**[3차 과제] Netfilter, Hooking**

**1.**

이세령(2014120016), 오형준(2014190004) / 제출일자: 12월 23일 / 프리데이 사용일수 : 3

**2. 역할**

이세령 : 네트워크 코드 분석, Netfilter 개념 이해

오형준: Module내 코딩 및 라우터 설정, Netfilter 개념 작성

**3. 개발환경**

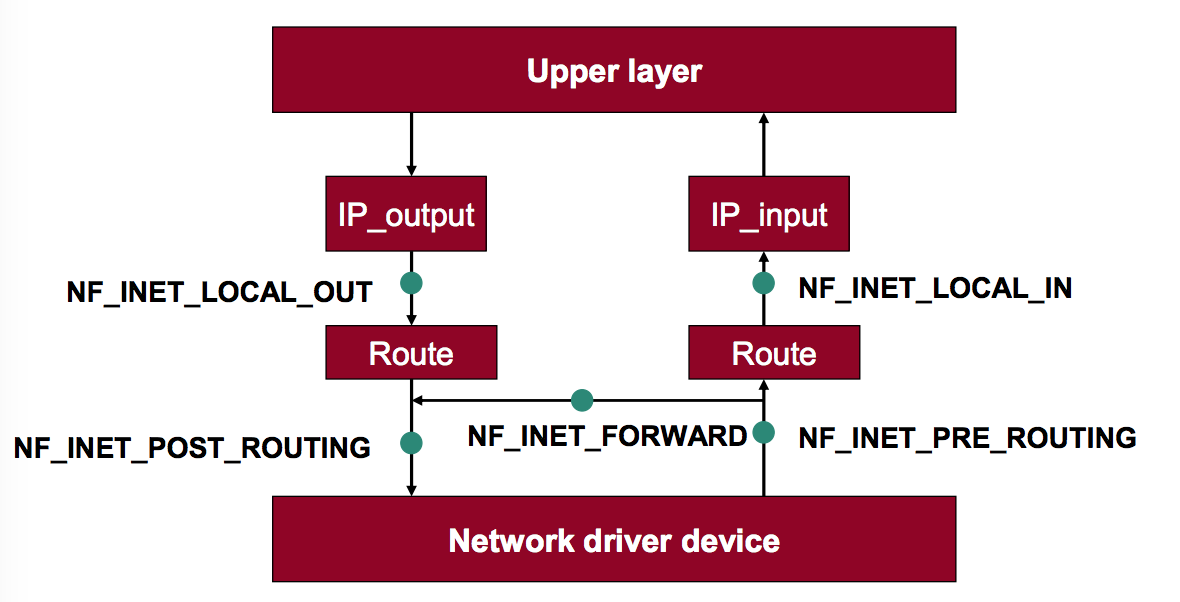
사용언어 : C, Kernel version : linux-kernel-4.4

하드웨어 : processor 2.9GHz intel core i5, 8GB DDR3

**4. Netfilter & Hooking**

넷필터는 packet mangling에 대한 프레임워크이며, 원래 존재하던 네트워크 프로토콜 스택과는 독립적이다. 이 넷필터에서는 페킷들을 필터링 할 수 있으며, 패킷을 잘게 나눌 수도 있고(mangling) 네트워크의 주소를 바꿀 수도 있다. 넷필터에서는 hook 이라는 것을 정의 할 수 있다. 이런 hook은 특정 지점이다. 이 hook point를 지나갈 때 어떠한 함수를 호출 할 수 있다. 이 함수를 지정하고 호출하는 방식을 넷필터에서 등록할 수 있다.

IPv4에서는 5개의 훅 지점이 있다. 이 다섯개의 지점을 그림으로 설명하면 다음과 같다.



**/////////////////// 번역해서 작성해줘**

**NF\_INET\_PRE\_ROUTING**

– After sanity checks, before routing decision

**NF\_INET\_LOCAL\_IN**

– After routing decisions, if the packet destines for this host

**NF\_INET\_FORWARD**

– Packets are not destined for this host but need to be forwarded to another interface

**NF\_INET\_LOCAL\_OUT**

– All packets created from local host would come here before it is been sent out

**NF\_INET\_POST\_ROUTING**

– All packets have been routed and ready to hit the wire

이 hook point를 지나갈 때 어떠한 함수를 호출할지는 우리가 직접 지정할 수 있으며, 이를 이번 텀프로젝트에서 다룰 것이다. 하나의 훅포인트를 지나갈 때 여러개의 함수를 등록할 수 있다. 이때 이 함수의 호출 순서는 따로 우선순위를 지정해야 한다. 우리는 **NF\_INET\_PRE\_ROUTING**에서 패킷 정보를 hooking 하고, 이 정보에 따라 경로를 지정할 것이다. 만약 패킷이 서버의 33333 포트에서 들어왔다면 이 패킷의 Sport와 Dport를 7777로 바꾸어 줄 것이다. **NF\_INㄹT\_PRE\_ROUTING**에 후킹 함수를 추가하는 이유는 패킷이 들어올 때 가장 처음 만나게 되는 후킹 포인트 이며, 이에 따라 패킷을 포워딩시킬지 로컬로 받아드릴지를 결정할 수 있기 때문이다. **NF\_INET\_LOCAL\_IN이나, NF\_INET\_FORWARD는** 이미 경로가 지정된 이후이기 때문에 우리가 원하는 작업을 하기에 적합하지 않다. 이 훅 함수들은 내부에서 어떠한 것이든 할 수 있지만 특정 5가지 값 중 하나를 리턴해야만 한다. 그 값들은 아래와 같다.

NF\_DROP : 현재 패킷을 Drop

NF\_ACCEPT : 현재 패킷을 다음 루틴으로 넘김

NF\_STOLEN : 현재 패킷을 커널이 잊어버림 (뒤처리는 직접)

NF\_QUEUE : 사용자 공간에 올림 (스택을 타지 않고 바로)

NF\_REPEAT : 이 hook을 다시 호출

우리는 이런 넷필터와 훅을 이용하여, skb의 패킷 헤더를 가져와 조건에 따라 포워딩할 수 있도록 만들 것이다.

[참고]

패킷 포워딩은 한 네트워크 세그먼트에서 다른 세그먼트로 패킷을 전달하는 것이다. 주로 라우터 및 스위치에서 수행된다. 패킷이 들어오게 되면 라우터는 패킷의 목적지 IP 주소를 검사한 뒤에 라우터 테이블을 사용하여 가장 최적의 경로를 찾는다.

**5. 네트워크 코드 분석**

//////////////////

**6. 작성 코드 설명**

먼저 3가지의 후킹포인트를 사용했는데, PRE\_ROUTING, FORWARD, POST\_ROUTING 을 사용했다. 먼저 PRE\_ROUTING를 사용한 이유는 receive된 패킷이 가장 처음으로 만나는 후킹포인트이다. 이 곳에서 패킷을 조건에 따라 후킹하여 포워딩 시켜주어야 한다. NF\_INET\_LOCAL\_IN은 이미 LOCAL로 들어가는 후킹포인트이고 이미 라우터를 지나쳤기 때문에 포워딩 처리를 할 수 없다.

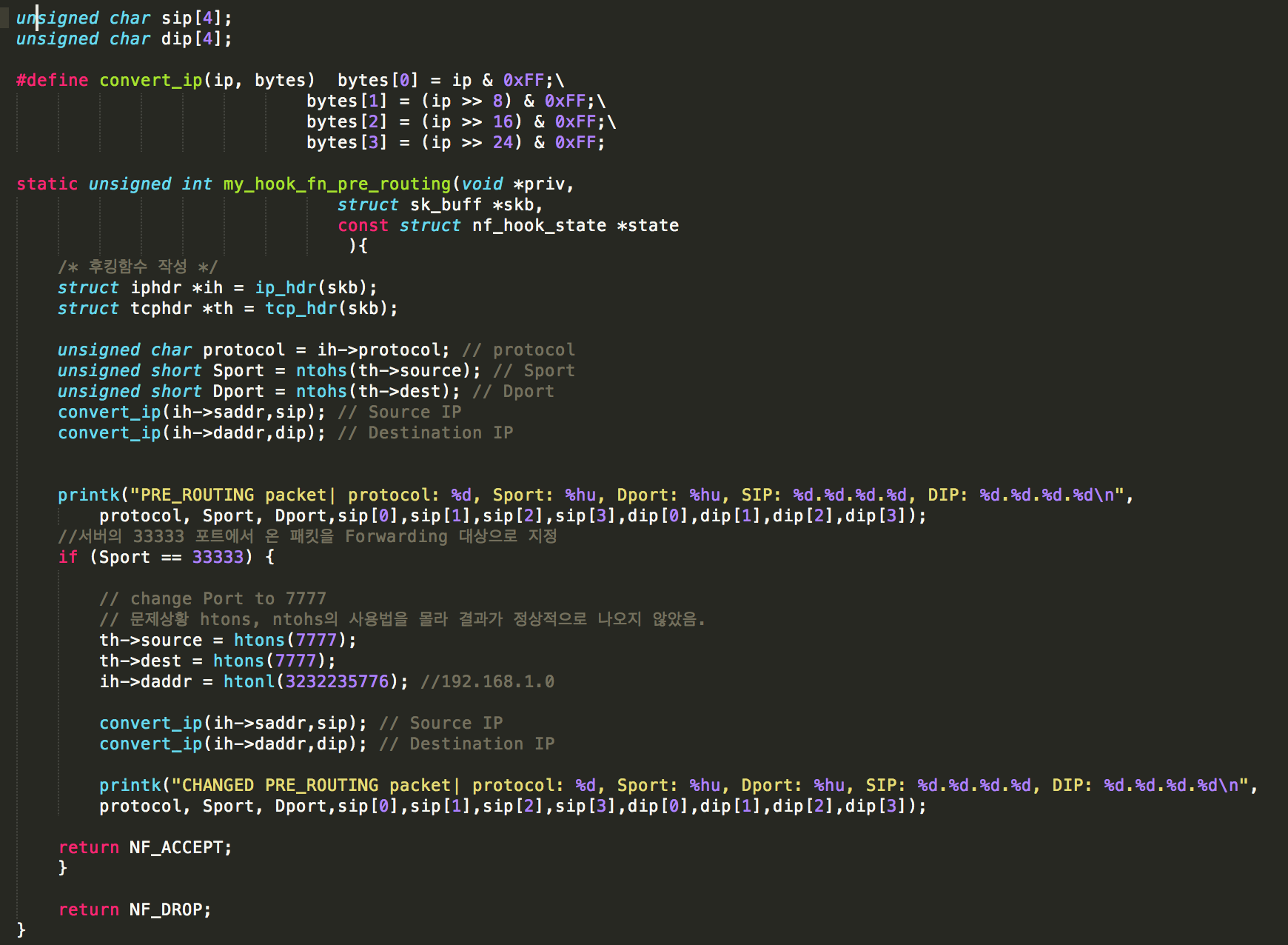
다음으로는 FORWARD, POST\_ROUTING 후킹포인트인데, 여기서는 특정 패킷이 잘 포워딩 되었는지 확인하기 위해 필요하다. 따로 패킷을 변화시키지는 않으며, 출력하기 위해 사용한다. 이 두 지점을 사용하는 이유는 포워딩 패킷이 나가게 되는 두 후킹 포인트이기 때문이다. 여기서 제대로 포워딩이 되는지 확인을 해야한다.

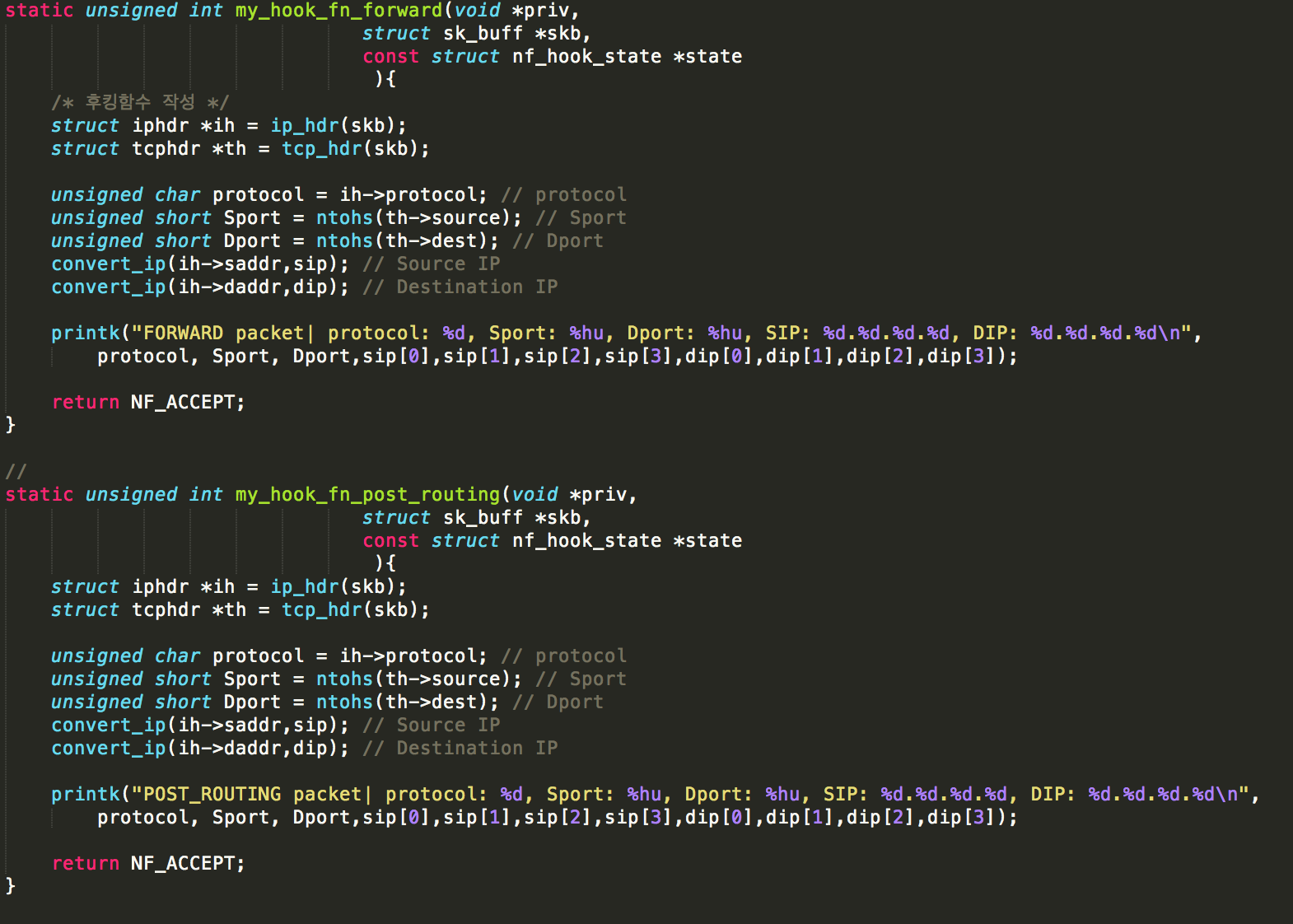
이제 코드를 살펴보면 아래와 같다. 먼저 전역변수로 source IP와 destination IP를 char 배열로 선언해 주었다. 이는 정수 값으로 되어있는 IP 주소를 우리가 흔히 보는 IP 주소의 모양 처럼 바꾸어 주기 위한 변수이다. 그 아래는 #define 지시문을 이용하여 정수 IP를 바꾸어 줄 수 있도록 작성했다. 이 부분을 처음에는 함수로 작성하였는데, C의 포인터의 개념을 잘 기억하지 못해 SIP와 DIP가 똑같이 출력되는 문제가 발생하였다. 이 부분을 구글링과 동기의 도움을 통해 해결하였다.

그 아래는 PRE\_ROUTING 후킹포인트의 후킹 함수를 작성하였다. 먼저 파라미터로 받아온 skb에서 부터 ip해더와 tcp해더를 각각 tcphdr, iphdr로 추출하기 위해 ip\_hdr() tcp\_hdr() 를 사용하였다. 이를 사용하여, 프로토콜, Source Port, Destination Port, SIP, DIP를 추출하였다. 여기서 네트워크의 Big\_Endian과 호스트의 little\_Endian을 맞추어 주기 위하여 ntohs 함수를 사용하였다. 이렇게 받아온 정보를 먼저 출력해 주었다.

그 뒤로는 과제의 조건에 따라 포트 33333으로 들어온 패킷에 대해 Sport 와 Dport를 7777로 바꾸어 주는 작업을 진행하였다. 그리고 destination ip도 바꾸어 주었는데, 이는 포워딩을 위해서 바꾸었다. 포워딩이 되기 위해서는 다른 ip주소로 목적지를 바꾸어 주는 작업이 필요했는데, 이를 위해서 임의의 IP(라우터에 등록된 IP와 같은 IP) 를 등록 시켜주었다. 그 뒤로는 다시 한 번 더 정상적으로 바뀌었는지 확인하기 위해서 한 번 더 패킷을 출력하여 확인해 주었다(과제에 없는 내용) 그 뒤로는 NF\_ACCEPT를 리턴하여 그 뒤의 과정을 진행하도록 하였다.(패킷은 라우터를 지나 포워딩 될 것이다) 다른 패킷들은 NF\_DROP 시켜 주었는데, 과제에 따로 조건이 주어지지 않아 DROP 시켰다. 원래 정상적인 상황이라면 NF\_ACCEPT를 사용하여 다른 패킷들은 정상적으로 원래의 도착지에 도착하도록 해야했을 것이다.

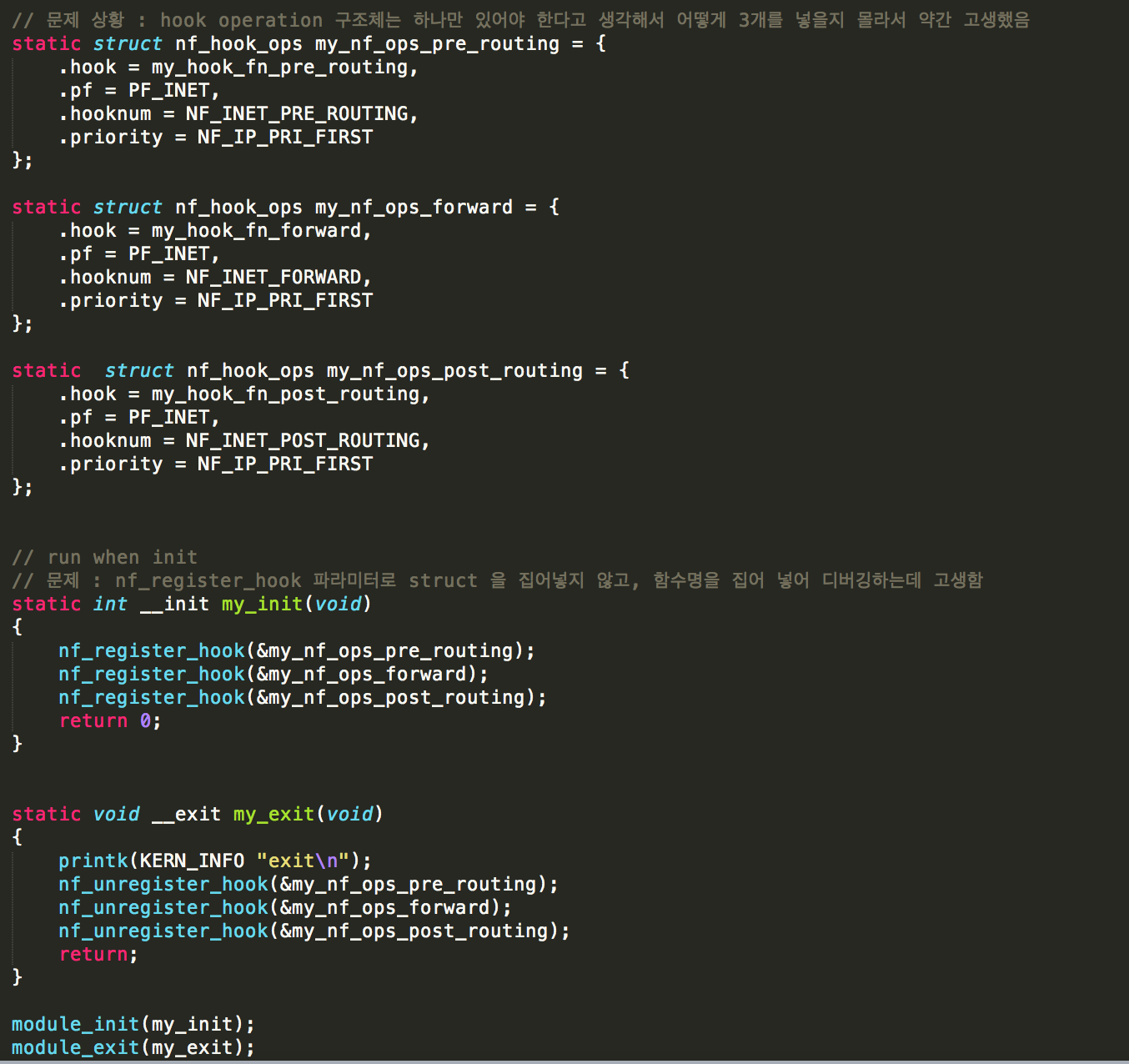
다음으로는 FORWARD 후킹 포인트에 등록할 함수와 POST\_ROUTE 후킹 포인트에 등록할 함수를 작성하여 주었다. 위에서 패킷 정보를 프린트 한 것과 마찬가지로 패킷의 정보를 출력할 수 있도록 하였다.

****



다음으로는 nf\_hook\_ops 구조체를 사용하여 작성한 후킹 포인트에 등록할 함수들을 등록시켜주었다. .hook에 대응되는 함수들을 각각 넣어주었고, .pf에는 INET을 사용하기 때문에 PF\_INET을 모두 넣어주었으며, 각각에 맞는 후킹 포인트들을 .hooknum으로 지정하여 넣어주었다. 또한 과제에 따라 우선순위를 가장 우선으로 해주었다.

그리고, my\_init,과 my\_exit함수를 통해 3개의 후킹 포인트에 대한 함수를 등록시켜 주었고, 모듈이 꺼질 때 후킹 함수들을 등록 취소시켜주었다.

****

**7. 실험 방법 설명 및 로그파일 분석**

**< 실험 방법 >**

1. 코드로 작성한 포워딩의 대상이 되는 ip를 라우팅 테이블에 등록시킨다

2. /proc/sys/net/ipv4/ip\_forward 의 값을 1으로 바꾸어 주어 포워딩이 가능하게끔 만들어 준다.

3. 서버를 동작시켜 패킷을 보내준다.

4. 미리 만들어 둔 모듈을 동작시킨다.

5. warm-up과제에서 만든 client 실행파일을 실행시킨다.

6. dmesg를 사용하여 포워딩과 포트 번호가 정상적으로 바뀌었는지 확인한다.

<로그파일 일부>

[ 5596.183243] PRE\_ROUTING packet| protocol: 6, Sport: 4444, Dport: 49088, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5596.183249] PRE\_ROUTING packet| protocol: 6, Sport: 33333, Dport: 45214, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5596.183250] CHANGED PRE\_ROUTING packet| protocol: 6, Sport: 33333, Dport: 45214, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5596.183259] FORWARD packet| protocol: 6, Sport: 7777, Dport: 7777, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5596.183260] POST\_ROUTING packet| protocol: 6, Sport: 7777, Dport: 7777, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5596.183306] PRE\_ROUTING packet| protocol: 6, Sport: 7777, Dport: 42716, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5596.183309] PRE\_ROUTING packet| protocol: 6, Sport: 6666, Dport: 38562, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5596.183311] PRE\_ROUTING packet| protocol: 6, Sport: 5555, Dport: 41242, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.174338] POST\_ROUTING packet| protocol: 6, Sport: 49088, Dport: 4444, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5597.174348] POST\_ROUTING packet| protocol: 6, Sport: 41242, Dport: 5555, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5597.174417] POST\_ROUTING packet| protocol: 6, Sport: 38562, Dport: 6666, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5597.174434] POST\_ROUTING packet| protocol: 6, Sport: 42716, Dport: 7777, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5597.174450] POST\_ROUTING packet| protocol: 6, Sport: 45214, Dport: 33333, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5597.175615] PRE\_ROUTING packet| protocol: 6, Sport: 5555, Dport: 41242, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.175662] PRE\_ROUTING packet| protocol: 6, Sport: 4444, Dport: 49088, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.175677] PRE\_ROUTING packet| protocol: 6, Sport: 7777, Dport: 42716, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.175692] PRE\_ROUTING packet| protocol: 6, Sport: 6666, Dport: 38562, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.176166] PRE\_ROUTING packet| protocol: 6, Sport: 33333, Dport: 45214, SIP: 192.168.56.101, DIP: 10.0.2.15

[ 5597.176189] CHANGED PRE\_ROUTING packet| protocol: 6, Sport: 33333, Dport: 45214, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5597.176204] FORWARD packet| protocol: 6, Sport: 7777, Dport: 7777, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5597.176212] POST\_ROUTING packet| protocol: 6, Sport: 7777, Dport: 7777, SIP: 192.168.56.101, DIP: 192.168.1.0

[ 5612.918216] POST\_ROUTING packet| protocol: 6, Sport: 45216, Dport: 33333, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5612.918423] POST\_ROUTING packet| protocol: 6, Sport: 41252, Dport: 5555, SIP: 10.0.2.15, DIP: 192.168.56.101

[ 5612.918793] POST\_ROUTING packet| protocol: 6, Sport: 38572, Dport: 6666, SIP: 10.0.2.15, DIP: 192.168.56.101

우리가 바꾸어야 할 포트 넘버 33333에서 들어온 패킷이 먼저 PRE\_ROUTING hooking point에서 그 정보가 프린팅 된다. 이 훅 포인트에서 33333이 들어오게 되면 과제의 조건과 같이 포트 번호를 7777로 모두 바꾸어 준다. 그리고, 포워딩하기 위해 destination IP를 바꾸어준다. 바뀌어 진 것을 확인 하기 위해서 과제의 조건에는 없지만 CHANGED PRE\_ROUTING packet이라는 출력 포인트를 만들어 두었다. 여기서 정상적으로 바뀐 것을 확인할 수 있다. 이렇게 바뀌면 이 패킷이 로컬로 들어가는 것이 아니라 포워딩 된다. 따라서 FORWARD hooking point에서 패킷 정보가 출력된다. 여기서는 PRE\_ROUTING에서 정보가 수정된 대로, 출력되게 된다. FOWARDING된 패킷은 로컬에서 다른 쪽으로 보내는 것과 같이 처리 되기 때문에 POST\_ROUTING hooking point를 지나가게 된다. 따라서 POST\_ROUTING에서도 한 번 정보가 출력되게 된다.

**8. 과제 수행 시 어려웠던 부분과 해결 방법**

- 팀원 중 한 명의 리눅스 커널이 모듈을 올렸을 때 커널 패닉에 계속해서 빠져 버렸다.

- C언어가 익숙하지 않아 고생을 했다.(포인터 개념)

- hook operation 구조체는 하나만 있어야 한다고 생각해서 어떻게 3개를 넣을지 몰라서 약간 고생했다.

- nf\_register\_hook 파라미터로 struct 을 집어넣지 않고, 함수명을 집어 넣어 디버깅하는데 고생했다.

- htons, ntohs의 사용법을 몰라 결과가 정상적으로 나오지 않았다.

- ip\_forward의 값을 1로 만들지 않아 포워딩이 되지 않았었다.