리눅스로 **Bridge Firewall** 만들기

이재홍

pyrasis (at) chol.com

리눅스로 Bridge Firewall을 만들어 보자

고친 과정

고침 0.12003-06-28고친이 pyrasis
최초 작성

1. 서문

네트워크 구조를 변경하지 않고 쉽게 방화벽을 설치할 수 있는 Bridge Firewall을 만들어서 사용하자.

본인은 방화벽을 만들면서 수많은 시행 착오를 겪었고 많은 문서들을 보아 왔다. 하지만 워낙 네트워크 상황이 다양하다 보니 문서대로 잘 되지 않는 경우도 많았고 환경도 많이 다른 것도 많았다. 이 문서를 보고 한번에 알맞는 방화벽을 만든다고는 장담 할 수 없다. 하지만 조금이나마 처음 해보는 사람들에게 도움이 되고자 이 문서를 작성 한다.

이 문서는 보호할 네트워크는 공인 IP를 사용한다는 전제 하에 만들어진 문서이다. 본인이 NAT를 사용해보지 않아 NAT에 관한 문서는 차후 준비하겠다.

1.1. 저작권 정보

1.2. 책임의 한계

본 저자는 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 본 문서에서 내포하고 있는 정보들 및 예제들은 여러분이 알아서 활용하십시오. 비록 최선을 다 했으나 이 문서는 틀린 점이나 오류가 있을 수도 있습니다. 만약 여러분이 틀린 점을 발견했다면 꼭 저에게 알려 주시기 바랍니다.

1.3. 감사의 글

이 문서를 작성하는데 도움을 주신 많은 분들에게 감사드립니다.

KLDP의 네트워킹/방화벽 디렉토리의 글들의 도움을 많이 받았습니다. 그곳의 글을 써주시고 변역해주신 모든 분들께 진심으로 감사 드립니다.

1.4. 피드백

이 문서에 대한 발전적인 제안이나 수정사항, 문제점 등에 대한 피드백은 언제든지 환영합니다. `<pyrasis (at) chol.com>`로 메일을 보내 주십시오.

2. 설치 준비 작업

설치를 위해 준비해야 할 것들.

2.1. 네트워크 구성

방화벽이 없는 네트워크 구성

```
      라우터 --------- 스위칭 허브 ----------- PC
                                |
                                ----------- 서버
```

방화벽이 설치 될 네트워크 구성

```
     라우터 ------------- eth0-(Bridge Firewall)-eth1 --- 스위칭 허
브 -------- PC
            (Cross Cable)                                       |
                                                                ---
----- 서
버
```

2.2. 준비할 것들

본인이 Bridge Firewall 을 설치한 리눅스 배포판은 데비안 리눅스 3.0 r1(2003년 6월)이며 커널 버전은 2.4.19이다. 다른 배포판에서도 무리 없이 설치할 수 있다.

테스트 해본 하드웨어 사양은 셀러론 1GHz 256RAM 이였고 현재 50대 정도의 컴퓨터가 방화벽 뒤에서 인터넷을 사용하고 있다. 사용하고 있는 랜카드는 3Com 3c590, Intel EtherExpress/100이다.

꼭 필요한 것들

- 리눅스 커널 2.4.19
- 랜카드 2장
- 크로스 케이블, 다이렉트 케이블
- bridge 커널 패치
- bridge utils
- iptables 커널 패치
- iptables 소스

2.3. 사용 할 각각의 파일들 구하기

bridge 커널 패치 및 bridge utils

Linux ethernet bridging http://bridge.sourceforge.net

`bridge-nf-0.0.7-against-2.4.19.diff`

`bridge-utils-0.9.6.tar.gz`

iptables 커널 패치 및 iptables

netfilter/iptables http://www.netfilter.org

`patch-o-matic-20030107.tar.bz2`

`iptables-1.2.8.tar.bz2`

리눅스 커널 소스

The Linux Kernel Archives http://www.kernel.org

`linux-2.4.19.tar.bz2`

위의 파일들을 `/root`에 받는다.

3. 설치하기

리눅스 커널에 iptables와 bridge패치를 하고 컴파일을 할 것이다. 모든 작업은 root로 한다.

3.1. 리눅스 커널 패치

커널 소스는 `/usr/src/linux` 에 풀어 놓는다.

```
# mv linux-2.4.19.tar.bz2 /usr/src
#cd /usr/src
/usr/src# tar vjxf linux-2.4.19.tar.bz2           압축을 푼다. bunzip2 패키
지가 필요하다.
/usr/src# ln -s linux-2.4.19 linux                압축이 풀린 디렉토리
를 linux라는 이름으로 심볼릭 링크를 생성한다.
```

bridge 패치를 패치한다.

```
# mv bridge-nf-0.0.7-against-2.4.19.diff /usr/src
# cd /usr/src/linux
/usr/src/linux# patch -p1 < ../bridge-nf-0.0.7-against-2.4.19.diff
```

iptable 패치는 patch-o-matic 이라는 방식으로 커널 패치를 한다.

```
# tar vjxf patch-o-matic-20030107.tar.bz2
# cd cd patch-o-matic-20030107
# ./runme extra
Hey! KERNEL_DIR is not set.
Where is your kernel? [/usr/src/linux]            커널 소스를 /usr/src에 풀
어서 linux라고 링크를 걸었다면 엔터를 친다
                                                  아니면 실제 커널이 있는 경
로를 적어준다
다음과 같이 나올 것이다.
Welcome to Rusty's Patch-o-matic!

Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so I don't recommend applying them all!
-------------------------------------------------------
Already applied: submitted/01_2.4.19

Testing... 02_2.4.20.patch NOT APPLIED ( 17 missing files)
The submitted/02_2.4.20 patch:
   Authors: Various (see below)
   Status: Included in stock 2.4.20 kernel

   This big patch contains all netfilter/iptables changes be-
tween stock kernel
   versions 2.4.19 and 2.4.20.

   submitted/DSCP.patch
   + New DSCP target to mangle table (Harald Welte + Matthew G. Marsh)
   submitted/ECN.patch
   + New ECN target to mangle table (Harald Welte)
   submitted/REJECT_mark.patch
```

```
   + Don't copy nfmark value of old packet (Henrik Nordstrom)
   submitted/ahesp-static.patch
   + Fix static build of ahesp match (Paul P Komkoff Jr)
   submitted/conntrack+nat-helper-unregister.patch
   + Fix helper unregister in case of clashing ports (Harald Welte)
   submitted/conntrack.patch
   + Add new 'conntrack' match (Marc Boucher)
   submitted/dscp.patch
   + New 'dscp' match (Harald Welte)
   submitted/ecn.patch
   + New 'ecn' match (Harald Welte)
   submitted/helper.patch
   + New 'helper' match (Martin Josefsson, Harald Welte)
   submitted/ip6tables-exthdr-bug.patch.ipv6
   + Fix broken ipv6 extensionheader parser (Andras Kis-Szabo)
   submitted/ipv6-agr.patch.ipv6
   + New ip6tables 'eui64' match (Andras Kis-Szabo)
   submitted/length.patch.ipv6
   + New ip6tables 'length' match (Imran Patel, James Morris)
   submitted/log-tunnel-fix.patch.ipv6
   + Fix ip6tables 'LOG' target MAC address in case of tunnels
   (Peter Bieringer, Andras Kis-Szabo)
   submitted/nat-memoryleak-fix.patch
   + Fix memoryleak at iptable_nat unload time (zhongyu)
   submitted/ownercmd.patch
   + Extend 'owner' match to match cmdline (Marc Boucher)
   submitted/pkttype.patch
   + New 'pkttype' match (Michal Ludvig)
   submitted/ulog-nlgroup-shift-fix.patch
   + Fix error with shifting nlgroup in ULOG target (Harald Welte)
   submitted/ulog-sparc-bitops-fix.patch
   + Include linux/bitops.h instead of asm/bitops.h
   submitted/z-newnat16.patch
   + Redesign of conntrack and nat helper frame-
work, for more info see http://cvs.netfilter.org/cgi-
bin/cvsweb/netfilter/documentation/newnat-summary.txt
   (Harald Welte, Jozsef Kadlecsik, and others)
   submitted/z-newnat_assertfix.patch
   + Fix erroneously printed ASSERT messages when debugging of newnat
   enabled (Martin Josefsson)
   submitted/z-newnat_changeexpect-lockfix.patch
   + Fix locking bug in ip_conntrack_change_expect() (Martin Josefsson)
   Further changes, not previuosly in patch-o-matic:
   + ip6tables usage counter fix (Harald Welte)
   + ip_queue cleanup (James Morris)
   + minor spelling fixes
   + __constant_htons() macro changes
   + ipt_unclean: srcport _can_ be zero
   + yet another ipchains GFP_ATOMIC fix
```

```
-----------------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/v/q/?]
엔터를 친다
```

iptable패치에는 내용이 여러 가지가 존재한다. y를 누르면 패치가 된다. 그러나 전부 패치를 하면 안 된다. 나중에 커널을 컴파일 하면 에러가 나기 때문에 꼭 필요한 것만 y를 눌러 패치 한다. b 를 누르면 뒤로 돌아갈 수 있다.

지금 패치 할 것은 패킷에서 String을 검색하여 패킷을 버리거나 거부 하는 **String match support** 이다. 이것을 이용하면 Nimda, CodeRed 등의 웜이나 바이러스의 패킷을 차단 할 수 있다.

계속 엔터를 쳐서 가다 보면 아래와 같은 화면이 나올것 이다.

```
Welcome to Rusty's Patch-o-matic!

Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so I don't recommend applying them all!
-------------------------------------------------------
Already applied: submitted/01_2.4.19

Testing... string.patch NOT APPLIED ( 2 missing files)
The extra/string patch:
   Author: Emmanuel Roger <winfield@freegates.be>
   Status: Working, not with kernel 2.4.9

   This patch adds CONFIG_IP_NF_MATCH_STRING which allows you to
   match a string in a whole packet.

   THIS PATCH DOES NOT WORK WITH KERNEL 2.4.9 !!!

-----------------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/v/q/?]

여기서 y를 누르면 String match가 패치 된다.

계속 엔터를 치면 다른 패치들이 나온다. 필요한 것들이 있으면 패치한다.
```

3.2. 커널 옵션 설정하기

iptables 와 bridge기능을 사용할 수 있게 커널 옵션을 설정한다. 사용 할 랜카드 2개도 설정을 할 것이다.

커널 옵션은 다음과 같이 설정한다. menuconfig 를 이용하려면 데비안에서는 libncurses5-dev 패키지가 꼭 필요하다. 데비안에서 패키지는 apt-get 나 dselect를 이용하여 설치한다. 다른 배포판도 마찬가지 이다. 대부분의 배포판에서는 기본적으로 설치되어 있을 것이다.

```
# cd /usr/src/linux
/usr/src/linux# make menuconfig
```

랜카드 드라이버 설정. 본인이 사용하고 있는 랜카드는 3Com 590, Intel EtherPress/100 이다. 각자 가지고 있는 랜카드를 설정하자.

```
Network device support  --->
  Ethernet (10 or 100Mbit)  --->
    [*] Ethernet (10 or 100Mbit)
    < >   Sun Happy Meal 10/100baseT support
    < >   Sun GEM support
    [*]   3COM cards
    < >     3c501 "EtherLink" support
    < >     3c503 "EtherLink II" support
    < >     3c505 "EtherLink Plus" support
    < >     3c507 "EtherLink 16" support (EXPERIMENTAL)
    < >     3c509/3c529 (MCA)/3c579 "EtherLink III" support
    < >     3c515 ISA "Fast EtherLink"
    <*>     3c590/3c900 series (592/595/597) "Vortex/Boomerang" support
    < >   AMD LANCE and PCnet (AT1500 and NE2100) support
    [ ]   Western Digital/SMC cards
    [ ]   Racal-Interlan (Micom) NI cards
    < >   AT1700/1720 support (EXPERIMENTAL)
    < >   DEPCA, DE10x, DE200, DE201, DE202, DE422 support
    < >   HP 10/100VG PCLAN (ISA, EISA, PCI) support
    [ ]   Other ISA cards
    [*]   EISA, VLB, PCI and on board controllers
    < >     AMD PCnet32 PCI support
    < >     Adaptec Starfire/DuraLAN support
    < >     Ansel Communications EISA 3200 support (EXPERIMENTAL)
    < >     Apricot Xen-II on board Ethernet
    < >     CS89x0 support
    < >     DECchip Tulip (dc21x4x) PCI support
    < >     TOSHIBA TC35815 Ethernet support
    < >     Generic DECchip & DIGITAL EtherWORKS PCI/EISA
    < >     Digi Intl. RightSwitch SE-X support
    < >     Davicom DM910x/DM980x support
    <*>     EtherExpressPro/100 support
    < >     Myson MTD-8xx PCI Ethernet support
    < >     National Semiconductor DP8381x series PCI Ethernet support
    < >     PCI NE2000 and clones support (see help)
    < >     RealTek RTL-
8139 C+ PCI Fast Ethernet Adapter support (EXPERIMENTAL)
    < >     RealTek RTL-8139 PCI Fast Ethernet Adapter support
    < >     SiS 900/7016 PCI Fast Ethernet Adapter support
    < >     SMC EtherPower II
    < >     Sundance Alta support
    < >     TI ThunderLAN support
    < >     VIA Rhine support
```

```
    < >     Winbond W89c840 Ethernet support
    [ ]   Pocket and portable adapters
```

bridge 기능을 사용하기 위한 설정

```
Networking options  --->
  <*> Packet socket
    [ ]   Packet socket: mmapped IO
    < > Netlink device emulation
    [*] Network packet filtering (replaces ipchains)
    [ ]   Network packet filtering debugging
    [*] Socket Filtering
    <*> Unix domain sockets
    [*] TCP/IP networking
    [*]   IP: multicasting
    [ ]   IP: advanced router
    [ ]   IP: kernel level autoconfiguration
    < >   IP: tunneling
    < >   IP: GRE tunnels over IP
    [ ]   IP: multicast routing
    [ ]   IP: ARP daemon support (EXPERIMENTAL)
    [ ]   IP: TCP Explicit Congestion Notification support
    [ ]   IP: TCP syncookie support (disabled per default)
      IP: Netfilter Configuration  --->
    < >   The IPv6 protocol (EXPERIMENTAL)
    < >   Kernel httpd acceleration (EXPERIMENTAL)
    [ ] Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
    < > 802.1Q VLAN Support
    ---
    < > The IPX protocol
    < > Appletalk protocol support
    Appletalk devices  --->
    < > DECnet Support
    <*> 802.1d Ethernet Bridging
    [*]   netfilter (firewalling) support
    < > CCITT X.25 Packet Layer (EXPERIMENTAL)
    < > LAPB Data Link Driver (EXPERIMENTAL)
    [ ] 802.2 LLC (EXPERIMENTAL)
    [ ] Frame Diverter (EXPERIMENTAL)
    < > Acorn Econet/AUN protocols (EXPERIMENTAL)
    < > WAN router
    [ ] Fast switching (read help!)
    [ ] Forwarding between high speed interfaces
    QoS and/or fair queueing  --->
    Network testing  --->
```

iptables에서 사용 할 각각의 기능들을 모듈로 설정한다. 모듈 설정은 스페이스 바로 두 번 선택하면 M이라고 표시된다. 이것이 모듈설정이다. *로 설정을 하면 커널로 완전히 포함된다. 모듈로 설정하면 꼭 필요한 것만 쓸 수 있게 하므로 메모리 낭비를 막을 수 있다.

```
Networking options  --->
  IP: Netfilter Configuration  --->
    <M> Connection tracking (required for masq/NAT)
    <M>   FTP protocol support
    <M>   IRC protocol support
    <M> Userspace queueing via NETLINK (EXPERIMENTAL)
    <*> IP tables support (required for filtering/masq/NAT)
    <M>   limit match support
    <M>   MAC address match support
    <M>   netfilter MARK match support
    <M>   Multiple port match support
    <M>   TOS match support
    <M>   AH/ESP match support
    <M>   LENGTH match support
    <M>   TTL match support
    <M>   tcpmss match support
    <M>   Connection state match support
    <M>   Unclean match support (EXPERIMENTAL)
    <M>   String match support (EXPERIMENTAL)
    <M>   Owner match support (EXPERIMENTAL)
    <M>   Packet filtering
    <M>     REJECT target support
    <M>     MIRROR target support (EXPERIMENTAL)
    <M>   Full NAT
    <M>     MASQUERADE target support
    <M>     REDIRECT target support
    [*]     NAT of local connections (READ HELP)
    <M>     Basic SNMP-ALG support (EXPERIMENTAL)
```

3.3. 커널 컴파일

이제 패치한 커널을 컴파일 할 차례 이다.

리눅스 배포판이 데비안일 경우 커널 패키지를 만들어서 커널을 설치하는 것이 편하다. kernel-package 라는 패키지가 필요하다.

```
# cd /usr/src/linux
/usr/src/linux# make-kpkg --revision=1.0 binary-arch              이렇게 하
면 /usr/src에 커널 헤더와 커널 이미지 패키지가 생성된다.
/usr/src/linux# cd ..
/usr/src# ls
-rw-r--r--    1 root     root        30158 Mar 27 20:39 bridge-nf-0.0.7-
against-2.4.19.diff
```

```
-rw-r--r--    1 root     src       3961230 Apr  9 22:58 kernel-headers-
2.4.19_1.0_i386.deb            커널 헤
더
-rw-r--r--    1 root     src       1274482 Apr  9 22:58 kernel-image-
2.4.19_1.0_i386.deb              커널 이미
지
lrwxrwxrwx    1 root     src            12 May 14 04:24 linux -> linux-
2.4.19
drwxr-xr-x   15 573      573           888 Jun 29 06:38 linux-2.4.19
/usr/src# dpkg -i kernel-headers-2.4.19_1.0_i386.deb
/usr/src# dpkg -i kernel-image-2.4.19_1.0_i386.deb

커널 이미지를 설치 할 때 부팅 디스크를 만들라는 곳에는 N으로 취소를 하고 /vmlinuz 라
고 링크를 만든다고 할 때는 Y를 눌러
링크를 만든다

부트로더로 lilo를 사용할 경우
# lilo
Grub를 사용할 경우 /boot/grub/menu.lst 파일을 수정해서 새 커널로 부팅 될 수 있도
록 한다
```

데비안이 아닌 리눅스 배포판의 경우

```
# cd /usr/src/linux
/usr/src/linux# make dep && make bzImage && make mod-
ules && make modules_install
/usr/src/linux# cd arch/i386/boot/
/usr/src/linux/arch/i386/boot# cp bzImage /boot/vmlinuz-2.4.19

부트로더를 lilo를 사용할 경우
/etc/lilo.conf 설정을 변경 한다. 아래와 같은 부분이 있을 것이다.
image=/boot/vmlinuz-2.4.19
        label=Linux
저장을 하고
# lilo

Grub를 사용할 경우
/boot/grub/menu.lst 아래와 같은 부분이 있는데 컴파일한 커널 이미지의 경로를 적어준
다.
title Linux
root (hd0,1) ¶
kernel /boot/vmlinuz-2.4.19 root=/dev/hda2
```

3.4. Bridge utils 설치

bridge 기능을 사용하기 위한 프로그램 설치

bridge utils 소스 컴파일 및 설치

```
# tar vxzf bridge-utils-0.9.6.tar.gz
# cd bridge-utils
~/bridge-utils# ./configure
~/bridge-utils# make
~/bridge-utils# make install
```

3.5. iptables 설치

패킷제어 명령을 내리기위한 iptables설치

iptables 소스 컴파일 및 설치, KERNEL_DIR=/usr/src/linux 에는 커널 소스가 있는 경로를 적어준다.

```
# tar vjxf iptables-1.2.8.tar.bz2
# cd iptables-1.2.8
~/iptables-1.2.8# make KERNEL_DIR=/usr/src/linux
~/iptables-1.2.8# make install KERNEL_DIR=/usr/src/linux
~/iptables-1.2.8# make install-devel
```

4. 실행 스크립트 및 세부 설정

4.1. 패킷에 대한 기본적인 이해

패킷(packet)이란 한글로 번역하자면 소포, 편지묶음 이라는 뜻을 가지고 있다. 그렇다면 패킷이란 도대체 무었인가.

전화 통신을 예로 들어 보겠다. 지금의 아날로그 전화는 회선 방식이다. 우리가 전화를 걸면 교환기를 통해 다른 집의 전화로 연결된다. 전화를 받든 안 받든 일단 연결을 한다. 그렇다면 연결 된 회선을 점유하고 있다는 것 이다. 전화를 받지 않아 통화를 하지 않는다 해도 연결이 되었기 때문에 회선을 쓰고 있는 것이 된다.

그런데 이것을 인터넷에 적용시킨다면 엄청나게 비효율적으로 된다. 그래서 인터넷에서는 패킷을 사용한다. 패킷은 작은 데이터의 덩어리 이다. FTP를 통해서 파일을 받는다고 할 때. 우리 눈으로 볼때는 파일이 한꺼번에 받아진다. 하지만 실제로 파일들은 수백 수천개의 패킷들로 조각나 우리에게로 전송되는 것이다. 컴퓨터는 그 패킷을 받아 조립하여 다시 파일을 만들어 낸다.

우리가 인터넷을 한다면 웹 브라우저로 웹서핑도 하고 FTP로 파일도 받고 메신저로 메시지도 주고 받을 것이다. 우리 컴퓨터에 연결된 랜선에는 웹 브라우저에서 요청한 HTML파일의 패킷, FTP에서 전송하고 있는 파일 패킷들, 메신저에서 주고 받는 메시지 패킷들이 서로 뒤섞

여 오고 간다. 이렇게 섞여서 오고 간다해도 패킷들에는 각각의 정보가 존재 한다. 그래서 패킷이 섞이지 않고 여러가지 일을 동시에 할 수 있는 것이다. 전화 같으면 회선을 완전 점유해서 사용하기 때문에 한가지 일 밖에 할 수가 없다.

이제 방화벽의 원리를 알아보자. 방화벽은 패킷을 구분하여 그 패킷을 통과 시킬 것인지 아니면 거부, 혹은 버릴 것인지를 결정하는것이다. iptables를 통해서 패킷을 어떻게 제어 하느냐에 따라서 접속을 막거나 혹은 열거나, 스트링 검색을 통해서 웜의 패킷을 막을 수 있는 것이다.

4.2. bridge 및 iptables 스크립트 작성

iptables는 명령을 내려줘도 메모리에만 올라가 있기 때문에 재부팅을 하면 설정 내용이 사라진다. 따라서 부팅할 때 실행되도록 스크립트를 만들어 줘야 한다.

iptables 명령 스크립트는 인터넷 상에 많은 예제들이 올라와 있다. 본인도 처음에는 그대로 복사해서 사용을 했지만. 내용을 이해하지 않고 사용을 해서 제대로 작동하지 않았다. 가장 중요한 부분은 패킷에 대한 이해와 프로토콜에 대한 기본적인 지식을 갖추어야 한다는 것이다. 지금 만든 방화벽이 설치되는 네트워크가 본인이 사용하고 있는 네트워크하고 똑같다고 할 수는 없을 것이다, 그러므로 자신의 네트워크 상황에 맞게 스크립트를 재작성 해야 할 것이다.

아래 스크립트는 부팅될 때 실행될 수 있도록 /etc/init.d/bridgefirewall 로 저장한다. 스크립트의 주석을 보고 각자 네트워크 상황에 맞게 설정하기 바란다. 네트워크 설정은 자신의 네트워크에 맞게 설정하기 바란다. 앞으로 예를 들 네트워크 구성은다음과 같다. 물론 존재하지 않는 IP이다. 방화벽과 방화벽 뒤에서 보호받는 네트워크 모두 공인 IP를 사용한다.

- 방화벽 자체 IP : 234.234.200.10
- 브로드캐스트 : 234.234.200.255
- 사용중인 IP : 234.234.200.0 ~ 234.234.200.255

```
#!/bin/sh

BR_IP="234.234.200.10"
# 방화벽의 자체 IP. 원래 Bridge firewall은 IP 주소가 없다. 하지만 IP를
# 할당하지 않으면 원격에서 접속할 수 없고 직접 방화벽 머신 앞에 앉아서
# 작업을 해야 하기 때문에 관리하기가 불편하다. 그래서 관리를 위해
# IP를 할당한다. 내부네트워크 에서만 접속이 가능하도록 설정 할 것이다.
# 보안상 걱정이 된다면 IP주소를 할당하지 않아도 된다. 그러면 원격에서는
# 절대 접속할 수 없다.

BR_IFACE="inca-br"     # 브리지 이름, 마음에 드는 이름을 준다.

LAN_BCAST_ADDRESS="234.234.200.255"       # 브로드 캐스트
INTERNAL_ADDRESS_RANGE="234.234.200.0/24" # 네트워크 범위.
INTERNAL_ADDRESS="255.255.255.0"          # 넷 마스크

INET_IFACE="eth0" # 외부에서 들어오는 선을 연결 할 랜카드
```

```
LAN_IFACE="eth1"  # 내부로 나가는 선을 연결하는 랜카드

LO_IFACE="lo"
LO_IP="127.0.0.1"

IPTABLES="/usr/local/sbin/iptables" # iptables 의 절대 경로

#########
/sbin/depmod -a

# 아래 부분은 iptables에서 사용 할 모듈을 적재하는 작업이다.
# 데비안 리눅스의 경우 modconf에서 적재할 모듈을 선택을 하면
# 부팅 할 때마다 아래와 같이 적어줄 필요는 없다.
# 다른 배포판의 경우 아래와 같이 필요한 모듈들을 올려줘야 iptables에서
# 각각의 명령들이 인식 될 것이다.
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/iptable_filter.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_conntrack.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/iptable_nat.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_conntrack_ftp.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_conntrack_irc.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ipt_string.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ipt_state.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ipt_REJECT.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ipt_REDIRECT.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ipt_MASQUERADE.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_nat_snmp_basic.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_nat_ftp.o
/sbin/insmod /lib/modules/2.4.19/kernel/net/ipv4/netfilter/ip_nat_irc.o

ifconfig $INET_IFACE down   # 모든 랜카드의 설정을 초기화
ifconfig $LAN_IFACE down
ifconfig $BR_IFACE down

ifconfig $INET_IFACE 0.0.0.0  # 브리지가 될 랜카드에 0.0.0.0의 아이피를 준다.
ifconfig $LAN_IFACE 0.0.0.0

$IPTABLES -F  # 체인의 규칙들을 지운다.
$IPTABLES -X  # 체인을 지운다, 초기화 작업.

# 초기 정책.
# INPUT(들어오는 것) DROP(버린다)
# OUTPUT(나가는 것) ACCEPT(허용), INPUT OUTPUT은 방화벽 자체에서 오고 나가는 패킷
을 뜻함.
# FORWARD DROP : 가장 중요한 부분이다.
# 브리지 방화벽을 통해 지나가는 모든 패킷은 FORWARD에서 제어한다.
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD DROP
```

```
# 브리지 설정
/usr/local/sbin/brctl addbr $BR_IFACE
/usr/local/sbin/brctl addif $BR_IFACE $INET_IFACE
/usr/local/sbin/brctl addif $BR_IFACE $LAN_IFACE

# 브리지 IP 설정
if [ "$BR_IP" != "" ] ; then
    ifconfig $BR_IFACE $BR_IP broadcast $LAN_BCAST_ADDRESS net-
mask $INTERNAL_ADDRESS
    else
        ifconfig $BR_IFACE up
fi

# Firewall SSH
# 방화벽 컴퓨터 자체의 설정.
# 234.234.200.0 부터 255 까지 아이피에서 방화벽 22번 포트로 접속하는 것을 허용
# 따라서 회사내부에서만 접속이 가능하며 SSH로만 접속이 가능하다.
# 관리자의 특정 IP만 허용하려면 234.234.200.0/24대신 234.234.200.12 이런식으로
# 적어주면 된다
$IPTABLES -A INPUT -p tcp -s 234.234.200.0/24 --dport 22 -j ACCEPT

# Deny IP list
# 이것은 예를 든 것이다. 사용하고 싶은 사람은 사용하기 바란다.
# 첫번째는 10.105.4.202에서 오는 icmp 패킷을 버린다는 뜻, tcp, udp를 사용할 수 있
다
# 두번째는 MAC 주소를 차단하는 방법으로 아래의 MAC 주소에서 오는 패킷은 차단된다.
#$IPTABLES -A FORWARD -p icmp -s 10.105.4.202 -j DROP
#$IPTABLES -A FORWARD -m mac --mac-source 00:02:2A:C4:86:17 -j DROP

# Nimda, CodeRed
# 님다 패킷 및 코드레드 패킷 차단하는 명령
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "/default.ida?" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "XXXXXXXX" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "cmd.exe" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "root.exe?" -j REJECT --reject-with tcp-reset

# SQL Slammer
# SQL 슬래머 웜의 패킷을 차단하는 명령
$IPTABLES -A FORWARD -p udp -m string --string "Qh.dllhel32hkern" -
j REJECT

# 80번 포트만 사용하는 웹 서버의 경우 80번포트만 열고 나머지 80포트로 오고 가는 패킷
에 대한
# 스트링 검사를 해서 Nimda, CodeRed의 패킷을 막는다.
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --dport 80 -j ACCEPT
```

```
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "/default.ida?" -j REJECT --reject-with tcp-
reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "XXXXXXXX" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "cmd.exe" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "root.exe?" -j REJECT --reject-with tcp-reset

# MSN
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 1863:1864 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6901 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 7801:7825 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6891:6900 -
j ACCEPT

# KTiman
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10020 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10250 -j ACCEPT

# IRC
$IPTABLES -A FORWARD -p tcp --dport 6667 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 6667 -j ACCEPT

# Remote
$IPTABLES -A FORWARD -p tcp --sport 6009 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 6009 -j ACCEPT

# FTP Client
$IPTABLES -A FORWARD -p tcp --sport 21 -m state --state ESTABLISHED -
j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 20 -m state --
state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024:65535 --dport 1024:65535 -
m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 113 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 113 -j ACCEPT

# syn packet drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 -j DROP

# net send drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 139 -j DROP
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 2603 -j DROP
$IPTABLES -A FORWARD -p udp -d 234.234.200.0/24 --dport 135 -j DROP
```

```
$IPTABLES -A FORWARD -p ALL -j ACCEPT
```

네트워크 범위 설정의 뜻

우리는 앞에서 네트워크 범위 설정이라는 부분을 하였다. 234.234.200.0 부터 255까지를 234.234.200.0/24라고 설정 하였다. 이것이 왜 이렇게 설정되는지 알아보자

표 **1. IP** 주소의 범위 표현

1	1	1	1	1	1	1	1	.	1	1	1	1	1	1	1	1	.	1	1	1	1	1	1	1	1	.	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	.	8	9	10	11	12	13	14	15	.	16	17	18	19	20	21	22	23	.	24	25	26	27	28	29	30	31

위의 표는 255.255.255.255를 2진수로 표현한 것이고 두번째 줄은 각 비트의 순서이다.

234.234.200.0/24라고 할때 234.234.200.0부터 234.234.200.255까지 되는 이유는 0/24 에서 24번째 비트까지를 의미한다.

표 **2. 0/24**

.	1	1	1	1	1	1	1	1
.	24	25	26	27	28	29	30	31

24 번째 비트까지 이면 1 1 1 1 1 1 1 1즉 255다. 0부터 255까지라는 말이다

0/25라고 하면 0 1 1 1 1 1 1 1로 128이 된다. 0부터 128이 된다.

0/26이면 0 0 1 1 1 1 1 1로 0부터 64

128/25라면. 128 부터 255까지 사용한다는 것이다.

4.3. 정책변경

방화벽을 부팅을 해놓고 정책을 수정할 수 있어야 한다. 그러려면 스크립트를 하나 더 만들어 줘야 하는데.. 매우 쉽게 할 수 있다. 위의 부팅 스크립트에서 브리지 설정 부분만 뺀 iptables명령 부분만 잘라서 스크립트로 만들면 스크립트 실행만으로 간단한 정책 변경이 가능하게 된다.

다음은 방화벽이 부팅되고 정책을 변경 할 수 있는 스크립트이다. ipt-sh라고 저장하고 chmod 755 ipt-sh 한뒤 ./ipt-sh 라고 실행하면 된다.

```
IPTABLES="/usr/local/sbin/iptables"

$IPTABLES -F  # 체인의 규칙들을 지운다.
$IPTABLES -X  # 체인을 지운다, 초기화 작업.
```

```
# Firewall SSH
$IPTABLES -A INPUT -p tcp -s 234.234.200.0/24 --dport 22 -j ACCEPT

#$IPTABLES -A FORWARD -p icmp -s 10.105.4.202 -j DROP
#$IPTABLES -A FORWARD -m mac --mac-source 00:02:2A:C4:86:17 -j DROP

# Nimda, CodeRed
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "/default.ida?" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "XXXXXXXX" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "cmd.exe" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp --tcp-flags ACK ACK --dport 80 -m string --
string "root.exe?" -j REJECT --reject-with tcp-reset

# SQL Slammer
$IPTABLES -A FORWARD -p udp -m string --string "Qh.dllhel32hkern" -
j REJECT

# WebServer
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --dport 80 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "/default.ida?" -j REJECT --reject-with tcp-
reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "XXXXXXXX" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "cmd.exe" -j REJECT --reject-with tcp-reset
$IPTABLES -A FORWARD -p tcp -d 234.234.200.5 --tcp-flags ACK ACK --
dport 80 -m string --string "root.exe?" -j REJECT --reject-with tcp-reset

# MSN
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 1863:1864 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6901 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 7801:7825 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6891:6900 -
j ACCEPT

# KTiman
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10020 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10250 -j ACCEPT

# IRC
$IPTABLES -A FORWARD -p tcp --dport 6667 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 6667 -j ACCEPT
```

```
# Remote
$IPTABLES -A FORWARD -p tcp --sport 6009 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 6009 -j ACCEPT

# FTP Client
$IPTABLES -A FORWARD -p tcp --sport 21 -m state --state ESTABLISHED -
j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 20 -m state --
state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024:65535 --dport 1024:65535 -
m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 113 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 113 -j ACCEPT

# syn packet drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 -j DROP

# net send drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 139 -j DROP
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 2603 -j DROP
$IPTABLES -A FORWARD -p udp -d 234.234.200.0/24 --dport 135 -j DROP

$IPTABLES -A FORWARD -p ALL -j ACCEPT

$IPTABLES -L
```

4.4. 기본적인 명령 사용법

iptables에 명령을 내려 어떻게 패킷들을 조작하는지 기본적인 것들을 알아보자.

-A FORWARD : FORWARD체인에 정책 추가, 가장 많이 사용한다.

-A INPUT, -A OUTPUT : INPUT, OUTPUT 체인에 정책 추가, 방화벽 자체의 오고 나가는 패킷정책. 거의 사용하지 않는다.

-p tcp : TCP프로토콜, Web, FTP, Telnet, SSH, 등

-p udp : UDP프로토콜

-p icmp : ICMP프로토콜, PING

-d : Destination IP, 목적지 IP 예) -d 234.234.200.123

-s : Source IP, 발생지 IP 예) -s 234.234.200.123

--dport : Destination Port, 목적지 포트 예) --dport 80 혹은 --dport 80:90 80번부터 90번까지

--sport : Source Port, 발생지 포트 예) --sport 80 혹은 --sport 80:90 80번부터 90번까지

앞에서 우리는 INTERNAL_ADDRESS_RANGE="234.234.200.0/24" 라고 설정했다. 내부 네트워크를 뜻하는데 아래 두줄의 명령은 같다. 단지 INTERNAL_ADDRESS_RANGE라고 변수

를 주고 IP를 대입시켜 주었는데 이해를 돕기 위해 IP를 그대로 넣는 방식으로 하겠다. 변수만 한가득히 나오면 처음 보는 사람은 잘 이해가 가지 않기 때문이다. 처음에는 IP를 그대로 넣고 변수를 사용하고 싶은 사람은 사용해도 상관 없다.

```
$IPTABLES -A FORWARD -p tcp -d $INTERNAL_ADDRESS_RANGE --dport 80 -j DROP
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 80 -j DROP
```

iptables 에서 FORWARD체인을 기본적으로 DROP으로 해놨었다. 그렇게 되면 모두 막히게 된다. 정책을 설정 할 때 열어줄 포트를 먼저 열고 모두 막아야 한다. 모두 막고 열어줄 것을 열면 열리지 않게 된다.

위의 부팅 스크립트에서는 syn 패킷을 드롭 한다. 이 말은 FORWARD체인에 (-A FORWARD) TCP프로토콜 (-p tcp) syn패킷 (--syn) 목적지 IP가 234.234.200.0/24일 때 (-d 234.234.200.0/24) 드롭한다. (-j DROP)

```
# syn packet drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 -j DROP
$IPTABLES -A FORWARD -p ALL -j ACCEPT
```

syn 패킷을 드롭 하는 이유는 234.234.200.0/24라는 내부 네트워크에 TCP 접속을 막기 위해서이다. TCP를 사용하는 telnet, web, FTP등 내부로의 접속을 막는 것이다. syn패킷은 TCP프로토콜의 초기 접속 요구 패킷이다. 목적지 IP가 내부 네트워크로 돼있고 그것을 드롭한다. 내부에서 외부로 나가는 것은 막히지 않는다. 주의할 점은 syn 패킷을 막지 않고 그냥 목적지가 내부 네트워크인 TCP프로토콜을 막는 경우 웹을 포함한 모든 TCP를 사용하는 것들은 통신을 하지 못한다. 외부로 나갈 수는 있지만 그 결과를 내부로 가져 올 수 없기 때문에 통신이 되지 않는 것이다. 두번째 줄의 모든 프로토콜을 허용한다는 명령인데 저것을 하지 않으면 통신이 되지 않는다. 꼭 해주어야 한다. FORWARD의 기본 정책이 DROP이기 때문이다.

특정 포트 열기와 특정 포트 막기

```
# 21번 포트 막기, 목적지IP가 내부, 목적지 포트가 21번인 패킷을 드롭, 효과는 TCP프로
토콜 21번 포트를 사용하는
# 프로그램은 내부에서 통신을 할수 없다. -s 234.234.200.0/24로 하더라도 효과는 같
다. 발생지 IP가 내부이고 목적지가 외부
# 21번 포트 이므로 효과는 같다.
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 21 -j DROP

# 21번 포트 열기, 목적지 IP가 내부, 목적지 포트가 21번인 패킷을 허용, 효과는 외부에
서 내부로 TCP연결을 할 수 있게 된다.
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 21 -j ACCEPT

# 정책 마지막 줄에는 항상 위의 syn패킷 드롭을 해주어야 한다. 그러지 않으면 아무리 열
고 닫는다 해도 FORWARD체인의 기본
# 정책을 DROP로 해놨기 때문에 통신이 안 된다.
```

특정 포트를 막는 것은 회사 같은 곳에서 특정 포트를 사용하는 프로그램(P2P 등)의 사용을 막고자 할 때 사용한다.

특정 포트를 열고자 하는 것은 방화벽 뒤에 메일서버, 웹서버 등이 있을때 그 서버들이 사용하는 포트를 열어 줄때 사용한다.

4.5. 자주 사용되는 것들

방화벽을 사용하다 보면 자주 생기는 일들이 있다. 인스턴트 메신져, FTP등의 설정을 알아보자.

인스턴트 메신져. MSN 등. 각각 메신저의 홈페이지에 사용하는 포트를 표시 해놓고 있다. 그 포트를 보고 알맞게 고치면 된다. 메신저가 사용하는 포트를 열어주지 않았다고 해도 메신저는 동작 할 것이다. 그러나 파일전송 같은 것이 동작이 되지 않는 경우가 발생한다.

```
# MSN
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 1863:1864 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6901 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 7801:7825 -
j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 6891:6900 -
j ACCEPT

# KTiman
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10020 -j ACCEPT
$IPTABLES -A FORWARD -p tcp -d 234.234.200.0/24 --dport 10250 -j ACCEPT
```

FTP클라이언트를 사용할 수 있게 설정

```
# FTP Client
$IPTABLES -A FORWARD -p tcp --sport 21 -m state --state ESTABLISHED -
j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 20 -m state --
state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024:65535 --dport 1024:65535 -
m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 113 -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 113 -j ACCEPT
```

패킷 스트링 검색 String match Support, Nimda, CodeRed Packets

```
#$IPTABLES -A FORWARD -p tcp -d 234.234.200.1 --tcp-flags ACK ACK --
dport 80 -m string --string "/default.ida?" -j REJECT --reject-with tcp-
reset
#$IPTABLES -A FORWARD -p tcp -d 234.234.200.1 --tcp-flags ACK ACK --
dport 80 -m string --string "XXXXXXXX" -j REJECT --reject-with tcp-reset
#$IPTABLES -A FORWARD -p tcp -d 234.234.200.1 --tcp-flags ACK ACK --
dport 80 -m string --string "cmd.exe" -j REJECT --reject-with tcp-reset
```

```
#$IPTABLES -A FORWARD -p tcp -d 234.234.200.1 --tcp-flags ACK ACK --
dport 80 -m string --string "root.exe?" -j REJECT --reject-with tcp-reset
```

위의 명령에서 --string "cmd.exe" 는 패킷 속에 cmd.exe라는 스트링을 포함하고 있으면 차단하거나 거부한다는 것이다 --tcp-flags ACK ACK 는 TCP프로토콜에서 초기 연결하는 패킷을 뜻한다.

net send (팝업 스팸) 윈도우2000이상의 운영체제에 메세지 창이 뜨는 스팸을 막는 방법

```
# net send drop
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 139 -j DROP
$IPTABLES -A FORWARD -p tcp --syn -d 234.234.200.0/24 --sport 2603 -j DROP
$IPTABLES -A FORWARD -p udp -d 234.234.200.0/24 --dport 135 -j DROP
```

이정도로 마치도록 하겠습니다. 처음 리눅스를 접하는 사람은 아무래도 커널 컴파일이나 설정 부분이 어려울 것입니다. 부족한 문서이지만 많은 도움이 되었으면 합니다. 문서의 잘못된 점이 있다면 바로 메일을 보내 주시기 바랍니다.