */
    u_int    tcp_seq;         /* sequence number */
    u_int    tcp_ack;         /* acknowledgement number */
    u_char    tcp_offx2;      /* data offset, rsvd */
#define TH_OFF(th)      (((th)→tcp_offx2 & 0xf0) >> 4)
    u_char    tcp_flags;
#define TH_FIN 0x01
#define TH_SYN 0x02
#define TH_RST 0x04
#define TH_PUSH 0x08
#define TH_ACK 0x10
#define TH_URG 0x20
#define TH_ECE 0x40
#define TH_CWR 0x80
#define TH_FLAGS (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
    u_short tcp_win;          /* window */
    u_short tcp_sum;          /* checksum */
    u_short tcp_urp;          /* urgent pointer */
};

void print_tcp_header(struct tcpheader *tcp) {
    printf("\nTCP Header\n");
    printf("  PORT From: %d\n", ntohs(tcp→tcp_sport));
    printf("  PORT To: %d\n", ntohs(tcp→tcp_dport));
    printf("  Sequence: %u\n", ntohl(tcp→tcp_seq));
    printf("Acknowledgment: %u\n", ntohl(tcp→tcp_ack));
    printf("  Data offset: %d\n", TH_OFF(tcp));
    printf("  Flags: %d\n", tcp→tcp_flags);
    printf("  Window size: %d\n", ntohs(tcp→tcp_win));
    printf("  Checksum: %d\n", ntohs(tcp→tcp_sum));
    printf("  Urgent pointer: %d\n", ntohs(tcp→tcp_urp));
}

void print_application_data(const u_char *data) {
    printf("Message: ");
    for (int i = 0; i < 100; i++) {
        if (data[i] == '\0') break;
        printf("%c", data[i]);
    }
    printf("\n");
}

void got_packet(u_char *args, const struct pcap_pkthdr *header,
    const u_char *packet)
{

```

```

struct ethheader *eth = (struct ethheader *)packet;

printf("\nEthernet Header\n");
printf("    From: %02x:%02x:%02x:%02x:%02x:%02x\n",
    eth->ether_shost[0],
    eth->ether_shost[1],
    eth->ether_shost[2],
    eth->ether_shost[3],
    eth->ether_shost[4],
    eth->ether_shost[5]);
printf("    To: %02x:%02x:%02x:%02x:%02x:%02x\n",
    eth->ether_dhost[0],
    eth->ether_dhost[1],
    eth->ether_dhost[2],
    eth->ether_dhost[3],
    eth->ether_dhost[4],
    eth->ether_dhost[5]);

if (ntohs(eth->ether_type) == 0x0800) { // 0x0800 is IP type
    struct ipheader *ip = (struct ipheader *) (packet + sizeof(struct ethheader));
    int ip_header_length = ip->iph_ihl * 4;

    printf("\nIP Header\n");
    printf("    From: %s\n", inet_ntoa(ip->iph_sourceip));
    printf("    To: %s\n", inet_ntoa(ip->iph_destip));

    switch(ip->iph_protocol) {
        case IPPROTO_TCP: {
            printf("    Protocol: TCP\n");
            struct tcpheader *tcp = (struct tcpheader *) (packet + sizeof(struct ethheader) + ip_header_length);
            print_tcp_header(tcp);

            int tcp_header_length = TH_OFF(tcp) * 4;
            int total_headers_size = sizeof(struct ethheader) + ip_header_length + tcp_header_length;
            int data_length = ntohs(ip->iph_len) - ip_header_length - tcp_header_length;

            if (data_length > 0) {
                const u_char *data = packet + total_headers_size;
                // 널 종료를 위한 버퍼 할당
                char *http_message = malloc(data_length + 1);
                if (http_message) {
                    memcpy(http_message, data, data_length);
                    http_message[data_length] = '\0'; // 널 종료
                    printf("HTTP message: %s\n", http_message);
                    free(http_message);
                }
            }
            break;
        }
        case IPPROTO_UDP:
            printf("    Protocol: UDP\n");
            break;
        case IPPROTO_ICMP:
            printf("    Protocol: ICMP\n");
            break;
    }
}

```

```

        default:
            printf(" Protocol: others\n");
            break;
        }
    }
}

int main()
{
    pcap_t *handle;
    char errbuf[PCAP_ERRBUF_SIZE];
    struct bpf_program fp;
    char filter_exp[] = "tcp";
    bpf_u_int32 net;

    // Step 1: Open live pcap session on NIC with name enp0s3
    handle = pcap_open_live("en0", BUFSIZ, 1, 1000, errbuf);

    // Step 2: Compile filter_exp into BPF psuedo-code
    pcap_compile(handle, &fp, filter_exp, 0, net);
    if (pcap_setfilter(handle, &fp) != 0) {
        pcap_perror(handle, "Error:");
        exit(EXIT_FAILURE);
    }

    // Step 3: Capture packets
    pcap_loop(handle, -1, got_packet, NULL);

    pcap_close(handle); //Close the handle
    return 0;
}

```

먼저 packet에서 Ethernet Header까지의 address를 잘라서, Ethernet Header의 정보를 출력하였습니다.

다음으로 packet의 주소에 Ethernet Header의 길이를 더하면 IP Header의 시작 address가 나옵니다. 마찬가지로 방법으로 IP Header의 정보를 출력하였습니다.

마찬가지로 packet의 주소에 Ethernet Header, IP Header의 길이를 더하면 TCP Header의 시작 address가 나온다는 점을 통하여 TCP Header의 출력하였습니다.

마지막으로 packet의 주소에 Ethernet Header, IP Header, TCP Header 길이를 모두 더하면, 5계층의 Data의 시작 주소가 도출된다는 점을 통하여, HTTP 메시지를 출력하였습니다.

각 헤더의 길이는 다음과 같습니다.

- Ethernet Header: sizeof(struct ethheader)
- IP Header: ip->iph_ihl * 4
- TCP Header: TH_OFF(tcp) * 4

실습

다음의 명령어로 컴파일을 진행하였습니다.

```
$ gcc assignment.c -lpcap
```

코드를 실행하는 동안 브라우저 혹은 아래의 명령어를 통하여 tcp 요청을 전송하였습니다.

```
$ sudo nping --tcp -p 80 --count 5 example.com
```

or

```
$ curl http://example.com
```

결과

실행 결과는 다음과 같습니다. 각 헤더의 정보와 HTTP 메시지를 출력하는 것에 성공하였습니다.

Ethernet Header

From: 58:86:94:84:8e:00

To: 02:93:27:6b:b5:22

IP Header

From: 96.7.128.198

To: 192.168.0.22

Protocol: TCP

TCP Header

PORT From: 80

PORT To: 60021

Sequence: 3540845862

Acknowledgment: 730046605

Data offset: 8

Flags: 16

Window size: 509

Checksum: 4981

Urgent pointer: 0

HTTP message: HTTP/1.1 200 OK

Content-Type: text/html

ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"

Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT

Cache-Control: max-age=363

Date: Mon, 24 Mar 2025 15:30:35 GMT

Content-Length: 1256

Connection: keep-alive

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<title>Example Domain</title>
```

```
<meta charset="utf-8" />
```

```
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

```
<style type="text/css">
```

```
body {
```

```
background-color: #f0f0f2;
```

```
margin: 0;
```

```
padding: 0;
```

```
font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", H
```

```

}
div {
  width: 600px;
  margin: 5em auto;
  padding: 2em;
  background-color: #fdfdff;
  border-radius: 0.5em;
  box-shadow: 2px 3px 7px 2px rgba(

```

```

Ethernet Header
  From: 58:86:94:84:8e:00
  To: 02:93:27:6b:b5:22

IP Header
  From: 96.7.128.198
  To: 192.168.0.22
  Protocol: TCP

TCP Header
  PORT From: 80
  PORT To: 60021
  Sequence: 3540845862
  Acknowledgment: 730046605
  Data offset: 8
  Flags: 16
  Window size: 509
  Checksum: 4981
  Urgent pointer: 0
HTTP message: HTTP/1.1 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=363
Date: Mon, 24 Mar 2025 15:30:35 GMT
Content-Length: 1256
Connection: keep-alive

<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI",
    }
  }
  div {
    width: 600px;
    margin: 5em auto;
    padding: 2em;
    background-color: #fdfdff;
    border-radius: 0.5em;
    box-shadow: 2px 3px 7px 2px rgba(

```