

NETWORK SECURITY

한양대학교 소프트웨어융합대학 소프트웨어학부
이연준 교수

주요 사항

■ **Firewall** 에 대한 이해

■ **Lab Overall**

■ **Lab Preparation**

- 실습 환경 구성

■ **Lab Task**

■ **Lab Question**

■ **Evaluation**

Firewall (침입 차단 시스템, 방화벽)



Firewall (침입 차단 시스템, 방화벽)

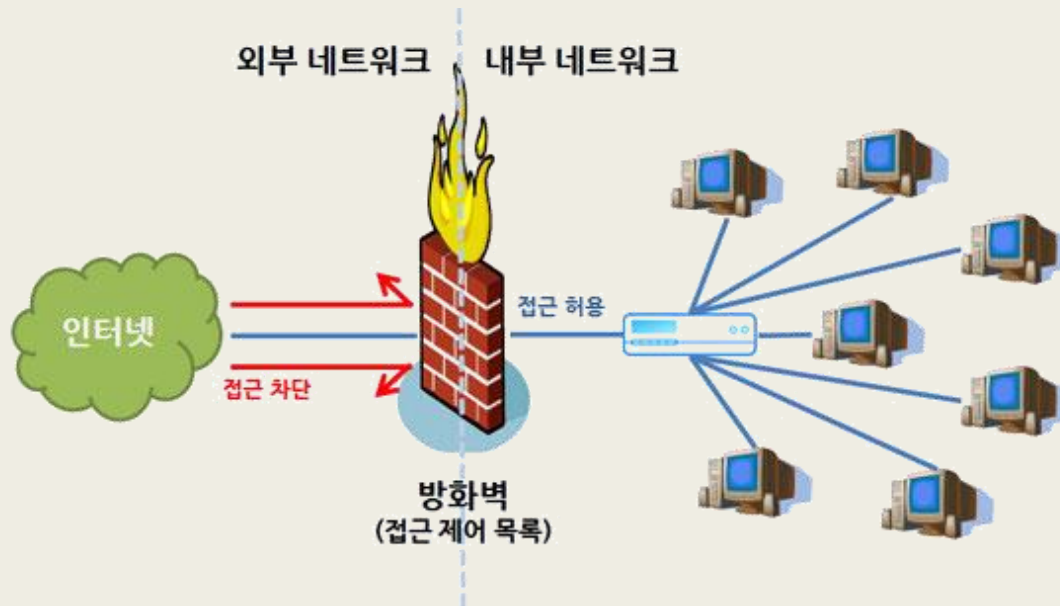
■ Firewall은 왜 필요한가?

- **Network**를 위협하는 다양한 공격들이 존재
 - **DoS, Packet Sniffing, IP Spoofing, etc**
- **Network**를 위협하는 요소는 지속적으로 증가
- **Network**를 효율적이고 안전하게 관리할 수 있는 안전한 보안 정책 및 시스템 도입이 필요

Firewall (침입 차단 시스템, 방화벽)

■ Firewall이란?

- 외부의 악의적인 사용자들의 침입으로부터 내부망을 보호하기 위한 정책 및 이를 지원하는 **H/W** 및 **S/W**를 총칭
- 내부의 신뢰할 수 있는 **Network**과 외부의 신뢰할 수 없는 **Network** 사이에 위치



Firewall (침입 차단 시스템, 방화벽)

■ Firewall의 설계 목표

- 내부에서 외부로 나가는 모든 트래픽과 그 반대의 경우도 모두 **Firewall**을 통과해야 함 → **Firewall**을 거치지 않고 **Local Network**에 접근하는 것을 물리적으로 차단
- 인가 받은 트래픽만 통과
- 안전한 **OS**를 가진 믿을 수 있는 시스템을 사용해야 함

■ Inbound / Outbound

- **Firewall**의 시점에서 판단
- **Inbound** = 외부 → 내부로 들어오는 **Packet**
- **Outbound** = 내부 → 외부로 나가는 **Packet**

Firewall (침입 차단 시스템, 방화벽)

■ Firewall의 주요 기능

- 접근 통제

- **Host, User, Service**의 속성을 기초로 내부망에 대한 접근 통제
- **Packet Filtering**

- 식별 및 인증

- 내부망으로 접근하려는 **User** 또는 **Computer**의 신원을 식별하고 인증

- 사고 발생 시 추적

- 모든 **Traffic**은 **Firewall**을 거침 → 접속 정보에 대한 기록을 보유 가능
- 로그 정보를 통해 접근 통계, 취약성 점검, 역추적 가능

Firewall (침입 차단 시스템, 방화벽)

■ Firewall의 주요 기능

- 암호화
 - 기밀성, 무결성 기능 제공
- 주소 변환 (**Network Address Translation, NAT**)
 - **IP Address** 변환 기능 제공

LAB OVERALL

학습 목표

- 실습을 통해 **Firewall**이 작동하는 방식에 대한 이해
- 간단한 **Packet Filtering** 방식의 **Firewall** 구현
- 해당 실습에서 다루는 것들
 - **UFW (Uncomplicated FireWall)**
 - **Netfilter**
 - **Loadable Kernel Module**

Firewall in Linux OS

- **Linux OS**는 기본적으로 **iptables**라는 **Tool**을 통해 **Firewall**을 사용
- **UFW (Uncomplicated FireWall)**
 - **Debian** 계열 및 **Ubuntu**에서 사용하는 **Firewall**
 - 기본적인 프로그램의 구성은 **iptables**를 사용
 - **Packet Filtering** 방식
 - 상대적으로 사용하기 간편함

Packet Filtering Using by LKM & Netfilter

- **iptables**와 **UFW**는 **Packet Filtering** 방식의 **Firewall**
- **In/Outbound Packet**을 분석하고 이를 관리자가 설정한 정책에 따라 **Filtering**을 시행
- **Packet**은 **Kernel**에서 처리함 → **Filtering** 또한 **Kernel** 단에서 처리
- **Kernel**을 수정 및 재구축해야 하는 번거로움
- 최근의 **Linux OS**는 **Packet** 조작을 위한 메커니즘을 제공
 - **LKM (Load Kernel Module)**
 - **Netfilter**

Packet Filtering Using by LKM & Netfilter

■ LKM (Loadable Kernel Module)

- 실행 중인 **Kernel**에 새로운 **Module**을 추가 가능
- **Kernel**을 재구축하거나 **Rebooting** 필요 없음
- **Firewall**의 **Packet Filtering** 부분은 **LKM**으로 구현
- **But, In/Outbound Packet**을 차단하려면 **Packet** 처리 경로에 해당 **Module**을 삽입해야 함

Packet Filtering Using by LKM & Netfilter

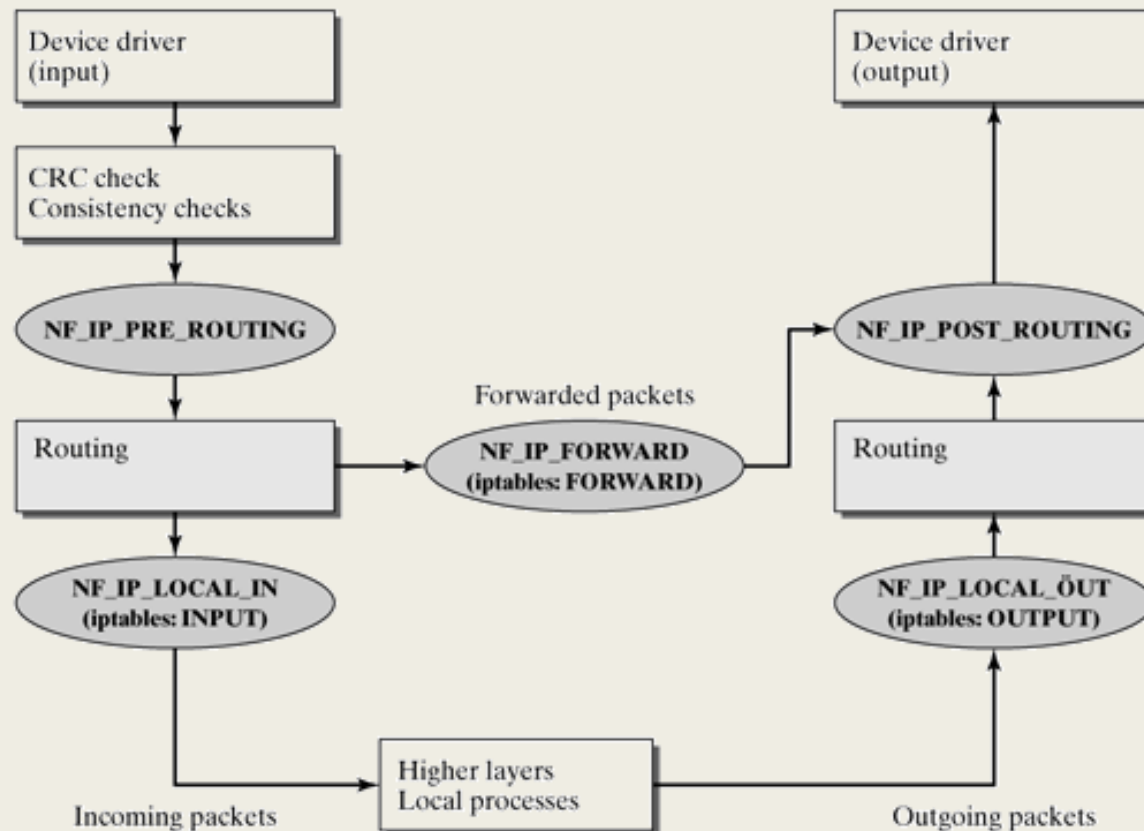
■ Netfilter

- 인가된 사용자가 **Packet**을 쉽게 조작할 수 있음
- **Kernel**에서 **Packet**을 처리하는 과정에 필요하면 **Rule**에 따라 처리 가능하도록 5군데의 **Hook**을 제공
- **Hook**의 원래 뜻은 ‘갈고리’로 **Netfilter**에서의 **Hook**은 특정 지점에서 **Packet**을 가로챌다는 개념

■ LKM + Netfilter를 통해 Packet Filtering Module을 구현 가능함

Packet Filtering Using by LKM & Netfilter

■ Netfilter Hook Point



LAB PREPARATION

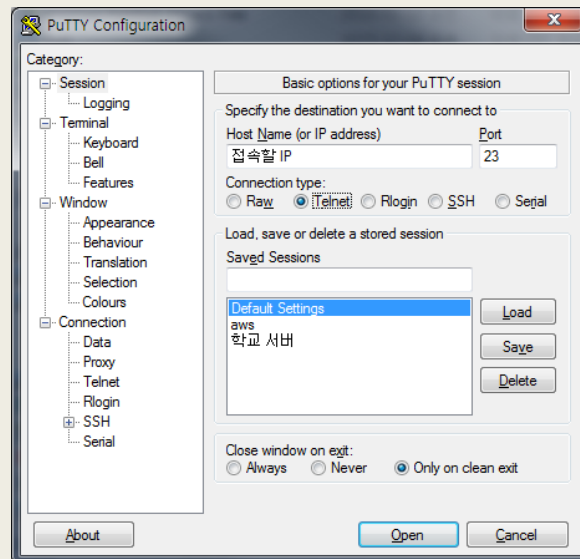
실습 환경 구성 준비

- 실습 환경은 지난 실습에 사용했던 **2대의 VM**을 사용
 - **Firewall**을 사용할 **VM**은 **1대**
 - **VM_A : Firewall**을 사용하는 **Server**
 - **VM_B : Client**

- 실습에 사용할 **Package** 설치 : **w3m**
 - **CLI** 환경에서 사용 가능한 **Web Browser**
 - **sudo apt-get install w3m w3m-img**
 - **Usage : w3m [URL]**

실습 환경 구성 준비

- 실습에 사용할 **Tool (putty) Download**
 – **<https://www.putty.org/>**



- **Windows** 환경이 아닐 경우, **Mac**은 자체 **Terminal** 이용

실습 환경 구성 준비

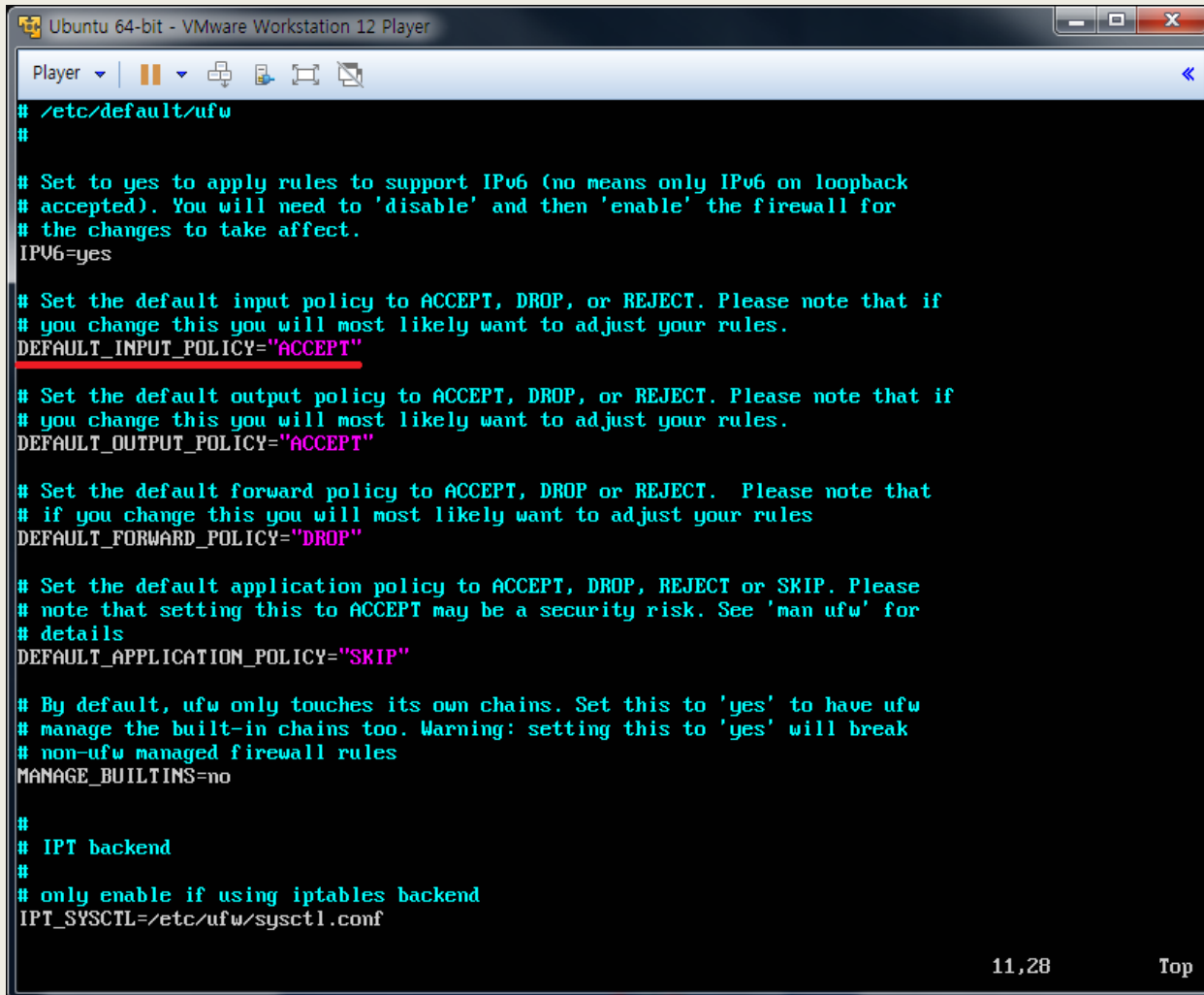
■ Firewall 구현 실습을 위해 별도의 **directory** 생성

- **mkdir firewall_task**_본인학번

■ **ufw** 기본 정책 변경

- **ufw**는 기본적으로 활성화되어 있지 않음
- 또한 기본 정책으로 들어오는 트래픽은 **DROP** 처리하게 되어 있으므로 다음과 같이 변경해야 함
- **sudo vi /etc/default/ufw**
- **DEFAULT_INPUT_POLICY="DROP"**을
DEFAULT_INPUT_POLICY="ACCEPT"로 변경

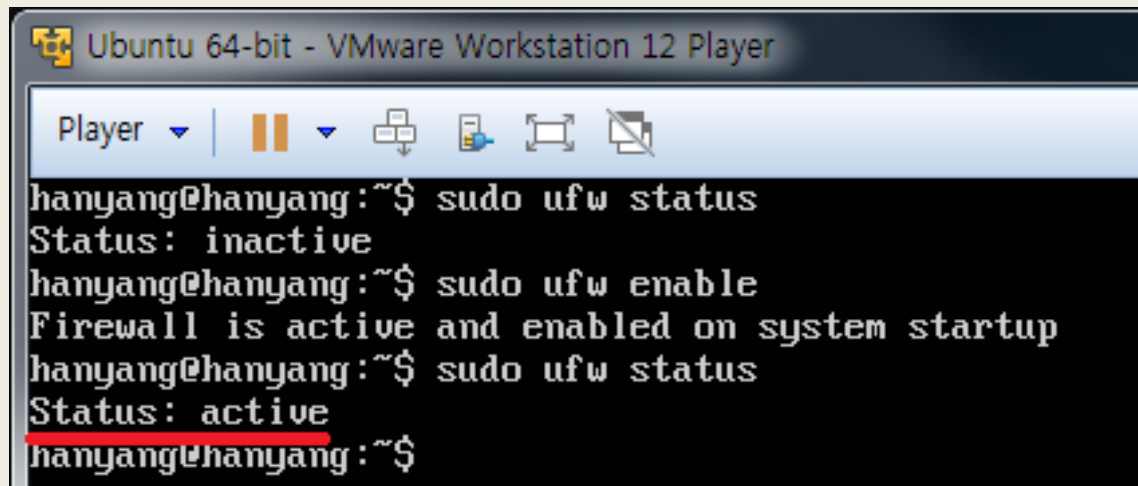
실습 환경 구성 준비



```
Ubuntu 64-bit - VMware Workstation 12 Player
Player
# /etc/default/ufw
#
# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback
# accepted). You will need to 'disable' and then 'enable' the firewall for
# the changes to take affect.
IPv6=yes
# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="ACCEPT"
# Set the default output policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_OUTPUT_POLICY="ACCEPT"
# Set the default forward policy to ACCEPT, DROP or REJECT. Please note that
# if you change this you will most likely want to adjust your rules
DEFAULT_FORWARD_POLICY="DROP"
# Set the default application policy to ACCEPT, DROP, REJECT or SKIP. Please
# note that setting this to ACCEPT may be a security risk. See 'man ufw' for
# details
DEFAULT_APPLICATION_POLICY="SKIP"
# By default, ufw only touches its own chains. Set this to 'yes' to have ufw
# manage the built-in chains too. Warning: setting this to 'yes' will break
# non-ufw managed firewall rules
MANAGE_BUILTINS=no
#
# IPT backend
#
# only enable if using iptables backend
IPT_SYSCTL=/etc/ufw/sysctl.conf
11,28 Top
```

실습 환경 구성 준비

- 변경된 **ufw** 정책 반영을 위해 **service** 재시작 및 활성화
 - **sudo service ufw restart**
 - **sudo ufw enable**
 - **sudo ufw status**

A screenshot of a terminal window titled 'Ubuntu 64-bit - VMware Workstation 12 Player'. The terminal shows the following commands and output:

```
hanyang@hanyang:~$ sudo ufw status
Status: inactive
hanyang@hanyang:~$ sudo ufw enable
Firewall is active and enabled on system startup
hanyang@hanyang:~$ sudo ufw status
Status: active
hanyang@hanyang:~$
```

The output 'Status: active' is highlighted with a red underline.

LAB TASK

Using Firewall

■ ufw에 다음 세 가지 **Rule**을 적용할 것

1. **VM_A**에서 **VM_B**로 **telnet** 연결을 허용하지 않음
2. **VM_B**에서 **VM_A**로 **telnet** 연결을 허용하지 않음
3. **VM_A**에서 학교 **Web Site**에 접속을 허용하지 않음

■ 각 **Rule**을 적용하며 실습 내용을 **Capture**

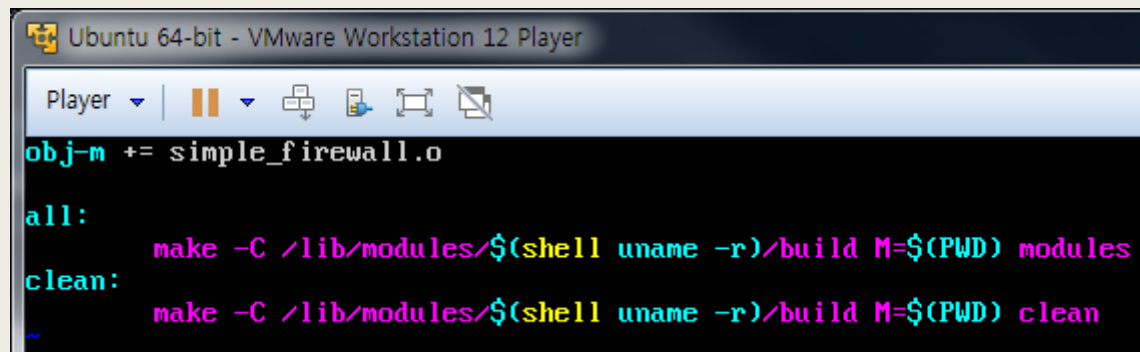
- **Rule** 설정이 되기 전과 후의 **telnet** 접속 여부 확인
- **Rule** 설정이 되기 전과 후의 **w3m**을 통한 학교 **Web Site** 접속 여부 확인

■ Hint

1. **ufw** 사용법은 **man ufw** 또는 **ufw help**를 통해 확인
2. 학교의 **IP**는 **Ping**을 통해 확인할 수 있음

Implement Simple Firewall Using by LKM + Netfilter

- 구현한 **Firewall**과 겹칠 수 있으므로 이전 **Task**에서 활성화 하한 **ufw**를 비활성화
 - **sudo ufw disable**
- 실습을 위해 만든 **directory**로 이동하여 다음 파일을 작성
 - **vi Makefile**



```
obj-m += simple_firewall.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```


Implement Simple Firewall Using by LKM + Netfilter

■ simple_firewall.c (Skeleton Code)

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/inet.h>

static struct nf_hook_ops nfho;

void print_addr(struct iphdr *iph)
{
    printk(KERN_INFO "DROPPING PACKET FROM %d.%d.%d.%d to %d.%d.%d.%d\n",
        iph->saddr & 0x000000ff,
        (iph->saddr & 0x0000ff00) >> 8,
        (iph->saddr & 0x00ff0000) >> 16,
        (iph->saddr & 0xff000000) >> 24,
        iph->daddr & 0x000000ff,
        (iph->daddr & 0x0000ff00) >> 8,
        (iph->daddr & 0x00ff0000) >> 16,
        (iph->daddr & 0xff000000) >> 24);
}

unsigned int hook_func(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;
    iph = ip_hdr(skb);
    tcph = (void *)iph + iph->ihl * 4;

    if(iph->protocol == ... && tcph->dest == ... &&
        iph->saddr == ... && iph->daddr == ...)
    {
        print_addr(iph);
        return NF_DROP;
    }
}
```

Implement Simple Firewall Using by LKM + Netfilter

■ simple_firewall.c (Skeleton Code)

```
        else
            return NF_ACCEPT;
    }

int setFilter(void)
{
    printk(KERN_INFO "Filter Registered\n");
    nfho.hook = hook_func;
    nfho.hooknum = ...;
    nfho.pf = PF_INET;
    nfho.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &nfho);

    return 0;
}

void removeFilter(void)
{
    printk(KERN_INFO "Filter is being removed\n");
    nf_unregister_net_hook(&init_net, &nfho);
}

module_init(setFilter);
module_exit(removeFilter);
```

Implement Simple Firewall Using by LKM + Netfilter

■ 직접 구현한 **firewall**에 다음 다섯 가지 **rule**을 적용할 것

1. **VM_A**에서 **VM_B**로 **telnet** 연결을 허용하지 않음
2. **VM_B**에서 **VM_A**로 **telnet** 연결을 허용하지 않음
3. **VM_A**에서 학교 **Web Site**에 접속을 허용하지 않음
4. **VM_A**에서 **VM_B**로 **ping**을 허용하지 않음
5. **VM_A**에서 **VM_B**로 **ssh**를 허용하지 않음

■ **Code** 작성 후 **rule** 적용 및 해제

- **make (rule module 생성)**
- **sudo insmod simple_firewall.ko (rule module 적용)**
- **sudo rmmod simple_firewall.ko (rule module 해제)**

Implement Simple Firewall Using by LKM + Netfilter

■ 각 **Rule**을 적용하며 다음 실습 내용을 **Capture**

- 위 **Rule**을 적용하기 위해 작성한 **Code**
- **Rule** 설정이 되기 전과 후의 **telnet** 연결 여부 확인
- **Rule** 설정이 되기 전과 후의 **w3m**을 통한 학교 **Web Site** 접속 여부 확인
- **Rule** 설정이 되기 전과 후의 **ping** 전송 여부 확인
- **Rule** 설정이 되기 전과 후의 **ssh** 연결 여부 확인
- **Putty**를 통해 **Firewall**이 설치된 **VM**으로 접속하여 **dmesg** 명령어를 통해 **kernel** 메시지를 확인할 것

Implement Simple Firewall Using by LKM + Netfilter

■ Hint

- 필요에 따라 **skbuff.h, ip.h, icmp.h, tcp.h, udp.h, netfilter.h**와 같은 별도 **Header File**이 필요할 수도 있음
- **IP Header** 구조체

```
struct iphdr *ip_header = (struct iphdr *)skb_network_header(skb);
unsigned int src_ip = (unsigned int)ip_header->saddr;
unsigned int dest_ip = (unsigned int)ip_header->daddr;
```

- **TCP/UDP Header** 구조체

```
struct udphdr *udp_header = (struct udphdr *)skb_transport_header(skb);
src_port = (unsigned int)ntohs(udp_header->source);
dest_port = (unsigned int)ntohs(udp_header->dest);
```

- **ntohs() : unsigned short int를 network → host byte order로 변경**
- **htons() : unsigned short int를 host → network byte order로 변경**

Implement Simple Firewall Using by LKM + Netfilter

■ Hint

- **Network & Host Byte Order**
- **Network Byte Order = Big-Endian**
- **Host Byte Order = Little-Endian**

Big-Endian

→ 메모리 주소 증가

메모리 주소	0x100	0x101	0x102	0x103	...
변수 값		0x12	0x34	0x56	0x78	

Little-Endian

→ 메모리 주소 증가

메모리 주소	0x100	0x101	0x102	0x103	...
변수 값		0x78	0x56	0x34	0x12	

Implement Simple Firewall Using by LKM + Netfilter

■ Hint

- **IP Address(String)**을 다른 **Format**으로 바꿔주는 **Library Function**

```
int inet_aton(const char *cp, struct in_addr *inp);  
in_addr_t inet_addr(const char *cp);  
in_addr_t inet_network(const char *cp);  
char *inet_ntoa(struct in_addr in);  
struct in_addr inet_makeaddr(int net, int host);  
in_addr_t inet_lnaof(struct in_addr in);  
in_addr_t inet_netof(struct in_addr in);
```

- e.g, “192.168.10.131”이라는 **String**을 **Network** 또는 **Host Byte Order**로 변경할 때 사용

LAB QUESTION

Lab Question

1.방화벽의 종류는 세대 별로 나뉘었을 때 크게 세 가지로 나뉩니다. 각 방화벽의 특징 및 장/단점에 대해서 설명하세요.

Evaluation

■ Lab Task 진행

- 2개의 **Task**에서 진행한 과정을 캡처하고 설명할 것
- **Task 1 결과 : 3개, Task 2 결과 : 5개**

■ Lab Question

- 주어진 문항에 대한 답과 해결 방안에 대해 간략하게 서술

■ Lab Task 수행 결과를 위와 같이 명시한 대로 캡처하여 **MS Word** 또는 **PDF** 파일로 결과를 제출할 것.

- 파일 형식 준수하지 않을 시 감점

Evaluation

- 과제 제출 기한 : 2019/12/09 23:59
- 과제 제출 시 메일 제목 및 파일명은 ‘본인 이름_학번’으로 제출
 - 예) 이석원_2019101059
 - 지연 제출의 경우 메일 제목 앞에 [지연제출]이라고 명시할 것
- sevenshards00@gmail.com으로 보낼 것.

Q&A