Fall ’19 COSE322-00

System Programming

Practice 8. Netfilter

2019. 11. 28.

Contents

- **Netfilter**
  - Packet Forward
  - Netfilter
    - Definition and role
    - Netfilter hook points
    - Netfilter hook functions
    - Netfilter implementation
- **Actual implementations for Assignment #2**
  - Loadable kernel module
  - Proc fs
  - Netfilter
    - How to implement networking filtering module
    - How to get packet's header regions

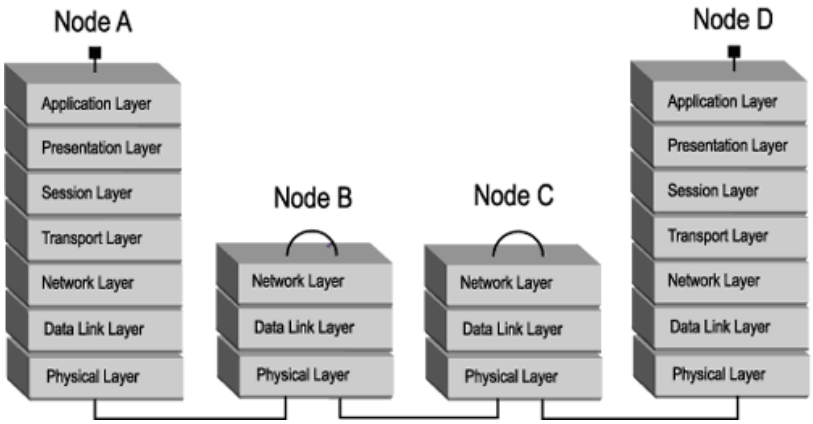
Netfilter

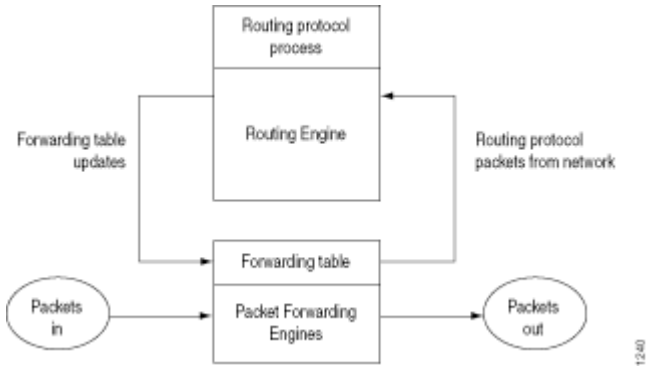
- **Packet Forward**
- **Netfilter**
  - Definition and role
  - Netfilter hook points
  - Netfilter hook functions
  - Netfilter implementation

Packet Forward

❖ Packet Forward

- Packet forwarding is the relaying of packets from one network segment to another by nodes in a computer network.
  - The Network Layer in the OSI model is responsible for packet forwarding
  - Packet forwarding is primarily performed on Routers and switches.
- Routers examine a packet's destination IP address and determine the best path by enlisting the aid of a routing table





Packet Forward

❖ Packet Forward

- IP forwarding is also called Kernel IP forwarding
  - It is a feature of Linux Kernel
- This feature can be enabled or disabled in Linux Kernel
  - Read /proc/sys/net/ipv4/ip_forward
    - 1: Enable
    - 0: Disable
  - Echo is used to turn this feature on and off
    - ex) echo 1 > /proc/sys/net/ipv4/ip_forward

Netfilter

- **Concepts**
  - A framework for packet mangling in kernel
  - Built in Linux 2.4 kernel or higher version
  - Independent of network protocol stack
  - Provide an easy way to do firewall setting or packet filtering, network address translation and packet mangling

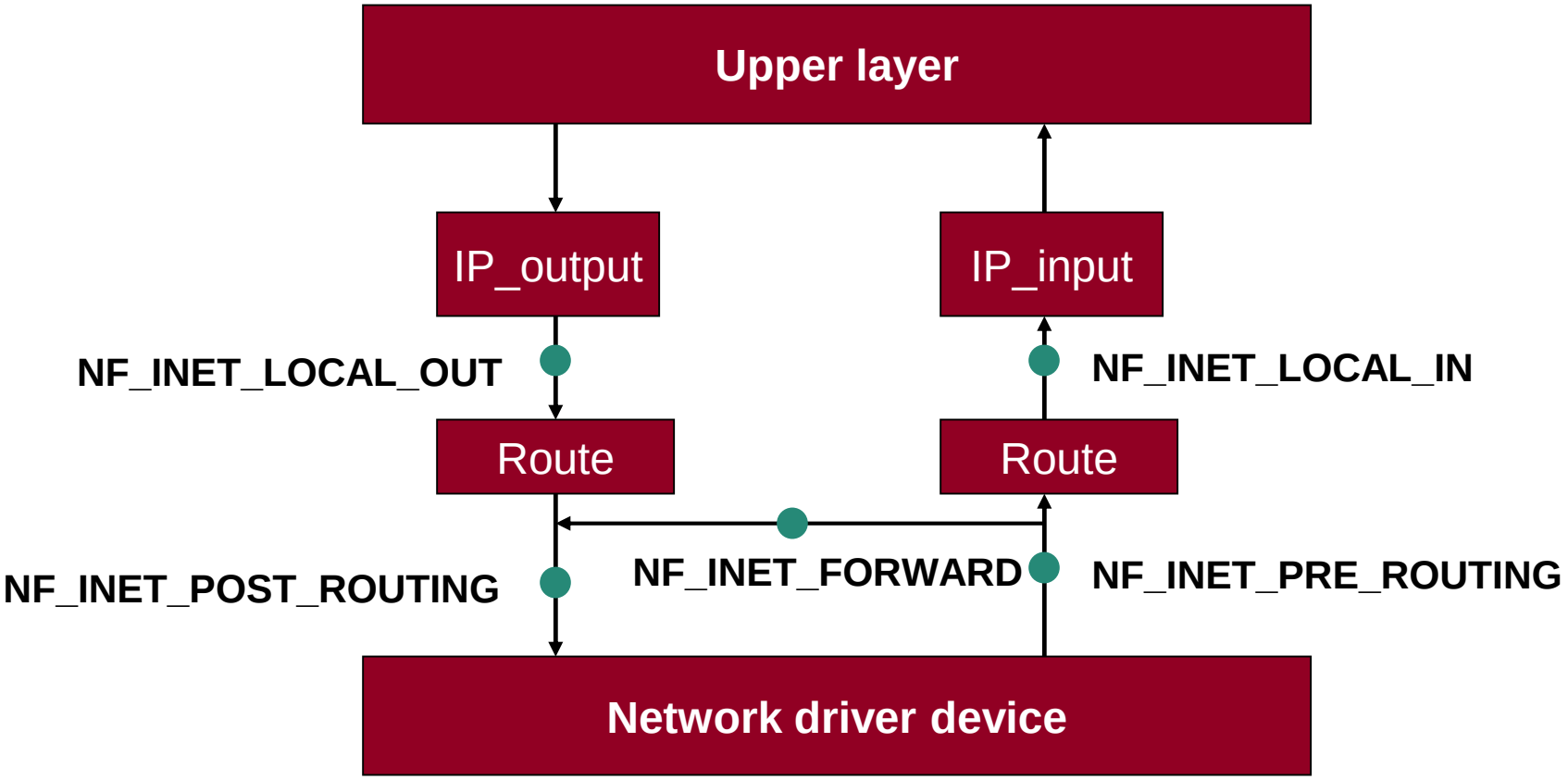
Netfilter

❖ Mechanisms

- Defines a set of hooks
  - Hooks are well defined point in the path of packets when these packets pass through network stack
  - The protocol code will jump into Netfilter framework when it hits the hook point
- Registers Kernel functions to these hooks
  - Called when a packet reaches at hook point
  - Can decide the fate of the packets
  - After the functions, the packet could continue its journey

Netfilter Hooks

- **Each protocol family can have different hooks**
  - IPv4 defines 5 hooks



Netfilter Hook Points

- **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

Netfilter Hook Functions

- **Multiple functions could register to the same hook point**
  - 각 함수의 priority를 설정해 주어야함
  - 패킷이 hook points를 통과할 때, hook point에 등록 된 각 함수는 priority 순서에 따라 호출 됨

- **Hook functions**
  - The function could do anything
  - netfilter에 다음 다섯가지 value중 하나를 return 해야함:
    - 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

Netfilter implementation

❖ 후킹 포인트에 nf_hook_ops를 등록 / 해제하는 함수

include/linux/netfilter.h

```
124 /* Function to register/unregister hook points. */
 …
132 int nf_register_hook(struct nf_hook_ops *reg);
133 void nf_unregister_hook(struct nf_hook_ops *reg);
 …
```

- nf_register_hook() : nf_hook_ops를 넷필터에 등록시킴
- nf_unregister_hook() : nf_hook_ops를 넷필터에서 해제시킴

Netfilter implementation

❖ 후킹 포인트에 등록하는 구조체

include/linux/netfilter.h

```
87 struct nf_hook_ops {
88         struct list_head        list;
89
90         /* User fills in from here down. */
91         nf_hookfn               *hook;
92         struct net_device       *dev;
93         void                    *priv;
94         u_int8_t                pf;
95         unsigned int            hooknum;
96         /* Hooks are ordered in ascending priority. */
97         int                     priority;
98 };
```

- 채워 넣어야 할 field는 *hook, pf, hooknum, priority임
  - *hook : 후킹 포인트에 등록하려는 함수
  - pf : 네트워크 family (TCP/IP의 경우 PF_INET)
  - hooknum : 후킹 포인트
  - priority : 이 후킹의 우선순위

Netfilter implementation

❖ 후킹 함수 구조

include/linux/netfilter.h

```
83 typedef unsigned int nf_hookfn(void *priv,
84                               struct sk_buff *skb,
85                               const struct nf_hook_state *state);
```

- skb : a packet

❖ 후킹 함수의 return value

include/uapi/linux/netfilter.h

```
10 /* Responses from hook functions. */
11 #define NF_DROP 0
12 #define NF_ACCEPT 1
13 #define NF_STOLEN 2
14 #define NF_QUEUE 3
15 #define NF_REPEAT 4
16 #define NF_STOP 5
```

- NF_DROP : 현재 패킷을 Drop
- NF_ACCEPT : 현재 패킷을 다음 루틴으로 넘김

Function Pointer의 가독성 높이기

❖ 함수의 Prototype을 typedef로 변수처럼 선언한다.

```
int test(int num){
        printf("test %d \n", num);
}

int main(){
        int (*testptr)(int a);
        testptr = test;
        testptr(100);
        return 0;
}
```

```
typedef int TestFuncPtr(int a)

int test(int num){
        printf("test %d \n", num);
}

int main(){
        TestFuncPtr *testptr = test;
        testptr(100);
        return 0;
}
```

Netfilter implementation

❖ 후킹 포인트들 (후킹 함수를 등록하는 지점)

include/uapi/linux/netfilter.h

```
46 enum nf_inet_hooks {
47        NF_INET_PRE_ROUTING,
48        NF_INET_LOCAL_IN,
49        NF_INET_FORWARD,
50        NF_INET_LOCAL_OUT,
51        NF_INET_POST_ROUTING,
52        NF_INET_NUMHOOKS
53 };
```

/* NF_INET_NUMHOOKS는 enum의 마지막 인덱스로, 후킹 포인트 개수를 나타내기 위함 */

nf_hook_ops 구조체의 hooknum 필드값

❖ Hooking priorities

include/uapi/linux/netfilter_ipv4.h

```
57 enum nf_ip_hook_priorities {
58        NF_IP_PRI_FIRST = INT_MIN,
59        NF_IP_PRI_CONNTRACK_DEFRAG = -400,
…
…
71        NF_IP_PRI_LAST = INT_MAX,
72 };
```

nf_hook_ops 구조체의 priority필드값

Actual implementations for Assignment #2

- Loadable kernel module
- Proc fs
- Netfilter

Loadable kernel module

- **Note**
  - It is a chunk of code, inserted and unloaded dynamically
  - Like the normal user space programs but it works in kernel space a nd have access to kernel resources
  - Modules take effect immediately after loading it without recompiling
  - Saving time to extend the kernel

- **Different from normal C program**
  - Modules execute in kernel space
  - May not use some standard function libraries of C
  - No main functions
    - Module_init(“startfunctionname”); / Module_exit(“endfunctionname”);

Loadable kernel module : 소스파일 작성

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init simple_init(void){
        printk(KERN_INFO "Simple Module Init!!\n");
        return 0;
}
static void __exit simple_exit(void){
        printk(KERN_INFO "Simple Module Exit!!\n");
}

module_init(simple_init);
module_exit(simple_exit);

MODULE_AUTHOR("Korea University");
MODULE_DESCRIPTION("It's simple!!");
MODULE_LICENSE("GPL");
MODULE_VERSION("NEW");
```

자세한 LKM내용은 지난 실습자료 및 구글링 등을 통한 참고자료 확인

How to implement networking filtering module

- **Steps**
  - Fill out the **nf_hook_ops** structure
  - Write a **hook function**
    - It has specific format
  - **Register** nf_hook_ops to system
  - **Compile** and load the modules

How to implement networking filtering module

❖ Step 1) Fill out the nf_hook_ops structure

include/linux/netfilter.h

```
87 struct nf_hook_ops {
…
90        /* User fills in from here down. */
91        nf_hookfn               *hook;
…
98 };
```

```
static struct nf_hook_ops my_nf_ops{
        .hook = my_hook_fn,
        /* hook operation 구조체 나머지필드 채움*/
}
```

- 직접 채워 넣어야 할 field는 *hook, pf, hooknum, priority임
  - *hook : 후킹 포인트에 등록하려는 함수
  - pf : 네트워크 family (TCP/IP의 경우 PF_INET)
  - hooknum : 후킹 포인트
  - priority : 이 후킹의 우선순위

How to implement networking filtering module

❖ Step 2) Write a hook function

– 후킹 함수는 다음의 정의에 맞게 작성해야 함

include/linux/netfilter.h

```
83 typedef unsigned int nf_hookfn(void *priv,
84                               struct sk_buff *skb,
85                               const struct nf_hook_state *state);
```

```
static unsigned int my_hook_fn(void *priv,
                               struct sk_buff *skb,
                               const struct nf_hook_state *state){
        /* … */
}
```

– 반드시 return value는 사전에 정의된 return value 중 하나여야 함

- NF_DROP, NF_ACCPET, etc.

How to implement networking filtering module

❖ Step 3) Register nf_hook_ops to system

- 모듈이 로드되었을 때 실행되는 함수에서 아래의 후킹 포인트 등록 함수를 사용

include/linux/netfilter.h

```
124 /* Function to register/unregister hook points. */
 …
132 int nf_register_hook(struct nf_hook_ops *reg);
133 void nf_unregister_hook(struct nf_hook_ops *reg);
 …
```

```
nf_register_hook(&my_nf_ops);
/* … */
nf_unregister_hook(&my_nf_ops);
```

- 모듈이 내려갈 때 실행되는 함수에서 후킹 포인트에서 해제시키는 함수를 실행할 것

How to get packet's header regions

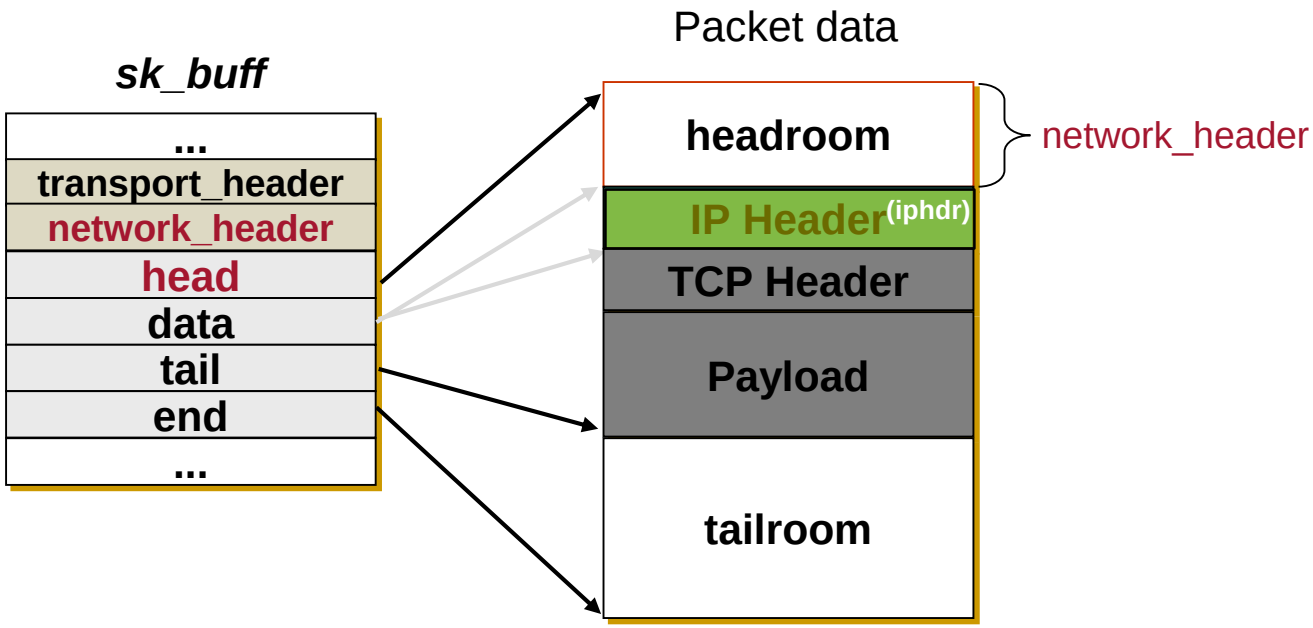
❖ 헤더 영역을 가져오는 방법

- 소켓 버퍼의 특정 헤더 영역을 잡는 방법
  - xxx_hdr 매크로
  - xxx_hdrlen 매크로
  - linux/ip.h, linux/tcp.h에서 확인

```
static unsigned int my_hook_fn(void *priv,
                               struct sk_buff *skb,
                               const struct nf_hook_state *state){

        struct iphdr  *ih = ip_hdr(skb);
        struct tcphdr *th = tcp_hdr(skb);
}
```

How to get packet's header regions



(struct iphdr *) skb->head + skb->network_header

iphdr구조체 포인터로 Type casting : **(head의 주소 + ip헤더 offset)의 위치로**

How to get packet's header regions

❖ ip_hdr()

ip_hdr()
include/linux/ip.h

```
23 static inline struct iphdr *ip_hdr(const struct sk_buff *skb)
24 {
25         return (struct iphdr *)skb_network_header(skb);
26 }
```

skb_network_header()
include/linux/skbuff.h

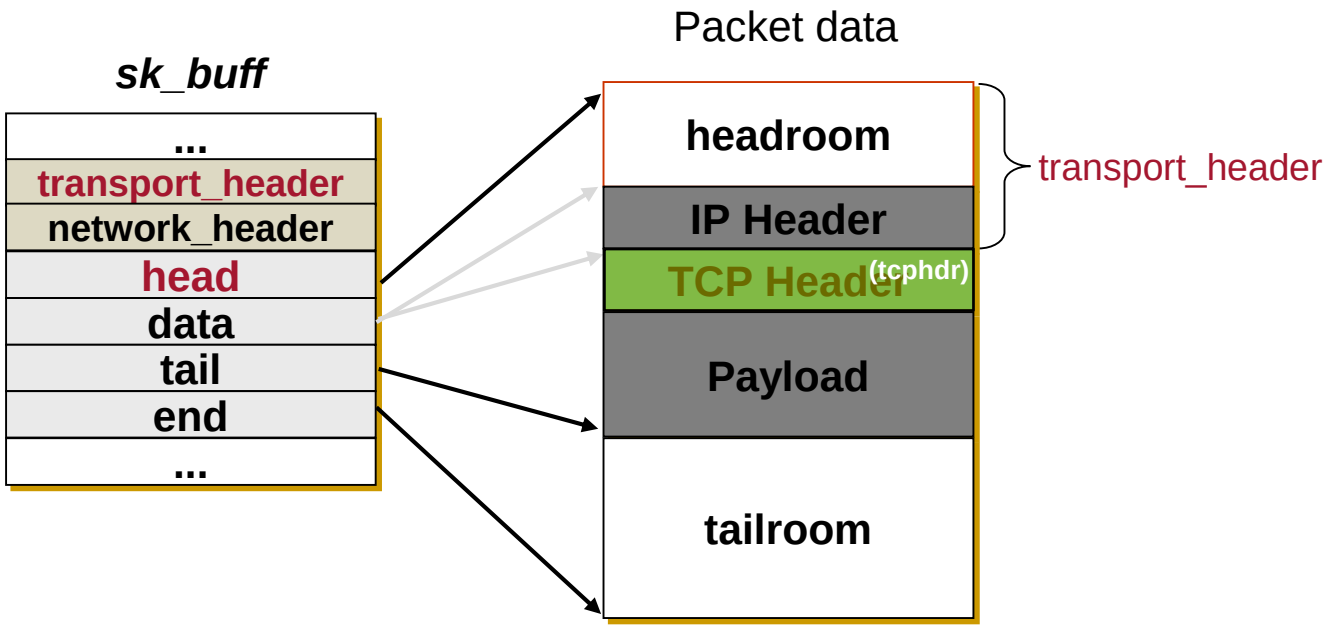
```
2012 static inline unsigned char *skb_network_header(const struct sk_buff *skb)
2013 {
2014         return skb->head + skb->network_header;
2015 }
```

❖ ip_hdrlen()

ip_hdrlen()
include/net/ip.h

```
53 static inline unsigned int ip_hdrlen(const struct sk_buff *skb)
54 {
55         return ip_hdr(skb)->ihl * 4;
56 }
```

How to get packet's header regions



(struct tcphdr *) skb->head + skb->transport_header

tcphdr구조체 포인터로 Type casting : **(head의 주소 + tcp헤더 offset)의 위치로**

How to get packet's header regions

❖ tcp_hdr()

tcp_hdr()
include/linux/tcp.h

```
27 static inline struct tcphdr *tcp_hdr(const struct sk_buff *skb)
28 {
29         return (struct tcphdr *)skb_transport_header(skb);
30 }
```

skb_transport_header()
include/linux/skbuff.h

```
1995 static inline unsigned char *skb_transport_header(const struct sk_buff *skb)
1996 {
1997         return skb->head + skb->transport_header;
1998 }
```

❖ tcp_hdrlen()

tcp_hdrlen()
include/linux/tcp.h

```
32 static inline unsigned int tcp_hdrlen(const struct sk_buff *skb)
33 {
34         return tcp_hdr(skb)->doff * 4;
35 }
```

소스파일 작성

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <linux/netfilter.h>
/* 그 외 include 및 define */
static unsigned int my_hook_fn(void *priv,
                               struct sk_buff *skb,
                               const struct nf_hook_state *state){
        /* 후킹함수 작성 */
}
static struct nf_hook_ops my_nf_ops{
        .hook = my_hook_fn,
        /* hook operation 구조체 나머지필드 채움*/
}
static int __init simple_init(void){
        proc_dir = proc_mkdir(…);
        proc_file = proc_create(…);
        nf_register_hook(&my_nf_ops);
        return 0;
}
static void __exit simple_exit(void){
        remove_proc_entry(…);
        nf_unregister_hook(&my_nf_ops);
}
module_init(simple_init);
module_exit(simple_exit);

MODULE_AUTHOR("Korea University");
MODULE_DESCRIPTION("It's simple!!");
MODULE_LICENSE("GPL");
MODULE_VERSION("NEW");
```

주의사항

- 넷필터를 이용한 패킷 필터 모듈 구현은 참고자료가 매우 다양
- 다만 구현된 소스와 차이가 나는 점이 곳곳에 있으므로, 오류 발생 시 실습자료에 제시한 소스코드 파일을 중심으로 오류 발생을 찾고 해결할 것
- 구현의 문제 • 오류 → 시간을 두고 보거나 Google을 통해 해결가능
- 그 밖의 오류 (시스템 오류 등등) → Google 검색 후 안되면, Facebook