리눅스를 이용한 방화벽 구축

2005. 5.

인프라보호단 / 보안관리팀

KISA 한국정보보호진흥원
Korea Information Security Agency

목 차

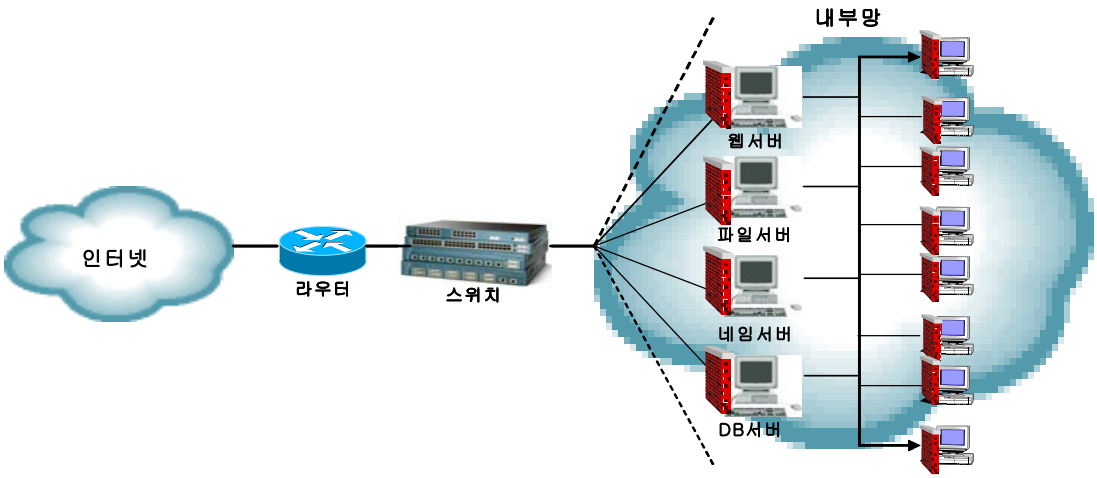
그림 목차

Ⅰ.서 론

건물의 한곳에서 화재가 발생한 후 주변으로 번지는 것을 방지하는 역할을 하는 기능이 방화벽이듯이 전산에서 사용되고 있는 방화벽 (Firewall, 침입차단시스템)의 개념은 인터넷상의 보안사고 및 위협이 주변 망이나 개인 PC로 전파되는 것을 방지하는 역할을 하는 것이다. 이는 특정 포트나 특정 IP에 대한 접근통제를 기본원칙으로 하며, 시스템이나 네트워크 앞단에 설치되어 진다. 보안의 필요성이 강조되면서 방화벽의 개념은 널리 알려져 있고, 이에 대한 활용도 또한 높아지고 있다. 이 장에서는 이러한 방화벽 기능을 상용제품이 아닌 리눅스[1]를 활용하여 구축하는 방법에 대해 알아보기로 한다.

Ⅱ. 리눅스를 이용한 방화벽 구축 방안

리눅스를 이용하여 방화벽의 기능을 구축할 수 있는 세 가지 방법에 대해 알아보겠다. 첫 번째, 리눅스 서버 자체에 방화벽을 탑재한 것으로 시스템 자체에 방화벽 기능이 포함되었기 때문에 각각의 리눅스 시스템 커널에 직접 방화벽을 설치하는 경우이다. 규모가 크지 않고, 개별적인 서비스를 제공하는 시스템이라면 이 방법을 추천한다. 이러한 경우 기존에 서비스중인 서버에 직접 방화벽을 소프트웨어적으로 설치하는 것이므로 별도의 방화벽 장비(시스템)가 필요하지 않다는 장점이 있다.

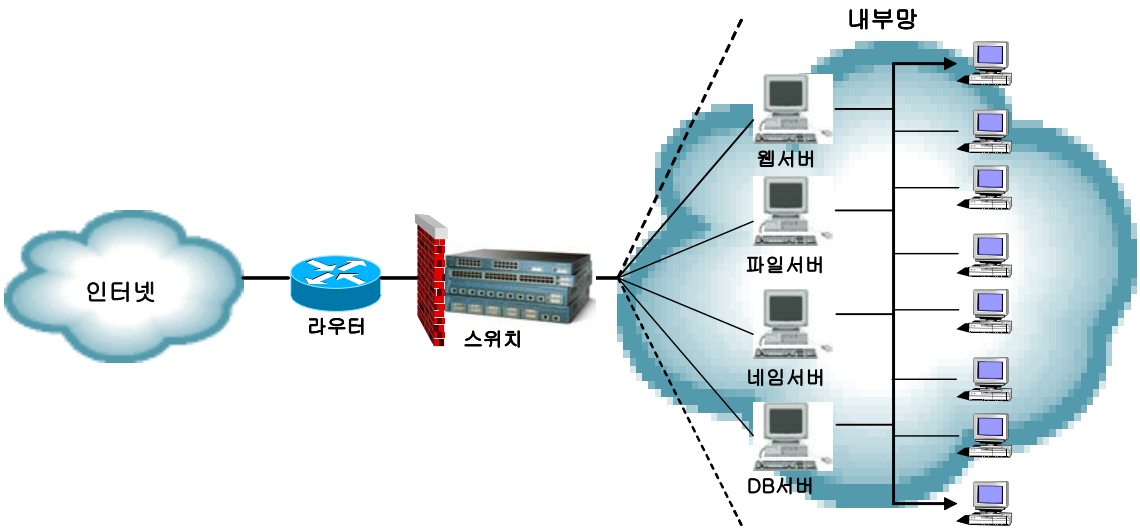


[그림 1] 각 시스템에 방화벽 설치

1) 리눅스 : 리누스토발즈(linus torvalds)에 의해 개발된 운영체제

두 번째, 리눅스 방화벽 시스템이 네트워크 말단에 놓여지는 것으로 **NAT**[2] 또는 브리지 방화벽이 될 수 있다. NAT 는 초기 방화벽에서 자주 사용되었던 방법이고 최근에도 꾸준히 사용되고 있는 방법으로 방화벽이 일종의 게이트웨이 또는 라우터 역할을 하는 경우이다. 외부와 연결되는 방화벽의 인터페이스(eth0)에는 공인 IP를 할당하고 내부의 인터페이스(eth1)에는 사설 IP를 할당함으로써 방화벽에서 공인 IP를 내부의 사설 IP의 특정 포트로 매칭시켜주는 방법이다. 이를테면, 230.1.2.3 의 80번으로 들어오는 패킷을 내부의 192.168.1.3의 80번으로 매칭시켜주는 것이 그 예이다.

브리지 방화벽[3] 역시 형태적으로는 동일하지만 NAT와는 크게 다른 방식으로서, 리눅스 방화벽은 단지 스위치처럼 패킷을 받아서 스위칭만 할 뿐 NAT와 같은 라우팅은 하지 않는다. 따라서 방화벽에는 원격에서 접속할 일이 없다면 굳이 IP를 할당할 필요가 없으며 방화벽 내부의 각 서버는 공인 IP를 그대로 사용하면 된다. 즉, 브리지 방화벽을 통과하는 패킷중 비정상적인 패킷은 필터링하고, 정상적인 트래픽을 허용하는 것은 NAT와 동일하지만 방화벽 자체에서 라우팅을 하는지 여부와 방화벽 내부의 서버들이 공인 IP를 사용하는지 사설 IP를 사용하는지에 따라 달라질 수 있다.



[그림 2] NAT, 브리지 방화벽 구조

2) NAT(Network Address Translation)-내부의 개별주소와 정식 IP주소를 상호 변환하는 기능이다. 개별노드가 할당되지 않은 노드에서도 인터넷에 접속 가능하고, TCP/IP의 전송계층이나 응용계층의 통신규약에 대한 변환을 하여 특정 TCP/IP 응용을 이용하도록 한다.

3) 브리지방화벽-두 동일한 이더넷 환경을 연결시켜줌으로써 오가는 패킷을 그대로 전달한다.

그림상의 위치를 보면 NAT와 브리지방화벽의 구조는 같아 보일 수 있다.
NAT나 브리지든 관계없이 방화벽에 랜카드(인터페이스)는 최소 2장 이상 연결되어 있어야 한다. 하지만 NAT 구조에서는 외부와 연결된 랜카드에는 공인 IP가, 내부의 스위치로 연결된 랜카드에는 사설 IP가 할당되어야 한다. 하지만 브리지 방화벽에서는 두 랜카드에 IP를 할당할 필요가 없다. 브리지 방화벽은 그냥 스위치라고 생각하면 되는 것이다.
그리고 방화벽 내부에서는 NAT 의 경우 모두 사설 IP를 할당하여야 한다. 사설 IP는 네트워크의 규모에 따라 적당히 설정하면 되는데, 통상적으로 192.168.x.x 대역이나 172.16.x.x 또는 10.x.x.x 대역을 주로 사용한다. 그러나 어떤 IP를 사용하든, 네트워크 대역에 관계없이 공통적인 사항은 내부 서버들의 게이트웨이는 반드시 방화벽의 사설 IP 에 할당된 IP를 설정하여야 한다. 즉 방화벽의 내부 랜카드인 eth1이 내부 네트워크의 게이트웨이 역할을 하는 것이다. 그렇지 않으면 내부 네트워크에서는 전혀 통신을 할 수 없게 된다.
반면 브리지 방화벽의 내부에서는 공인 IP를 그대로 사용한다. IP 뿐만 아니라 netmask나 broadcast등도 똑같이 설정하면 된다. 다시 한번 강조하지만 브리지 방화벽은 그냥 스위치라고 생각하면 된다.

그럼, NAT를 사용하여야 할까 아니면 브리지 방화벽을 사용하여야 할까?
각각의 장단점이 있는데, 일단 브리지 방화벽은 NAT에 비해 편리하다. 내부 서버에서도 공인 IP를 그대로 사용할 수 있기 때문에 복잡한 NAT를 고민할 필요가 없다. 그리고 뒤에서 자세히 살펴보겠지만 방화벽 설정에서도 NAT의 경우 공인 IP 및 포트를 사설 IP 로 포워딩해 주거나 반대로 사설 IP를 공인 IP 로 포워딩하는 NAT도 고려해주고, 또한 패킷 필터링도 고려해 주어야 하지만 브리지 방화벽은 공인 IP를 그대로 사용하기 때문에 패킷 필터링만 신경 쓰면 된다. 따라서 규칙도 브리지 방화벽이 훨씬 단순하다.

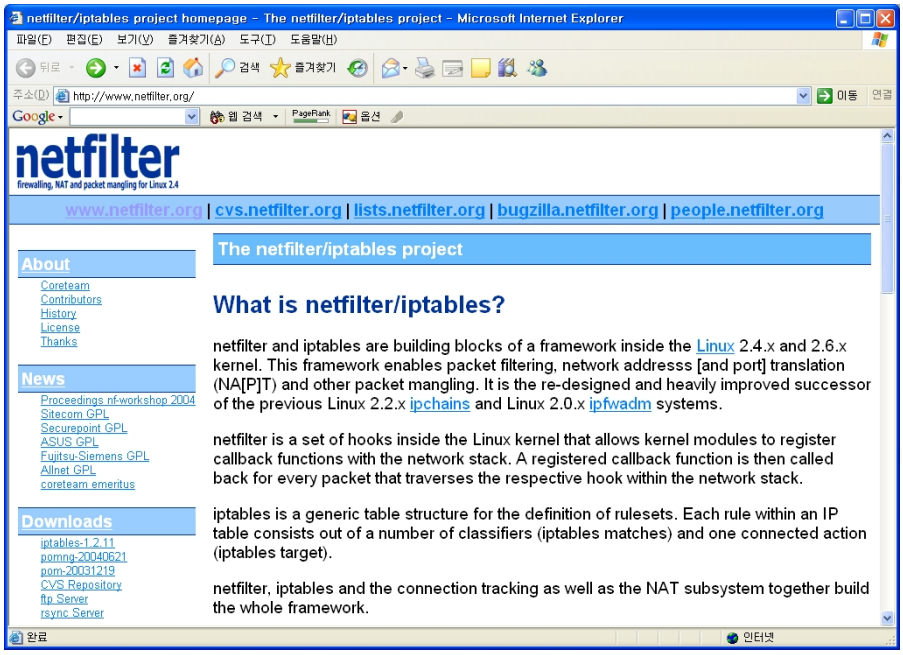
그러나 NAT의 경우 실제 서버에서는 사설 IP를 사용하기 때문에 NAT 방화벽에서 별도로 패킷을 포워딩하고 또한 허용해 주지 않는 한 구조적으로 외부에서 내부로는 절대 접속할 수 없다. 따라서 극도의 보안을 추구한다면 NAT가 좋은 대안이 될 수 있을 것이다. 반면 브리지 방화벽의 경우 물론 규칙 설정을 확실히 하여 꼭 필요한 서비스 외에는 외부에서의 일체 접속을 차단하여야 하지만 만약 실수에 의해 허용되었을 경우 심각한 문제가 유발될 수 있으므로 주의해야한다. 실수로 방화벽의 규칙이 허용되어 있어 방화벽이 있지만 패킷 필터링을 제대로 못할 수 있기 때문이다.
초기에는 NAT가 대세였지만 최근에는 편의성을 고려하여 NAT 보다는 브리지 방화벽이 많이 사용되고 있는 추세이다. 각각의 방법에 대한 구체적인 설정 방법 등에 대해서는 뒤에서 자세히 알아보도록 한다.

1. iptables 설치 및 리눅스 커널 설정하기

각자 위와 같은 구조에 따라 서버자체 또는 NAT나 브리지 방식을 적절히 사용하면 될 것이다. 이제 리눅스 방화벽의 기본이 되는 netfilter 기반의 iptables에 대해 알아보고 이는 어떤 장점과 기능을 제공하는지 알아본다.

⇒ http://www.netfilter.org

⇒ http://www.iptables.org



[그림 3] netfilter 홈페이지

iptables는 다른 상용 방화벽이 제공하는 기능을 대부분 가지고 있는데, 그 중에서 가장 대표적인 기능 또는 이전 버전에 비해 향상된 기능은 다음과 같다.

1) 상태추적 기능 제공

최근의 방화벽에서 제공하는 고급기능중 하나인 상태추적은 방화벽을 통과하는 모든 패킷에 대한 연결 상태를 추적(tracking)하여 이 정보를 메모리에 기억하고 있다가 기존의 연결을 가장하여 접근할 경우 메모리에 저장된 상태 목록과 비교하여 적합하면 통과하고 그렇지 않으면 거부하는 기능으로서 지능화된 공격시도를 차단할 수 있는 기능중 하나이다. 이를테면 단순히 포트나 tcp flag 등을 매칭하여 필터링한 구 버전의 방화벽에서 기존의 연결 없이 단지 ack flag를 설정한 tcp 패킷이 들어온다면 이미 연결중인 트래픽으로 판단하여 허용하지만 상태 추적이 제공될 경우

에는 아무리 ack flag를 달고 들어온다 하더라도 이전의 접속 목록에 syn 및 syn/ack 와 관련된 정보가 없기 때문에 비정상 트래픽으로 간주하여 필터링하게 되는 것이다. iptables 이전 버전인 ipchains 와 같이 상태추적을 제공하지 않는 방화벽은 stateless 라고 하며 지금부터 알아볼 iptables 와 같이 상태추적이 제공되는 방화벽은 stateful 이라고 한다.

2) 향상된 매칭 기능 제공

iptables는 방화벽에서 기본적으로 제공하는 매칭 정보인 패킷의 소스 ip, 목적지 ip 및 소스 포트, 목적지 포트 번호 뿐만 아니라 추가적으로 다양한 매칭 기능을 제공하고 있다. 이를테면 상태 추적을 통한 현재의 연결 상태나 하드웨어 MAC 주소, 패킷 발신자의 유저나 그룹 프로세스, ip 헤더의 TOS(Type Of Service)등 여러 가지 조건을 이용하여 세부적이고 복잡한 매칭 및 필터링이 가능하다. 물론 일부 기능은 추가적으로 커널 패치를 통해 구현할 수 있다.

3) 포트 포워딩(port forwarding) 기능 포함 제공

이전 버전인 ipfw이나 ipchains를 사용할 때는 NAT를 이용하기 위해서 별도로 분리되어 있던 툴인 ipmasqadm을 사용하여야 했으나 iptables에는 NAT 기능이 자체적으로 포함되어 있어 NAT를 위해 별도의 프로그램을 이용할 필요가 없이 커널 메뉴에서 추가로 지정해서 컴파일 해 주면 iptables만으로도 바로 사용할 수 있다.
2.6버전이 아닌 2.4버전의 커널 사용시에는 아래와 같은 커널 설정확인을 거쳐야한다.

STEP 1. KERNEL 설정하기 (버전 2.4 기준)

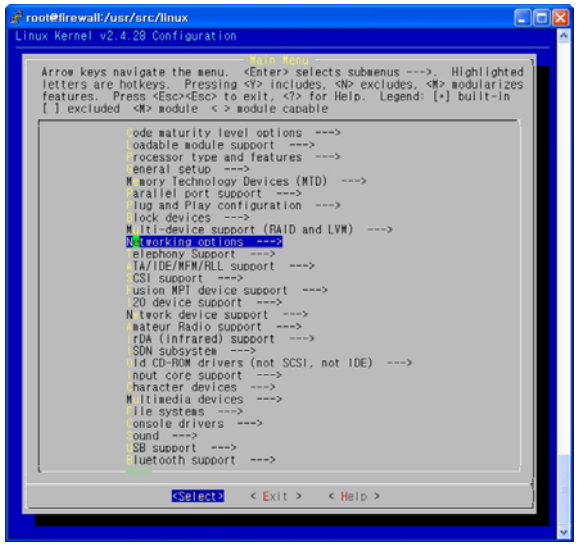
커널소스 디렉토리에서 make config나 make menuconfig를 실행하여 커널 메뉴로 들어가면 된다. 나머지는 각자 환경에 따라 적용하기 바라며 커널 2.4 버전의 경우 아래와 같이 보이게 된다. 이 중에서 방화벽 관련 설정은 “Networking options” 부분에 정의되어 있으므로 이 메뉴를 선택하도록 한다.
매우 많은 선택 메뉴들이 있는데, 가장 기본적인 메뉴만 위와 같이 선택하도록 한다. 만약 모듈(module)로 선택하였을 경우에는 이후에 해당 모듈을 로드(load) 해주어야 한다. 그리고 선택하려는 메뉴가 어떤 기능과 역할을 하는지 모른다면 해당 메뉴에서 우측의 Help를 클릭하면 상세한 안내를 볼 수 있으니 참고하면 된다. 그럼 다음과 같은 순서로 KERNEL을 설정하도록 해보자.

STEP 2. iptables 설치하기

① Kernel 메뉴설정

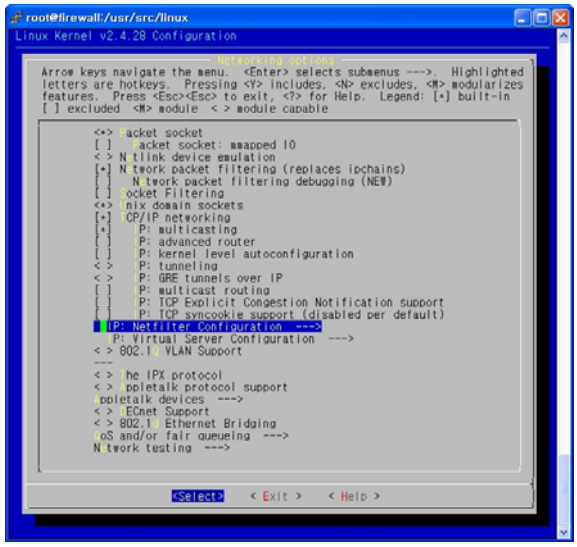
make config (또는 make menuconfig)

입력 → Networking options 선택



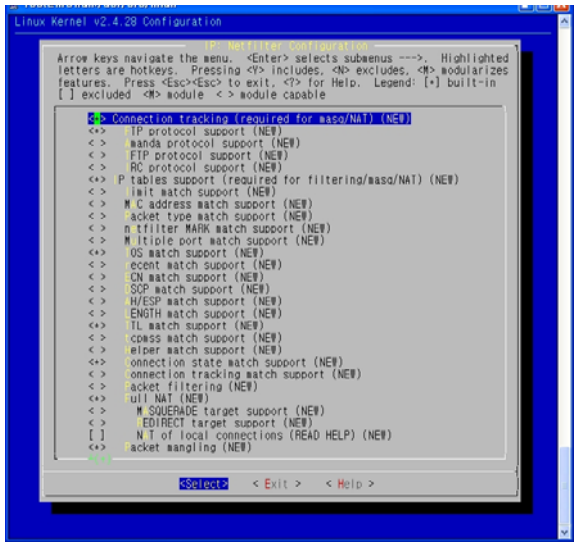
② netfilter 설정확인

IP Netfilter Configuration 선택

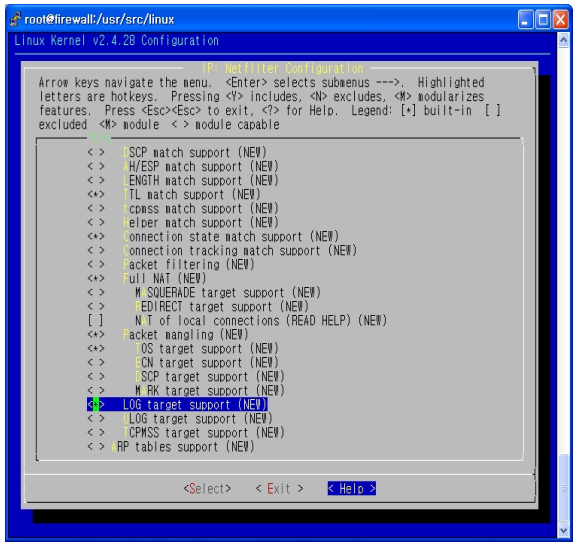


③ netfilter 설정확인

Connection tracking 선택



④ netfilter 설정확인
LOG target support 선택



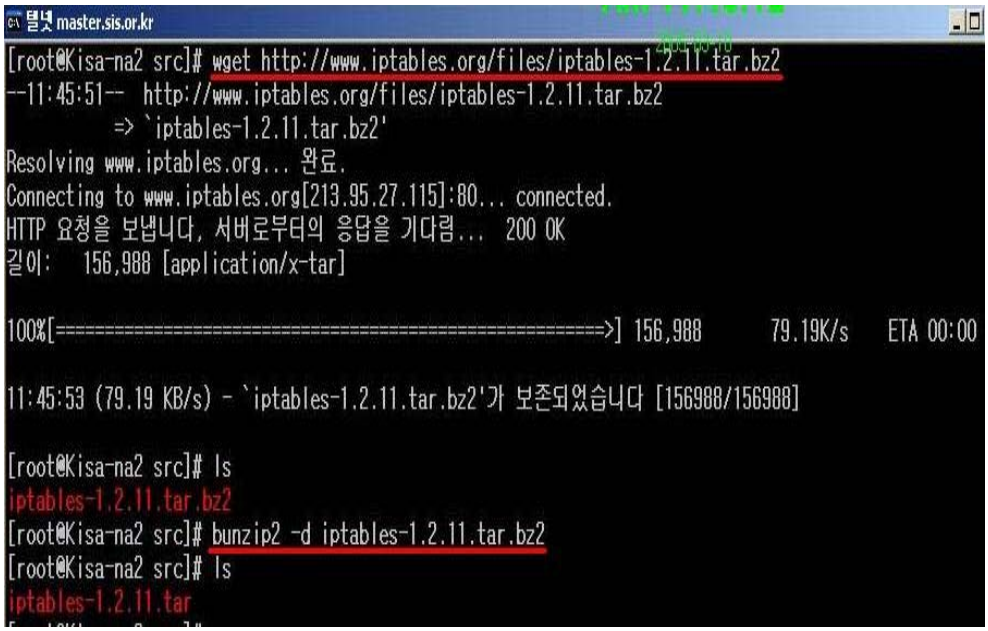
최근의 리눅스 배포판에서는 별도로 커널 컴파일을 하지 않고 CD로 설치된 상태에서도 기본적으로 iptables 방화벽 기능이 모듈로 작동하고 있지만 가급적 커널 컴파일을 통해 불필요한 기능은 끄고, 꼭 필요한 기능만 사용하도록 하는 것을 권장한다. 하지만 커널 컴파일에 익숙하지 않다면 이 과정을 생략하고 바로 "STEP 3. iptables를 활용한 룰 설정"을 익혀도 된다.

iptables 방화벽은 시스템의 응용 프로그램 수준이 아니라 커널 수준에서 작동하기 때문에 먼저 커널에서 방화벽이 작동할 수 있도록 설정하여야 한다. 이를 위해서는 커널소스를 다운받아 커널 컴파일을 하여야 하는데, 소스 파일은 아래 URL에서 다운로드 할 수 있다. 그럼 다음과 같은 순서로 설치하도록 하자.

① iptables rpm 제거

rpm -q iptables : 먼저 RPM 명령어를 이용하여 기존의 iptables 버전을 확인한다.
rpm -e iptables --nodeps : 최신 버전이 아니라면 기존의 iptables 패키지를 삭제하여 새로 설치할 준비를 한다.

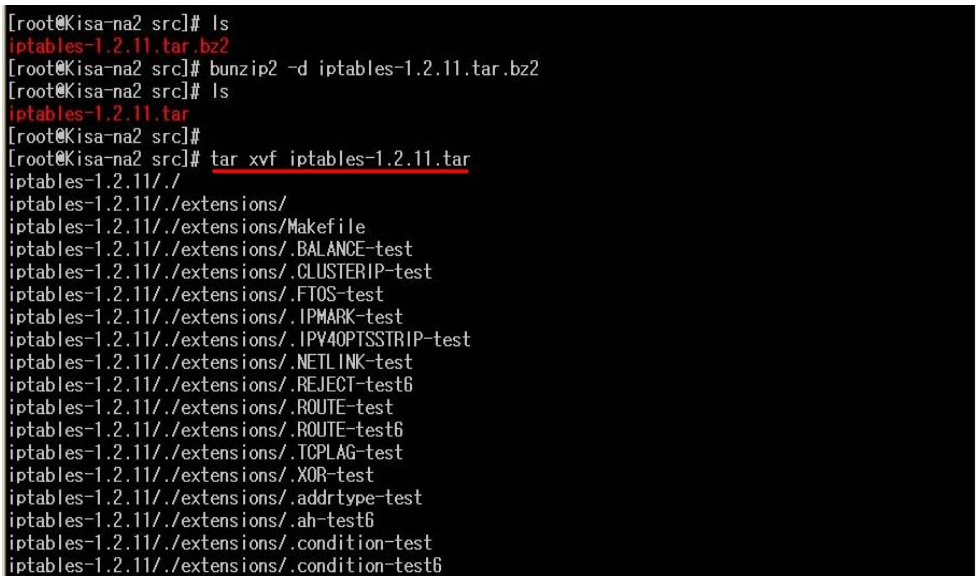
② iptables 소스 다운받기



wget http://www.iptables.org/files/iptables-1.2.11.tar.bz2 : wget이나 lynx를 이용하여 iptables 최신 소스 다운을 받는다.

bunzip2 -d iptables-1.3.1.tar.bz2 : 다운 받은 파일의 압축을 푼다.

③ 소스 압축풀기



rm -f linux : 링크를 확인 후 존재한다면, 삭제한다. linux 디렉토리가 다른 디렉토리로 링크되어 있는지 확인한다.

tar xvf iptables-1.2.11.tar : tar 압축된 파일을 푼다.

④ iptables 디렉토리로 이동

```
[root@Kisa-na2 src]# ls
iptables-1.2.11  iptables-1.2.11.tar
[root@Kisa-na2 src]# cd iptables-1.2.11
[root@Kisa-na2 iptables-1.2.11]# ls
COMMIT_NOTES       Rules.make          ip6tables-save.8       iptables-restore.8     iptables.c
COPYING            TODO                ip6tables-save.c       iptables-restore.c     libipq
CURRENT_ISSUES     extensions          ip6tables-standalone.c iptables-save.8        libiptc
INCOMPATIBILITIES  include             ip6tables.8.in         iptables-save.c
INSTALL            ip6tables-restore.8 ip6tables.c            iptables-standalone.c
Makefile           ip6tables-restore.c ippool                 iptables.8.in
```

cd iptables-1.2.11 : 압축을 푼 디렉토리로 이동한다.

⑤ ipatbles 컴파일

```
텔넷 192.168.0.1
[root@Kisa-na2 iptables-1.2.11]# pwd
/linux/src/iptables-1.2.11
[root@Kisa-na2 iptables-1.2.11]# ls -l /boot
합계 2860
-rw-r--r--  1 root root  237041 10월 12 02:11 System.map-2.6.6-8hl
drwxr-xr-x  3 root root    4096  3월  8 18:11 boot
-rw-r--r--  1 root root   49027 10월 12 02:11 config-2.6.6-8hl
drwxr-xr-x  2 root root    4096  3월 10 15:47 grub
-rw-r--r--  1 root root  196994 12월 29 17:22 initrd-2.6.6-8hl.img
-rw-r--r--  1 root root    6406 10월 11 00:26 message
-rw-r--r--  1 root root 1201765 10월 12 02:11 vmlinuz-2.6.6-8hl
-rw-r--r--  1 root root 1202818  3월 10 15:39 vmlinuz-2.6.6-namool
[root@Kisa-na2 iptables-1.2.11]# make KERNEL_DIR=/boot
Extensions found:
cc -O2 -Wall -Wunused -I/boot/include -Iinclude/ -DIPTABLES_VERSION=₩"1.2.11₩"  -D_UNKNOWN_KERNEL_PO
INTER_SIZE -fPIC -o extensions/libipt_ah_sh.o -c extensions/libipt_ah.c
ld -shared  -o extensions/libipt_ah.so extensions/libipt_ah_sh.o
cc -O2 -Wall -Wunused -I/boot/include -Iinclude/ -DIPTABLES_VERSION=₩"1.2.11₩"  -D_UNKNOWN_KERNEL_PO
INTER_SIZE -fPIC -o extensions/libipt_connlimit_sh.o -c extensions/libipt_connlimit.c
ld -shared  -o extensions/libipt_connlimit.so extensions/libipt_connlimit_sh.o
cc -O2 -Wall -Wunused -I/boot/include -Iinclude/ -DIPTABLES_VERSION=₩"1.2.11₩"  -D_UNKNOWN_KERNEL_PO
INTER_SIZE -fPIC -o extensions/libipt_connmark_sh.o -c extensions/libipt_connmark.c
```

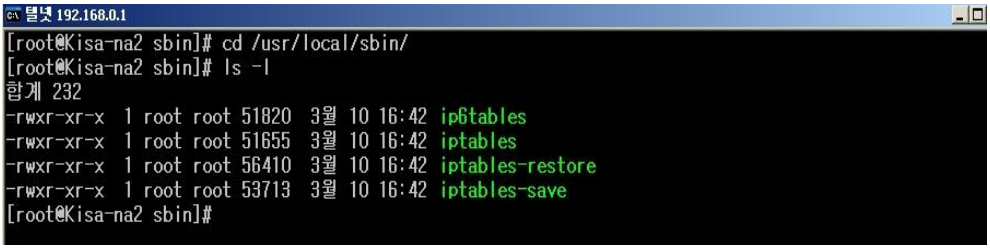
make KERNEL_DIR=/boot : make를 이용하여 컴파일 한다. KERNEL_DIR을 커널디렉토리로 지정해준다.

⑥ iptables install하기

```
sed -e '/@MATCH@/ r extensions/libip6t_matches.man' -e '/@TARGET@/ r extensions/libiptt_targets.man'
 ip6tables.8.in >ip6tables.8
cc -O2 -Wall -Wunused -I/boot/include -Iinclude/ -DIPTABLES_VERSION=₩"1.2.11₩"  -D_UNKNOWN_KERNEL_PO
INTER_SIZE   -c -o libipq/libipq.o libipq/libipq.c
ar rv libipq/libipq.a libipq/libipq.o
ar: creating libipq/libipq.a
a - libipq/libipq.o
rm libiptc/libip4tc.o libipq/libipq.o libiptc/libip6tc.o
[root@Kisa-na2 iptables-1.2.11]# make install KERNEL_DIR=/boot
```

make install : 컴파일 후 인스톨을 수행한다. KERNEL_DIR을 커널디렉토리로 지정해준다.

⑦ 설치 확인하기



cd /usr/local/sbin/ : 설치된 파일을 확인하기 위해 설치 폴더로 이동하여 파일을 확인한다. 상기와 같은 파일이 존재한다면 설치는 이상없이 완료되었음을 확인할 수 있다.

<참고> 선택항목 설명

각각의 환경에 따라 적당히 선택한 후 커널 컴파일한 후 시스템을 재부팅하면 새로운 커널로 적용할 수 있게 된다. 커널 적용이 된 후 방화벽 관리 명령어인 iptables를 설치하여 사용하도록 한다. 만약 iptables rpm 등이 설치되어 있다면 먼저 rpm 버전을 삭제한 후 최신 버전의 소스를 받아 설치하는 것이 좋다.

STEP 3. table. chain, rule 이해하기

방화벽 관련 문서나 서적을 보면 테이블(table)이나 체인(chain), rule(룰) 이라는 용어가 자주 나오는데, 먼저 테이블은 방화벽에서 특정한 기능을 제공하는 것을 기본적으로 filter, nat, mangle 이렇게 3개가 있는데, 각각은 별도의 기능을 가지고 있다. 이를테면 filter 테이블의 경우 이름이 뜻하듯이 패킷을 필터링하는 기능을 가지고 있으며 nat 의 경우 패킷을 필터링하지는 않고 단순히 패킷의 소스나 목적지 주소를 변환시켜 주는 역할을 하며 mangle은 패킷의 특성을 변경하거나 지정하는 기능을 한다. 룰을 사용할 때 특정한 table을 언급하려면 "iptables -t filter"와 같이 -t [table명]의 형식을 사용하면 된다. 만약 -t를 생략하고 별도의 언급이 없을 경우에는 기본적으로 filter 테이블이 사용된다. 그만큼 패킷 필터링이 iptables의 핵심적이고 기본적인 기능이라고 생각하면 된다.

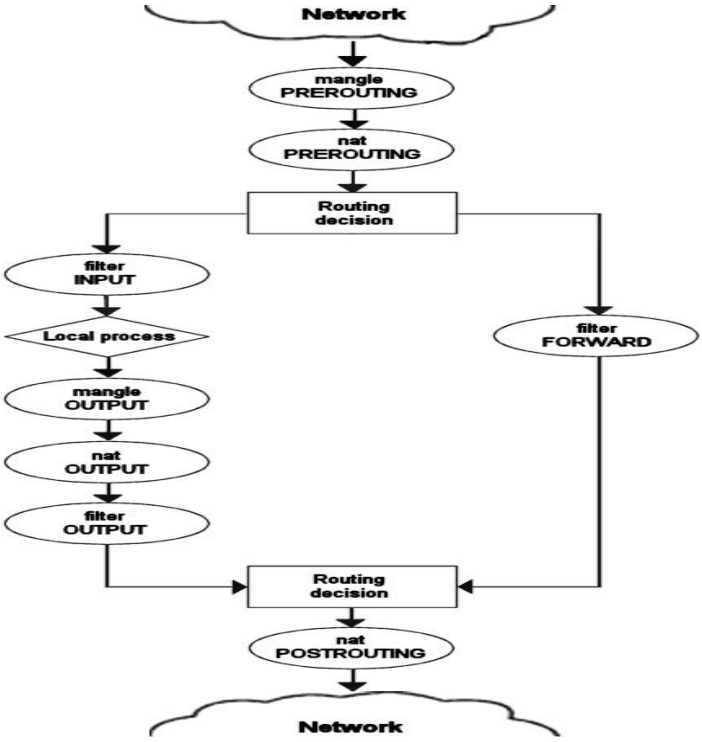
○ filter 테이블
방화벽의 가장 핵심적인 테이블로서 특정 룰에 따라서 패킷을 필터링하거나 허용하는 역할을 한다. filter 테이블에는 기본적으로 3개의 chain이 있는데, INPUT은 외부에서 방화벽 자체로 향하는 패킷에 대한 필터링을 담당하며, FORWARD는 방화벽을 통과하여 방화벽이 보호하는 다른 서버 등으로 향하거나 내부의 다른 서버에서 방화벽을 통해 외부로 나가는 패킷에 대한 필터링, OUTPUT은 방화벽 자체에서 외부로 나가는 패킷에 대한 필터링을 담당한다. 즉, 한 대의 리눅스 서버 자체 내에서 웹 서비스 등 일반 서비스와 함께 패킷 필터링을 제공하고자 할 경우 FORWARD는 사용하지 않고, INPUT과 OUTPUT chain만 사용하면 될 것이다. 그리고 리눅스를 전용 방화벽 장비로 설정하여 보호하고자 하는 서버의 앞단에 설치할 경우에는 FORWARD chain을 사용하여 내부의 서버들에 대한 패킷 필터링 정책을 설정하면 될 것이다.
○ nat 테이블
nat 테이블은 패킷을 필터링하는 기능은 없으며 단지 방화벽으로 향하는 패킷을 방화벽이 보호하는 내부 네트워크의 다른 주소로 포워딩하거나 방화벽 내부 네트워크에서 방화벽을 통해 외부 네트워크로 나갈 때 다른 ip주소로 변환하는 역할을 한다. nat 는 POSTROUTING 과 PREROUTING chain 이 주로 사용되는데, POSTROUTING 은 Source NAT(SNAT) 타겟과 매칭되어 내부 네트워크에서 방화벽을 통해 외부로 나갈 때 사용되며, PREROUTING은 Destination NAT(DNAT)타겟과 매칭되어 주로 외부에서 방화벽 내부 서버로 향하는 패킷을 방화벽이 보호하는 내부서버로 포워딩 할 때 사용된다. 즉, POSTROUTING은 사무실 등에서 사설ip를 사용하면서 하나의 공인 ip로 인터넷을 공유하고자 할 때 즉, 공유기의 용도로 사용할 수 있고 PREROUTING은 사설 ip 로 서버를 운영하면서 외부로 서비스를 하고자 할 때 사용된다.
○ mangle 테이블
이는 그리 자주 사용되지는 않지만 알아둘 필요는 있다. mangle 은 패킷의 TTL이나 TOS 값을 변경하거나 매칭할 때 사용되는데, mangle 테이블은 PREROUTING과 OUTPUT chain 으로 이루어져 있다. PREROUTING에서는 라우팅 경로가 결정되기 전에 방화벽으로 들어오는 패킷에 대해 변경하거나 매칭하고, OUTPUT chain에서는 내부에서 생성된 패킷이 방화벽을 통해 나갈 때 변경하거나 매칭한다.

다음으로 체인(chain)은 패킷이 이동하는 경로를 나타내는데, 각각의 table은 각각의 다른 chain을 가지고 있다. 예를 들면 filter 테이블의 경우 INPUT, FORWARD, OUTPUT chain 등이 있다. rule은 각각의 chain 에 설정하는 일련의 방화벽 정책을 뜻한다.

chain과 관련하여 그렇다면 패킷은 어떠한 경로, 즉 어떤 chain을 통해 이동하는지에 대해 알아보도록 하자. 다음 그림에서와 같이 방화벽으로 향하는 패킷은 라우팅 경로에 따라 크게 두 갈래로 나누어진다. 즉, 방화벽 자체로 향하는 패킷과 방화벽을 통과하여 방화벽 내부에 위치한 다른 서버를 향하는 패킷에 따라 패킷 경로가 아래 그림에서 좌측 또는 우측으로 갈라지게 되는 것이다.

따라서 첫 번째 경우인 서버 자체 방화벽을 운영할 경우 즉, 리눅스 서버 자체에 서비스 하면서 시스템내 커널에 방화벽 기능을 활성화하여 방화벽을 가동할 경우에

는 패킷이 좌측의 경로를 따르게 되고, NAT나 브리지 방화벽을 사용하여 패킷을 방화벽 내부의 서버로 포워딩이나 스위칭 할 경우에는 우측의 경로를 따르게 된다. 그러나 이와 관계없이 mangle table의 PREROUTING chain, nat table의 PREROUTING chain, POSTROUTING chain은 공통적으로 거쳐 가는 것을 알 수 있다.



[그림 4] 방화벽에서의 패킷 이동 경로

STEP 4. iptables 명령어 사용 익히기

iptables를 사용할 준비가 되었으므로 이제 iptables의 명령어 사용방법, 즉 rule 설정 방법에 중요한 몇 가지만 알아본다. iptables 의 기본적인 명령어 용법은 다음과 같다.

```
iptables [-t table명] 명령어 [매칭 옵션] [타겟]
```

제일 먼저 지정할 것은 사용할 table명을 설정하는 것인데, 앞에서 언급한 바와 같이 별도로 table명을 지정하지 않을 경우에는 **filter 테이블**이 사용된다. 즉, 패킷을 필터링하기 위해서는 filter 테이블을 사용하게 되는데, 이때 "-t filter" 와 같이 언급해도 되고, filter 는 기본값이므로 별도로 지정하지 않아도 무방하다. 그러나 nat 나

mangle 기능을 이용하려면 반드시 -t nat, -t mangle 과 같이 해당 테이블 명을 명시하여야 하며 그렇지 않으면 filter 테이블로 인식하게 되므로 주의하여야 한다.
그리고 테이블 명 다음에는 매칭 옵션(이를테면 tcp 인지, 목적지 ip 가 무엇인지, 포트번호가 무엇인지 등)을 현재 테이블에 추가할 것인지(append 또는 insert) 삭제할 것인지(delete)등의 명령어를 지정하게 된다.
다음으로는 어떠한 패킷을 필터링하거나 nat 또는 mangle할 것인지 매칭 옵션을 지정할 수 있는데, 이를테면 어떠한 ip 주소를 소스로 한 패킷인지, 어떠한 인터페이스를 통과하는지 또는 정책을 설정할 패킷이 tcp인지, udp인지 등이 그것이다.
마지막으로 이 룰에 매칭되는 패킷이 있을 경우 이 패킷을 통과시킬 것인지 아니면 거부할 것인지 등을 타겟(target)에서 지정할 수 있다.

1) 기본명령어

iptables에서 제공하는 옵션과 명령어는 매우 많아서 어디에서부터 보아야 할지 막막한 것이 사실이다. 여기에서 소개하는 명령어는 기본적으로 알고 있어야 할 부분이므로 천천히 읽어 보면 된다.

-A, --append
예: iptables -A INPUT
-A 를 이용하여 룰을 입력하면 입력한 룰은 지정한 chain의 제일 마지막에 추가된다. 즉 위 예와 같이 실행하였을 경우, INPUT chain 의 제일 마지막에 추가되는 것이다. 방화벽에서는 어떤 룰이 먼저 설정되었는가에 따라 패킷 허용 및 차단의 결과가 크게 달라질 수 있으므로 룰의 순서를 주의하여야 한다. 여기에서 -A 는 실행한 순서대로 제일 마지막에 추가되는데 반해 insert의 의미인 -I 를 입력하였을 경우에는 룰의 제일 앞에 위치하게 된다. 또 한 가지 주의할 것은 대소문자의 구별인데, 경우에 따라 대문자와 소문자의 의미가 달라질 수 있으므로 주의하여야 한다.
-D, --delete
예) iptables -D INPUT -p tcp --dport 80 -j DROP iptables -D INPUT 1
-D 또는 --delete 는 룰을 잘못 입력했거나 필터링 정책이 변경되었을 경우 이미 입력한 룰을 테이블에서 삭제하는 명령어이다. -D를 이용하여 룰을 삭제하는 방법은 두 가지로 나눌 수 있는데, 첫 번째 예제처럼 룰을 추가할 때 사용했던 완전한 룰을 지정하되 -A 나 -I 대신 -D를 사용하는 방법도 있고 두 번째 예제처럼 입력한 룰의 순서대로 지정된 번호를 이용하는 방법도 있는데, 일반적으로 첫 번째 방법을 더 많이 이용한다.
-I, --insert
예)iptables -I INPUT -p tcp --dport 80 -j ACCEPT
앞에서도 잠깐 언급했듯이 -I 또는 --insert는 -A 와 같이 해당 chain 에 룰을 추가하는 것이다. 그러나, -A 는 chain 의 제일 마지막에 추가하는 것과는 달리 -I 는 chain의 제일 처음에 추가된다는 차이가 있다. -I 는 일반적으로 먼저 룰이 적용된 상태에서 새로운 룰을 테스트하고자 할 때 또는 임시로 특정한 룰을 적용하고자 할 때 사용된다.
-L, --list
예)iptables -L INPUT

-L 은 list의 의미로 커널 메모리에 로드되어 적용되어 있는 chain 의 룰 목록을 보여준다. 위 예제의 경우 INPUT을 지정하였으므로 현재 INPUT chain 에 설정되어 있는 룰을 보여준다. 이때 -n 을 함께 사용하면 ip 주소 및 서비스명에 대해 역 질의를 하지 않는다.
그리고 -v와 함께 사용하면 각각의 룰에 매칭된 패킷의 개수(pkts) 와 바이트(bytes)정보도 보여준다.

-F, --flush

예) iptables -F INPUT

-F 는 모든 룰을 플러싱(flushing) 또는 삭제하는 명령어이다. -D 가 룰을 한 개씩 삭제하는 것과 달리 -F 는 모든 룰을 한꺼번에 삭제하는 명령어라고 할 수 있다. 위 예와 같이 -F 다음에 INPUT 을 지정할 경우에는 INPUT table 에 있는 모든 룰을 삭제하게 되고, 아무것도 입력하지 않을 경우에는 filter 테이블내 모든 룰을 삭제하게 된다. 따라서 nat나 mangle 테이블에 있는 룰을 초기화하려면 "iptables -t nat -F" 나 "iptables -t mangle -F" 와 같이 테이블명을 지정해 주어야 한다.
-F 는 주로 새롭게 룰을 생성하고자 할 때 혹시 남아있을지 모르는 룰을 깨끗하게 삭제하기 위해 사용되는데, 주의할 점은 -P(뒤에서 살펴본다.)로 지정한 기본 정책은 변경되지 않는다는 것이다. 즉, INPUT의 기본 정책을 DROP 으로 설정한 후 특정 ip 에서의 접속을 허용하는 룰을 생성한 후 -F 로 룰을 초기화하게 되면 허용하는 룰은 삭제되고 기본정책인 DROP 은 그대로 남게 되므로 일체의 모든 접속이 거부되므로 주의하여야 한다.

-N, --new-chain

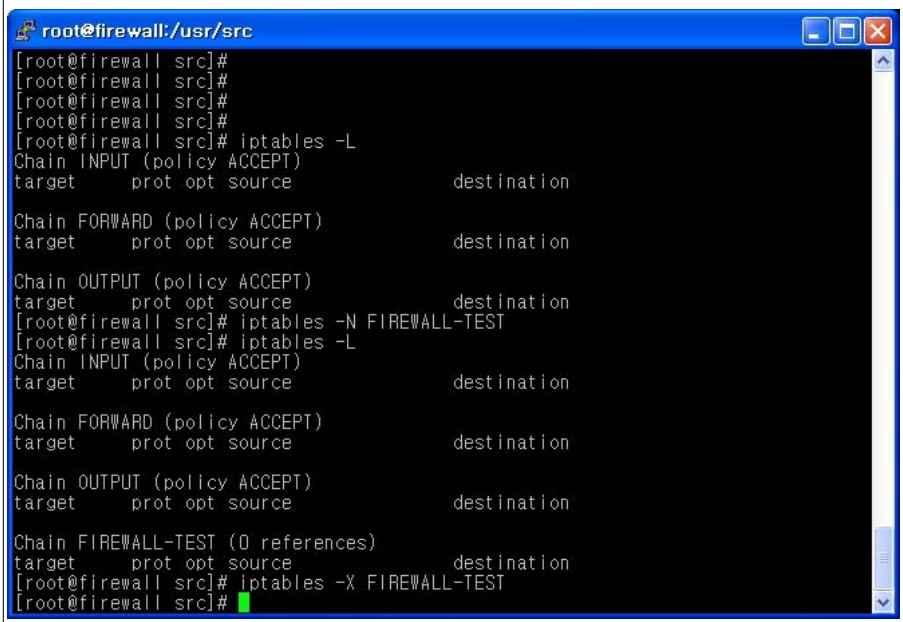
예) iptables -N TEST

-N 은 새로운 chain 을 생성하는 명령어이다. 앞에서 언급한 것처럼 새로운 chain 을 생성할 수는 있지만 새로운 table은 생성할 수 없다. 일반적으로 룰이 단순하다면 굳이 chain을 생성할 필요는 없지만 반복적이고 복잡하다면 생성해서 사용하는 것도 효율적이다.

-X, --delete-chain

예)iptables -X TEST

-X 는 지정한 chain 을 삭제하는 명령어이다.



그러나 아래와 같이 기본적으로 내장되어 있는 OUTPUT chain을 삭제하려고 하자, built-in 이라 삭제되지 않는 것을 알 수 있다. 왜냐하면 INPUT 이나 FORWARD, OUTPUT chain 은

filter 테이블에서 반드시 있어야 할 기본 chain 이기 때문이다. 
-P,--policy
예) iptables -P INPUT DROP
-P 는 앞에서 잠깐 언급했듯이 지정한 chain 에 대한 기본정책(default policy)을 의미한다. 이를테면 방화벽으로 들어오는 패킷은 INPUT 테이블에 지정된 룰 순서대로 매칭이 되는지 점검하게 되는데, 만약 INPUT 테이블에 있는 모든 룰에 매칭되지 않을 경우에는 기본정책인 -P 에서 지정된 정책이 적용되는 것이다. 일반적으로 기본 정책에서는 DROP을 설정하고 (REJECT 는 지원하지 않는다.) 세부 룰에서 특정한 패킷을 허용(ACCEPT)하는 정책을 설정한다. 물론 정책에 따라 그 반대도 있을 수 있을 것이다. 참고로 기본정책을 설정할 때는 -j 를 사용하지 않는다.

2) 매칭옵션

일반적으로 한 룰에서 매칭 옵션을 많이 지정하면 할수록 좀 더 세부적인 패킷을 지정하게 된다.

-p, --protocol
예) iptables -A INPUT -p tcp
-p 옵션은 tcp 나 udp, icmp 처럼 매칭되는 프로토콜(protocol)을 지정하는데, not의 의미인 느낌표(!)를 함께 사용할 수도 있다. 즉, -p ! tcp 는 tcp가 아닌 즉 udp와 icmp를 의미하는 것이다. 소문자인 -p 가 protocol 을 의미하는 대신, 앞에서 살펴보았던 대문자인 -P는 기본정책을 의미한다는 것을 주의하기 바란다.
-s, --source
예) iptables -A INPUT -s 192.168.1.0/24
-s 옵션은 패킷의 소스(source) ip 주소를 뜻하는데, 192.168.1.1 과 같이 단일한 하나의 ip주소를 지정할 수도 있고, 192.168.1.0/255.255.255.0 이나 192.168.1.0/24 와 같이 넷마스크나 CIDR 를 사용하여 ip 대역을 지정할 수도 있다. 물론 단일한 ip를 지정할 경우 netmask를 255.255.255.255로 사용하거나 CIDR 는 /32를 지정해도 된다. 만약 위의 예와 같이 사용할 경우에는 192.168.1.x 대역의 모든 ip 주소 대역에서 들어오는 패킷을 뜻하게 되며 -s를 지정하지 않을 경우에는 ip 와 관계없이 모든 곳으로부터 오는 패킷을 뜻하게 된다.
-d, --destination
예) iptables -A INPUT -d 192.168.1.2
-s 옵션이 패킷의 소스 ip 주소를 의미하는데 반해 -d 옵션은 목적지(destination) 주소를 뜻한다. 문법은 -s 와 동일하며 -d 를 지정하지 않으면 모든 목적지에 대한 매칭이 된다. 만약 위와 같이 -s가 없이 -d만 지정되어 있으면 목적지 ip 인 192.168.1.2 로 향하는 모든 패킷을 뜻하지만 “-s 192.168.1.1 -d 192.168.1.2” 와 같은 경우 소스 ip 가 192.168.1.1 이면서 목적지ip 가 192.168.1.2 인 패킷을 뜻한다. 여기에서 “이면서” 는 그리고(and)의 개념이지 또는(or)의 개념이 아니라는 점을 주의하기 바란다.

-i, -o
예) iptables -A INPUT -i eth0
-i 또는 -o 는 eth0 이나 eth1, lo 등과 같이 어떤 인터페이스에서 들어오거나 나가는 패킷인지 지정하는 옵션이다. -i 는 in 으로서 들어오는 인터페이스를, -o 는 out 으로서 나가는 인터페이스를 의미한다. 인터페이스가 여러 개 연결되어 있는 복잡한 네트워크에서 유용하게 사용할 수 있는데, 만약 eth0 과 같이 1개의 인터페이스만 설정되어 있다면 어차피 인터페이스는 하나이므로 굳이 인터페이스를 지정할 필요가 없다.
--sport , --dport
예) iptables -A INPUT -i eth0 -p TCP -s 192.168.5.3 --sport 1024:65535 --dport 161
--sport 은 패킷의 소스포트를, --dport 는 목적지 포트를 의미하는데, 이 옵션을 사용할 때에는 “-p tcp” 나 “-p udp” 와 같이 tcp 인지 udp인지 반드시 명시하여야 한다. 아울러 앞에서 살펴본 ip 주소의 경우에는 -s, -d 와 같이 -이 1개이지만 포트 번호의 경우에는 --sport, --dport 와 같이 - 이 2개라는 것을 주의하기 바란다. --sport 와 --dport 는 위의 예와 같이 함께 사용할 수도 있고, --sport 또는 --dport 하나만 사용할 수도 있다. 물론 지정하지 않을 경우에는 모든 포트를 뜻하게 되므로 "-s 192.168.5.3 --sport 1024:65535" 와 같은 경우에는 목적지 포트는 관계없이 소스 ip 는 192.168.5.3 이면서 소스 포트는 1024부터 65535 까지인 패킷을 뜻하게 된다. 또한 사용할 때에는 --dport 161과 같이 특정 포트를 지정할 수도 있고 --sport 1024:65535 와 같이 특정 포트의 대역을 지정할 수도 있다. 만약 1024: 만 지정하였을 경우에는 1024부터 65535까지를 의미하며 :1024 만 입력하였을 경우에는 1024 이하 즉 0부터 1024 까지를 의미한다. 그리고 not 의 의미인 !을 함께 이용할 수도 있는데, 1024:65535는 1024부터 65535 까지가 아니므로 0:1023의 의미와 동일하다.
--tcp-flags
예)iptables -A INPUT -p TCP --tcp-flags ACK,FIN FIN
tcp 는 udp나 icmp 와 달리 패킷의 특성상 SYN,ACK,FIN,PSH,URG,RST 등 6개의 tcp-flags 라는 것을 가지고 있다. 이를테면 접속을 요청하는 tcp 패킷의 경우 패킷에 SYN 비트가 설정되고, 요청에 대해 응답을 할 경우에는 ACK, 접속을 종료할 경우에는 FIN 비트가 설정된다. iptables 에서는 이러한 tcp-flag 을 지정하여 관리자가 의도한 특정한 패킷에 대한 매칭을 할 수 있는데, 이는 tcp-flags 사이의 blank 를 기준으로 좌측은 체크할 tcp-flags를, 우측에는 체크한 항목 중 매칭되는 tcp-flags을 의미한다. 즉, 위의 예에 있는 룰을 예로 들면, INPUT chain 을 통하여 들어오는 tcp 패킷에서 6개의 tcp-flags중 모두 살펴 볼 필요 없이 ACK와 FIN 부분만 살펴보아서 ACK는 없이 FIN 만 설정되어 있는 패킷을 뜻한다. 즉, ACK 비트가 없는 FIN 패킷을 의미하는 것이다. 그리고 여기에서는 ACK와 FIN만 체크하였으므로 SYN 비트나 PSH등 다른 tcp-flags가 있거나 또는 없어도 관계없다. 이외 ALL 은 모든 tcp-flags 을 의미하며, NONE 은 아무런 tcp-flags 가 없는 것을 뜻한다.
--syn
예) iptables -A INPUT -p tcp --syn -j DROP
tcp에서 syn 비트가 설정된 패킷의 의미는 남다르다. 왜냐하면 syn 비트가 설정된 패킷은 tcp 연결을 맺기 위해 제일 먼저 보내어지는 이른바 접속 요청 패킷이기 때문이다. 만약 위의 예와 같이 들어오는 tcp 패킷 중 syn 비트가 설정된 패킷을 거부하였다면 모든 tcp 접속 연결 요청은 거부될 것이다. 그러나 방화벽에서 외부로 접속 요청하였을 경우에는 OUTPUT 에서는 syn 이지만 syn 에 대한 응답인 INPUT 에서는 syn 이 아니라 syn 과 ack 비트가 함께 설정된 패킷이 들어오게 되므로 내부에서는 외부로 tcp 접속을 할 수 있게

된다. --syn 은 ! 과 함께 사용하여 '! --syn' 과 같이 사용될 수 있는데, 이때에는 이미 접속된 tcp 연결을 의미하게 된다.
--icmp-type
예) iptables -A INPUT -s 192.168.1.1 -p ICMP --icmp-type echo-request
tcp 나 udp 는 포트 번호로 상호 어떤 서비스인지 구별한다. 즉, 같은 tcp 라도 23번은 telnet, 80번은 http, 110번은 pop3인 것이 그것이다. 그러나, icmp 는 tcp 나 udp 와 달리 포트의 개념이 없으며 대신 icmp-type 과 code 라는 것을 사용하여 구별한다. tcp 나 udp 의 경우 포트 번호 대신 포트 이름(예: http)을 써도 되는 것과 같이 icmp-type 을 지정할 때에는 정확한 type 이름을 써도 되고 해당이름에 대한 번호를 지정해도 된다. 그러나 가급적 tcp 나 udp 는 포트번호로, icmp 는 type 이름을 설정하는 것이 더욱 명료하고 권장된다. icmp-type과 code 에 대해서는 http://www.iana.org/assignments/icmp-parameters 를 참고하기 바란다. 위 예의 경우 INPUT chain 을 통해 들어오는 패킷 중 소스 주소가 192.168.1.1 이면서 icmp-type이 echo-request 인 icmp 패킷을 의미하는 것이다. 여기에서 프로토콜을 뜻하는 -p 다음의 icmp 는 위와 같이 대문자로 써도 되고 소문자로 써도 된다.
-m state
예) iptables -A INPUT -p TCP ! --syn -m state --state NEW
-m state 는 상태 추적(Connection State)에서 사용되는 것으로 상태추적에서는 NEW, ESTABLISHED, RELATED, INVALID 이렇게 4가지의 상태를 제공한다. 여기에서 NEW는 새롭게 연결을 시작하려 하거나 이전의 연결추적 테이블에 보이지 않는 패킷을 의미한다. tcp 의 경우 연결을 맺기 위한 정상적인 syn 패킷이 여기에 해당할 것이다. ESTABLISHED 는 클라이언트의 연결 시도 후 서버에서 응답하여 이미 연결되어 있는 상태를 의미하며 연결이 맺어진 이후의 tcp ack 메시지나 이미 데이터가 오간 udp 또는 icmp echo-request에 대한 icmp echo reply 메시지 등이 이에 해당한다. RELATED 는 새롭게 연결을 시작하려고 하나 이미 연결추적 테이블에 접속과 관련 있는 ESTABLISHED 가 있는 경우를 뜻한다. 주로 icmp 에러 메시지가 여기에 속하며 ftp 접속시 사용하는 두 번째 포트인 ftp-data(tcp 20번등)가 대표적인 경우이다. 마지막으로 INVALID는 연결 상태를 알 수 없거나 잘못된 헤더를 가지고 있는 경우를 뜻한다. 예는 들어오는(INPUT) tcp 패킷(-p tcp) 중 syn flag가 설정된 패킷이 아니면서 (! --syn) 연결 추적 테이블에는 처음 보이는 패킷(-m state --state NEW)을 의미한다. 연결 추적 테이블에 처음 보이는 tcp 패킷이라면 정상적이라면 syn 패킷 이외에는 없을 것이므로 따라서 위의 룰에 매칭되는 패킷은 위조된 패킷이나 비정상적인 패킷일 것이다.

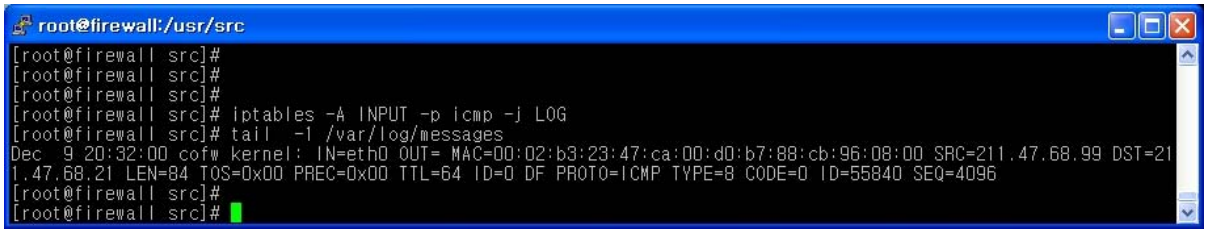
3) 타겟 (target)

iptables를 이용한 룰의 제일 마지막 부분인 타겟에 대한 설정 부분이다. 지금까지는 임의의 조건에 대한 패킷을 지정하는 부분이었고, 이러한 룰에 매칭되는 패킷이 있을 경우 어떻게 처리할 것인지를 지정하는 부분이 바로 타겟이 되고, '-j 타겟' 형식으로 지정하면 된다. -j 뒤의 타겟에는 ACCEPT, DROP, LOG, SNAT, DNAT 등이 올 수 있다. ACCEPT는 말 뜻 그대로 룰에 매칭되는 패킷을 허용한다는 의미이

며, DROP은 패킷을 필터링(거부)하는 것을 뜻한다. LOG는 단지 해당하는 룰에 매칭되는 패킷 정보를 로그에 남긴다는 의미이며 그 자체로 로그만 남길 뿐 허용하거나 거부하는 등의 다른 액션을 취하지는 않는다. 그리고, SNAT와 DNAT는 NAT 즉, 주소 변환을 한다는 것을 의미한다. 몇 가지 타겟에 대해 좀 더 알아보자.

- log target

룰에 매칭되는 패킷을 LOG 타겟으로 보내면 패킷의 ip 주소등의 정보를 로그에 남기도록 하며, 이 정보는 dmesg 등의 명령어나 /var/log/messages 파일 또는 임의로 지정한 다른 파일에서 확인할 수 있는데, 주로 룰이 정상적으로 작동하는지 디버그 할 때나 방화벽에서 필터링 된 패킷의 로그를 보고자 할 때 사용될 수 있다. LOG에서 주로 사용하는 옵션은 --log-level 과 --log-prefix 가 있는데, --log-level 에서는 syslogd 에서 제공하는 것처럼 어떠한 로깅 레벨을 사용할 것인지 정할 수 있다. 그리고 --log-prefix를 이용하면 매칭되는 패킷에 대해 로그정보를 남길 때 로그 앞에 어떤 메시지(일종의 말머리)를 남길 것인지 지정할 수 있다. 이를테면 iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT tcp packets" 와 같이 지정할 경우 이 룰에 매칭되어 저장되는 모든 로깅 정보 앞에는 “INPUT tcp packets” 라는 문자열이 붙게 되는 것이다.



[그림 5] -j LOG로 생성된 로그

위 그림의 경우 들어오는 패킷중 icmp에 대해서 로그를 남기도록 설정한 예인데, 설정 후 /var/log/messages 파일을 보면 위와 같이 로그가 생성된 것을 알 수 있다.

- REJECT 타겟

REJECT는 DROP 과 같이 룰에 매칭 되는 패킷을 거부하지만 DROP이 패킷을 거부한 후 어떠한 추가적인 동작이 없는 반면 REJECT는 패킷 거부와 함께 패킷의 소스 주소로 패킷이 거부되었다는 에러 메시지를 발송한다는 차이가 있다. 패킷을 REJECT할 때는 프로토콜에 따라 다른데, 만약 TCP인 경우에는 reset 비트가 설정된 TCP 패킷으로 또는 ICMP 패킷으로 응답하고, 이외 UDP 등의 패킷일 경우에는 ICMP 패킷으로 응답한다.

- NAT 에서의 타겟

iptables에서 제공하는 NAT를 이용하면 크게 두 가지 용도로 활용할 수 있다. 첫 번째는 일종의 인터넷 공유기로서 내부 네트워크에서는 사설 ip를 사용하면서 NAT 방화벽을 통과하여 외부 네트워크로 나갈 때는 공인 ip를 소스로 한 주소로 변환되어 사용하는 경우이다. 두 번째는 반대로 외부에서 공인 ip로 들어오는 접속 요청이 NAT 방화벽을 거치면서 내부의 사설ip 로 변경되는 경우로서 웹 서버 등을 구축하여 서비스를 제공할 때 실제 서버들은 사설 ip를 운영하면서 서버 앞단에 설치되어 있는 방화벽에서 공인 ip 로 들어온 요청을 내부 사설 ip의 특정 포트로 포워딩을 하는 경우이다. NAT를 사용할 때는 리눅스 서버를 방화벽 전용으로 사용하여야 하며 내부 네트워크와 외부 네트워크를 연결해 주므로 최소한 두 개의 인터페이스가 연결되어 있어야한다.
NAT에서는 Source NAT(SNAT), Destination NAT(DNAT), 마스커레이드(MASQUERADE), REDIRECT 이렇게 네 가지 종류의 타겟을 지원하며 이 타겟을 사용하려면 "-t nat"을 함께 사용하여야 한다. 아울러 또 하나 중요한 것은 NAT 방화벽을 통과하는 패킷을 공인에서 사설 또는 사설에서 공인 등 다른 서버나 장비로 포워딩 해주어야 하므로 방화벽이 설치된 시스템에서는 커널 파라미터인 net.ipv4.ip_forward 가 반드시 **on** 의 의미인 **1**로 설정되어 있어야 한다. 이 값을 1로 설정하려면 "echo 1 > /proc/sys/net/ipv4/ip_forward" 또는 "sysctl -w net.ipv4.ip_forward=1" 을 실행하면 된다.

STEP 5. iptables를 활용한 보안 룰 설정하기

이제 지금까지 살펴본 iptables의 정보를 활용하여 구체적인 룰을 설정해 보고 실제 서비스중인 서버에 직접 방화벽을 탑재하여 운영하는 서버자체 형태의 방화벽을 구현해 보도록 하자. 리눅스 시스템 자체를 방화역의 용도로만 사용하는 NAT 나 브리지 방식은 다음에 알아보도록 한다.
그럼, 이제부터 iptables를 이용하여 각 서비스에 대한 보안 정책을 다음과 같이 설정하여 사용해보도록 하자.

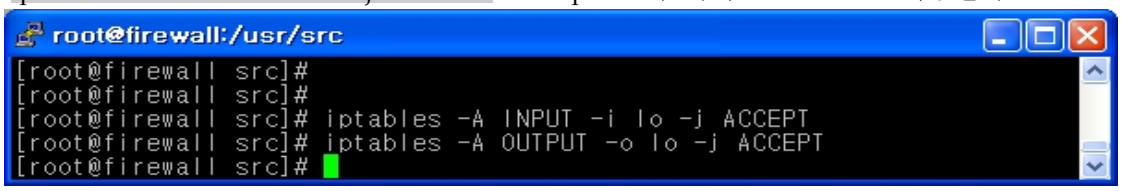
① 룰 초기화 하기
iptables -L : 현재 설정된 룰을 확인하다.
iptables -t filter -F : filter 룰을 삭제한다.
iptables -t nat -F : nat 룰을 삭제한다.
iptables -t mangle -F : mangle 룰을 삭제한다.

iptables를 이용하여 방화벽 룰을 설정할 때 제일 먼저 하여야 할일 혹 이미 존재할지 모르는 기존의 룰을 모두 삭제하는 것이다. iptables -L을 실행하여 현재 설정된 룰을 확인하여 하나하나 삭제해도 되지만 기존에 존재하는 모든 룰을 한꺼번에 삭제할 수 있는데, 이러한 것을 **플러싱(flushing)**한다고 한다. 다음과 같이 특정한 chain을 지정하지 않을 경우에는 INPUT이나 OUTPUT등 모든 테이블의 룰을 동시에 초기화한다.

② Loopback 트래픽 허용하기

iptables -A INPUT -i lo -j ACCEPT : Loopback에 대해 INPUT을 허용한다.

iptables -A OUTPUT -o lo -j ACCEPT : Loopback에 대해 OUTPUT을 허용한다



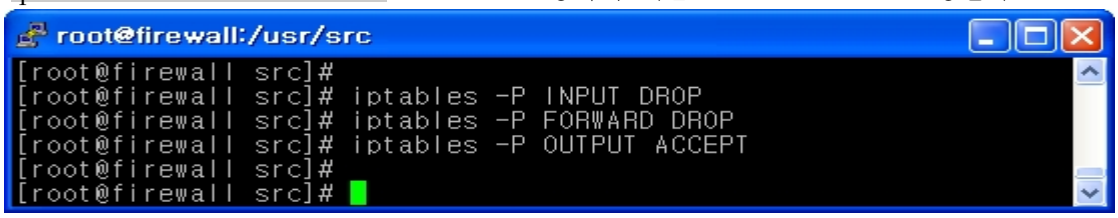
루프백 인터페이스는 시스템 내부의 가상 인터페이스이므로 루프백과 관련된 모든 트래픽은 허용하는 것이 좋다. 현재 자신의 시스템에서 루프백 트래픽이 사용되는지 알아보려면 간단히 tcpdump -i lo 로 확인해 보기 바란다. 여기에서 -i 는 interface 의 의미이다. 위 그림은 lo 인터페이스를 통해 들어오는 패킷과 lo 인터페이스를 통해 나가는 패킷을 허용 설정한 예이다. 이 룰에서의 -i 은 in 의 의미이고, -o 은 out 의 의미이다.

③ 기본정책(default policy) 설정

iptables -P INPUT DROP : INPUT 정책의 기본은 DROP으로 설정한다.

iptables -P FORWARD DROP : FORWARD 정책의 기본은 DROP으로 설정한다.

iptables -P OUTPUT ACCEPT : OUTPUT 정책의 기본은 ACCEPT으로 설정한다.



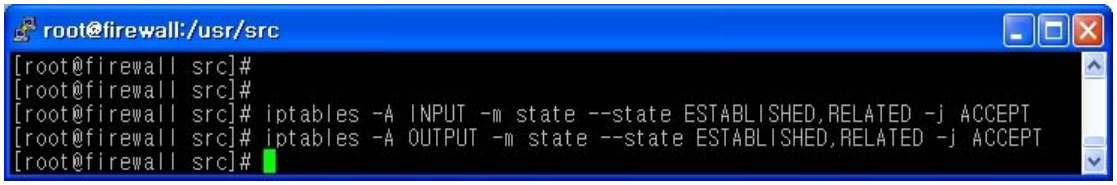
기본 정책은 앞에서 설명한 바와 같이 지정한 모든 룰에 매칭되지 않을 때 최종적으로 매칭되는 것으로 대문자인 -P 로 표현하며 **ACCEPT**, **DROP** 둘 중에 하나가 사용된다. 기본 정책으로 REJECT는 사용하지 않으며 -j DROP 과 같이 -j 를 사용하지 않는다는 점을 주의하기 바란다. 대부분 기본 정책으로 DROP을 설정하는데, 특히 원격에서 설정시에는 사전에 허용 정책이 없으면 접속 자체가 끊겨 버리므로 주의하여야 한다. 상기의 내용은 **INPUT, FORWARD, OUTPUT** 에 대해 기본 정책을 설정한 예인데, 자체 방화벽에서 FORWARD 는 사용하지 않기 때문에 DROP을 설정하였고, 들어오는 패킷만 제대로 처리하면 되므로 OUTPUT은 ACCEPT를 설정하고 INPUT은 DROP을 하였다. 여기에서

OUTPUT을 ACCEPT 로 설정한 이유는 어차피 INPUT 과 OUTPUT이 동시에 ACCEPT 되어야 통신이 될 수 있는데, OUTPUT 만 ACCEPT 해 두면 외부에서 내부로 들어오는 패킷에 대해서는 엄격하게 통제하되 내부에서 외부로 나가는 트래픽에 대해서는 허용하는 것이 좋기 때문이다. 이는 뒤에서 살펴볼 상태추적을 통해 이미 연결을 맺어 연결이 성립된 ESTABLISHED 와 RELATED 상태는 모두 허용하도록 하였기 때문에 가능하다.

④ 상태추적 설정

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT : INPUT 룰에 의해 허용된 트래픽을 지속적으로 허용한다.

iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT : OUTPUT 룰에 의해 허용된 트래픽을 지속적으로 허용한다.



상태추적이 제공되지 않는 stateless 구형 방화벽의 경우 매번 들어오고 나가는 패킷마다 패킷을 허용할 것인지 혹은 차단할 것인지 여부를 체크하여야 했지만 상태 추적을 이용하면 이미 룰에서 허용이 된 트래픽의 경우 뒤이어 전송되는 모든 패킷을 다시 첫 번째 룰부터 검사할 필요 없이 바로 허용할 수 있다. 이것이 바로 상태 추적의 가장 큰 장점 중 하나이다. 따라서 아래 두 줄을 방화벽 룰 설정시 가능한 먼저 설정해 주면 룰이 단순해지고 더욱 효율적이라 할 수 있다. 아래에서 ESTABLISHED 와 RELATED 는 이미 NEW 를 통해 트래픽이 허용된 후 이와 관련되어 통신하는 패킷에 대한 허용 설정이므로 NEW 에서만 제대로 설정해 주면된다.

⑤ 사설 IP 주소로 필터링

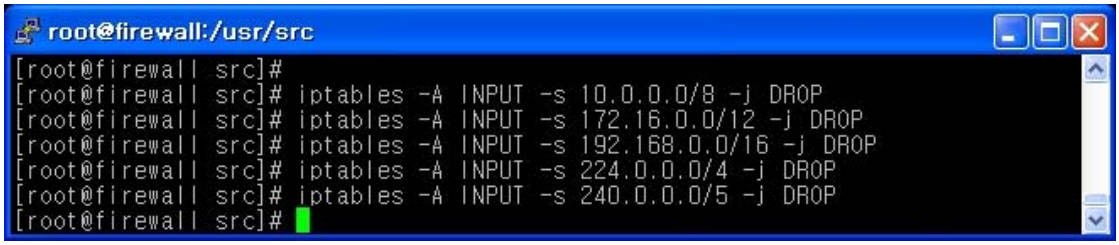
iptables -A INPUT -s 10.0.0.0/8 -j DROP : 소스 IP가 10.0.0.0/8인 패킷을 차단한다.

iptables -A INPUT -s 172.16.0.0/12 -j DROP : 소스 IP가 172.16.0.0/12인 모든 패킷을 차단한다.

iptables -A INPUT -s 192.168.0.0/16 -j DROP : 소스 IP가 192.168.0.0/16인 모든 패킷을 차단한다.

iptables -A INPUT -s 224.0.0.0/4 -j DROP : 소스 IP가 224.0.0.0/4인 모든 패킷을 차단한다.

iptables -A INPUT -s 240.0.0.0/5 -j DROP : 소스 IP가 240.0.0.0/5인 모든 패킷을 차단한다.



IANA(http://www.iana.org/)에서 특별한 목적으로 사용하기 위해 예약해 둔 사설 ip 대

역이 있다. 이러한 ip 주소는 RFC1918(www.ripe.net/db/rfc1918.html)에 명시되어 있는데, 특별한 목적으로 사용될 뿐 공인 네트워크인 인터넷에서는 라우팅될 수 없기 때문에 다음과 같이 사설 ip 주소를 소스로 하여 들어오는 트래픽은 위조된 트래픽이므로 차단하여야 한다.

⑥ 루프백(Loopback) IP 주소 차단

iptables -A INPUT -i eth0 -s 127.0.0.0/8 -j DROP : 인터페이스 eth0를 통해 들어오는 127.0.0.0/8 인 IP 주소를 차단한다.

루프백 ip 주소는 내부의 루프백 인터페이스(lo)를 통해서만 통신하기 때문에 루프백 주소를 소스로 해서 eth0과 같이 외부 인터페이스를 통해 들어오는 트래픽은 위조된 트래픽일 가능성이 높으므로, 보안상 차단하는 것이 좋다.

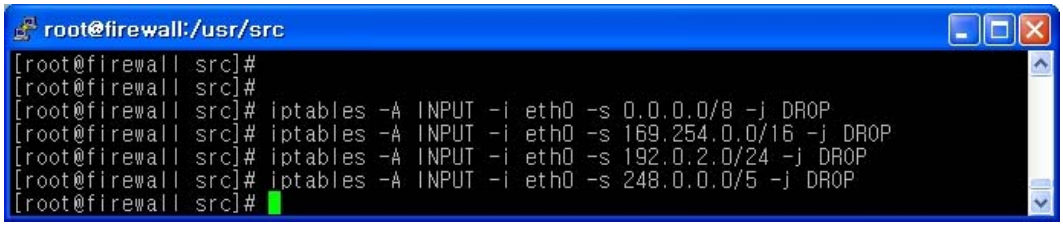
⑦ 예약된 IP 주소 차단

iptables -A INPUT -i eth0 -s 0.0.0.0/8 -j DROP : 인터페이스 eth0를 통한 IP주소가 0.0.0.0/8 인 트래픽을 차단한다.

iptables -A INPUT -i eth0 -s 169.254.0.0/16 -j DROP : 인터페이스 eth0를 통한 IP주소가 0.0.0.0/8 인 트래픽을 차단한다.

iptables -A INPUT -i eth0 -s 192.0.2.0/24 -j DROP : 인터페이스 eth0를 통한 IP주소가 0.0.0.0/8 인 트래픽을 차단한다.

iptables -A INPUT -i eth0 -s 248.0.0.0/5 -j DROP : 인터페이스 eth0를 통한 IP주소가 0.0.0.0/8 인 트래픽을 차단한다.



0.0.0.0/8 과 248.0.0.0/5는 예약된 ip 대역이며 169.254.0.0/16 은 DHCP등에서 임시로 사용하는 대역이며 192.0.2.0/24 는 TEST-NET 대역이다. 기타 차단하고 싶은 IP에 대해 상기와 같은 방법으로 차단하도록 한다.

⑧ SSH 서비스 허용하기

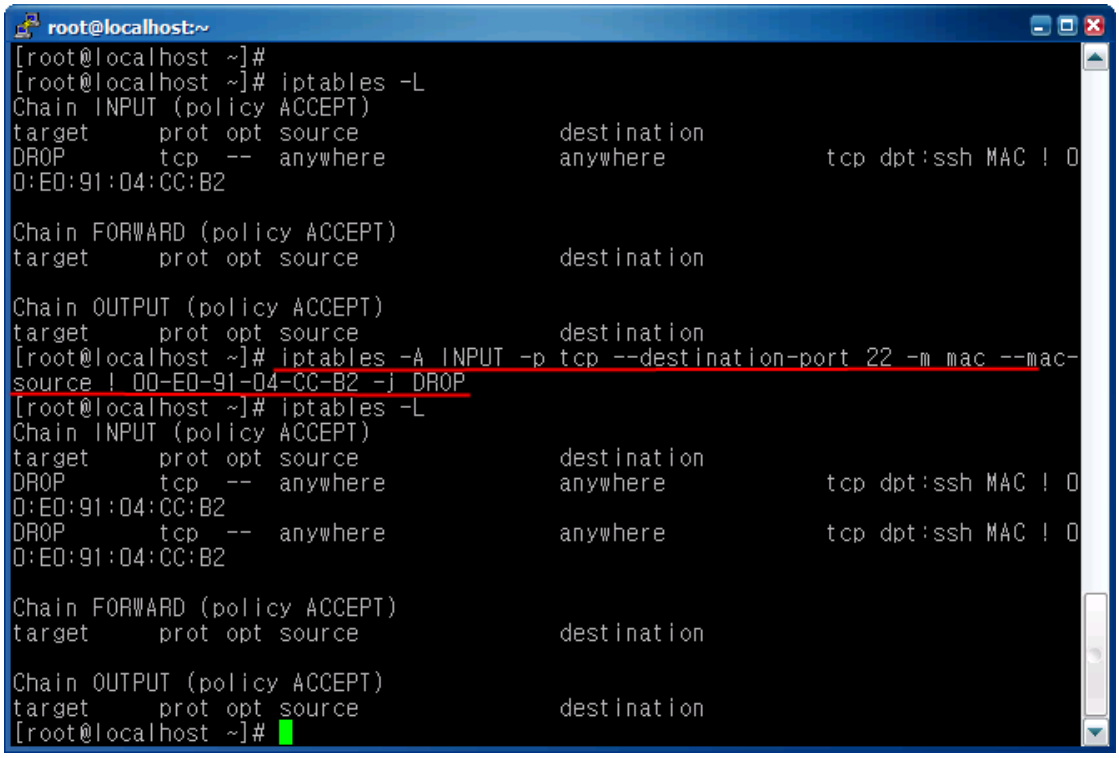
iptables -A INPUT -p TCP -s 192.168.10.11 --sport 1024:65535 --dport 22 -m state --state NEW -j ACCEPT : 소스 IP가 192.168.10.11에서 22번으로 들어오는 TCP 프로토콜을 이용한 트래픽에 대해 허용하며, 상태 추적 기능을 이용한다.

서버에 대한 원격 접속을 위해 최근에는 telnet에 대한 대안으로 세션을 암호화한 ssh가 많이 사용되고 있는 추세이다. ssh는 22/tcp 포트를 사용하므로 목적지 포트 22번에 대한 룰을 설정해 주면된다. ssh 서버는 22번을, 클라이언트는 1024 이후의 임의의 포트를 사용하므로, 방화벽에서 각각의 포트를 허용해 주면된다. 위 그림은 192.168.10.11에서 방화벽 서버의 22/tcp 로의 접근을 허용하는 룰이다. 이후 방화벽에서 192.168.10.11 로 응답하는 패킷은 앞에서 허용한 상태추적 룰에 따라 허용되게 된다. 만약 상태 추적에서 허용하지 않았더라도 OUTPUT의 기본 정책이 ACCEPT이므로 OUTPUT에 대해서는 별도로 생각해 주지 않아도 된다. 만약 모든 소스 IP 에 대해 ssh를 허용하려면 192.168.10.11 부분에 0/0 이나 any를 지정하거나 -s 부분을 빼면 된다.

iptables -A INPUT -p TCP -s 192.168.10.11 --sport 1024:65535 --dport 22 -j ACCEPT : 소스 IP가 192.168.10.11에서 22번으로 들어오는 TCP 프로토콜을 이용한 트래픽을 허용한다.

만약 상태추적을 사용하지 않는다면 위와 같이 -m state 이하 부분을 생략하고 포트 번호만 명시해도 된다. 물론 OUTPUT의 기본정책이 ACCEPT이므로 OUTPUT 은 언급하지 않아도 된다.

iptables -A INPUT -p tcp --destination-port 22 -m mac --mac-source ! 00-AA-BB-04-CC-B2(관리자의 MAC 주소) -j DROP : MAC 주소가 00-AA-BB-04-CC-B2에서만 22번으로 접근 가능, 이외의 모든 MAC 주소를 가진 접근은 차단한다.



관리자의 IP 주소가 일정하지 않을 때는 관리자의 MAC 주소를 이용하여 위와 같이 접근 제한을 할 수 있다.

⑨ 메일 서비스 허용 (SMTP/POP3)

iptables -A INPUT -p TCP ! --sport 0:1024 -dport 25 -m state --state NEW -j ACCEPT : 출발지 port가 25인 TCP 프로토콜을 이용하는 트래픽에 대해 허용하고, 상태 추적 기능을 이용한다.

SMTP 는 25/tcp 를 통해 서비스가 제공되는데, 만약 원격지에서 보내는 메일서버(SMTP) 용도로 허용하여 메일을 보낼 경우에는 외부에서 서버의 25/tcp 로 향하는 트래픽을 허용해 주어야 한다. 이때 메일을 받은 서버는 임시로 큐에 저장했다가 외부로 메일을 발송하게 되는데, 이 경우 내부에서 외부로의 접속이므로 이는 별도의 룰 설정 없이도 허용하게 된다. 위 그림은 외부에서 오는 메일을 받을 수 있도록 25/tcp를 허용한 예인데, 소스포트에서 1024:65535 대신 ! --sport 0:1023 으로 하였는데, 이는 0부터 1023까지가 아니므로 1024:65535 와 동일한 의미가 되는 것이다.

iptables -A INPUT -p TCP --sport 1024: --dport 110 -m state --state NEW -j ACCEPT : Destination port가 110인 TCP 프로토콜을 이용한 트래픽을 허용하고 상태 추적 기능을 사용한다.

pop3 는 110/tcp를 통해 서비스를 제공하는데, 만약 pop3 서비스를 제공한다면 외부에서의 110/tcp 접속도 허용하여야 할 것이다. 위 룰은 소스 ip 에 대한 언급은 없었으므로 모든 ip에 대한 허용이 되고 소스포트는 1204: 이후의 포트이고 목적지 포트는 110/tcp인 패킷을 허용하는 룰이다. 물론 요청에 대한 응답은 상태추적 및 OUPUT 의 기본 정책이 허용이므로 별도로 언급하지 않아도 된다. 만약 특정한 ip 또는 ip 대역에서만 pop3/tcp를 허용한다면 --sport 앞에 "-s 192.168.1.0/24" 와 같이 언급하면 된다.

⑩ FTP 서비스 허용

iptables -A INPUT -p TCP --sport 1024: --dport 21 -m state --state NEW -j ACCEPT : 목적지 포트가 21인 TCP 프로토콜을 사용하는 트래픽에 대해 허용하고, 상태 추적 기능을 이용한다.

FTP 서비스는 다른 서비스와 달리 2가지 모드를 사용하고 2가지 포트를 사용한다. 흔히 FTP 서비스가 사용하는 포트는 21/tcp라고 알고 있으나 21/tcp 외에 추가적으로 다른 포트도 사용한다. 이때 사용하는 포트는 어떤 모드를 사용하는가에 따라 다른데, 이를테면 Active 모드의 경우 20/tcp를 사용하고, Passive 모드의 경우 1024 이후의 임의의 포트가 사용된다. 따라서 각각의 모드에 따라 룰을 따로따로 설정해 보면 매우 복잡한데, 통합적으로 아래와 같은 하나의 룰로 FTP 서비스에 대한 제어를 할 수 있다. 즉, 두 번째 사용되는 포트는 상태추적과 OUTPUT 의 기본정책이 허용됨에 따라 별도로 설정하지 않아도 되는 것이다.

DNS 트래픽 허용 및 ICMP 트래픽 허용은 <별첨 #1>을 참조하도록 하자.

<참고> 방화벽에서 외부로의 접속 허용

앞의 룰은 외부에서 방화벽의 ssh 포트로의 접근을 허용한 경우인데, 반대로 방화벽에서 외부의 다른 서버로 ssh 접근을 허용하려면 어떻게 하여야 할까? 이는 방화벽에서 외부로 나가는 것이므로 먼저 OUTPUT을 생각하면 되는데, OUTPUT 은 기본적으로 ACCEPT 이므로 고려하지 않아도 되고, 요청에 대한 응답은 INPUT 이지만 상태추적에 의해 ESTABLISHED 는 허용되어 있으므로 방화벽에서 외부로의 접속은 기본적으로 허용되어 있다. 따라서 별도의 룰 설정 없이도 접속이 가능하게 된다.

지금까지 방화벽을 활용하기 위해 필요한 iptables의 문법과 활용 및 여러 가지 개념에 대해 알아보고 직접 각각의 서비스에 대한 룰도 작성해 보았는데, 이러한 것들을 통합하여 자체적으로 운영중인 서버에 방화벽을 직접 설정해 보도록 하자. 지

금까지 알아본 룰은 원격으로 연결된 상태에서 하나하나 실행하여 적용하지 말고, 스크립트 파일을 이용하여 일괄적으로 설정 후 사용하도록 하는 것이 실수를 줄일 수 있다. 스크립트 파일은 <별첨 #2>를 참조하자.

Ⅲ. NAT 방화벽 설정

앞에서는 웹이나 메일 등 서비스 중인 서버에 직접 iptables를 설치하여 서비스를 제공하면서 패킷을 필터링하는 방법에 대해 알아보았다. 하지만 서비스중인 한 서버를 보호하는 것이 아니라 여러 대의 서버나 Windows 또는 유닉스 등 다른 운영체제를 사용하는 서버 또는 전체 네트워크에 방화벽을 설치하여 보호하고자 할 때에는 NAT나 브리지 방식을 사용하여야 한다. 이번 절에서는 NAT 방식에 대해 알아보고 직접 구현해 보도록 한다.

1. NAT 구성 방법

NAT가 처음 소개된 지는 10년 이상 되었지만 지금도 여전히 대중적으로 사용되고 있는 기술 중 하나이다. NAT를 통해 갈수록 부족해지는 ip 주소의 부족 현상을 해결할 수 있고, 또한 실제 서버는 외부에서 직접 접근이 불가능한 사설 ip를 사용함으로써 보안을 강화할 수 있다는 측면에서 장점을 가지고 있기 때문이다. 그러나 NAT를 사용할 경우 NAT 방화벽과 같이 게이트웨이 역할을 하는 하나의 장비에 내부의 네트워크가 모두 연결되어 있어 트래픽이 몰릴 수 있고, 만약 NAT 게이트웨이 장비에 장애가 있을 경우에는 이하의 모든 시스템에 영향을 줄 수 있다는 단점도 있다.

NAT 는 구성 형태에 따라 크게 두 가지 형태로 나뉠 수 있다. 첫 번째는 가장 전통적인 방식으로서 내부에서는 사설 ip를 사용하면서 외부의 네트워크로 접속할 때는 방화벽을 거치면서 내부의 사설 ip가 아니라 방화벽에서 접속한 것처럼 보이게 된다. 그리고 외부의 호스트가 이 요청에 응답하면 응답 패킷을 받은 방화벽은 해당하는 응답 패킷을 요청했던 내부의 PC로 전달해 준다. 이러한 방법으로 내부 네트워크에서는 사설ip를 사용하여 외부에서 보이지 않으면서도 외부로 연결되어 응답을 받을 수 있게 되므로 내부 네트워크에 있는 컴퓨터가 사설 ip를 사용하더라도 인터넷에 접속할 수 있는 것이다. 게다가 공인 ip 1개만으로도 내부에서는 무제한으로 인터넷을 이용할 수 있게 된다. 이와 같은 기능을 'masquerading'한다고 하며 NAT의 가장 기본적인 기능이기도 한다.

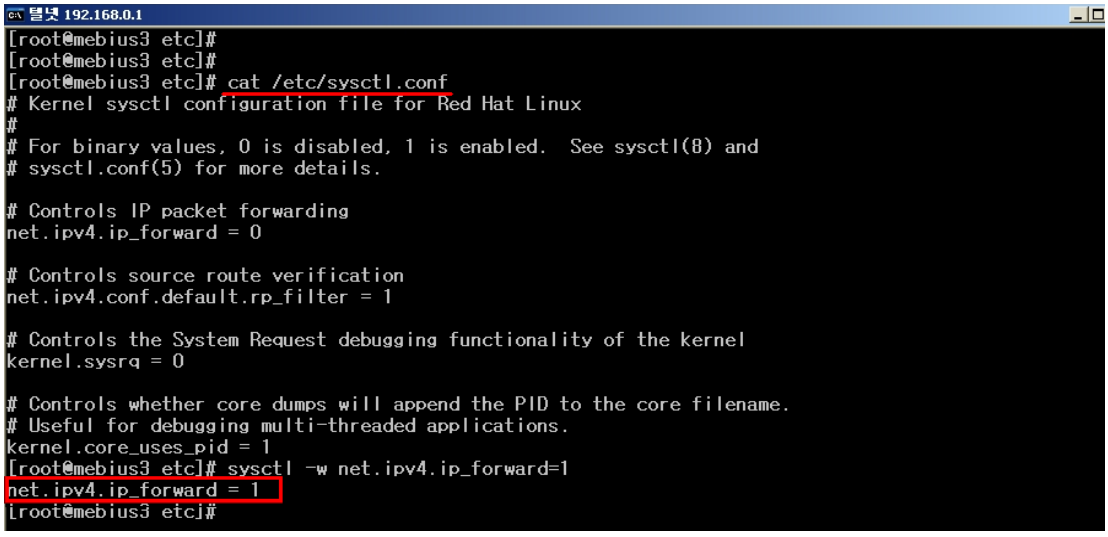
두 번째는 일종의 패킷 포워딩 기능으로 외부에서 내부로 접속을 할 수 있도록 하는 특징이 있다. 따라서 사설 ip 환경에서 서버를 운영한다면 포트 포워딩 기능을 이용하면 된다. 이를테면 공인 ip가 1개 있고 내부에 사설 ip로 설정된 3대의 서버가 있을 경우 해당 공인 ip 의 25번으로 오는 패킷은 첫 번째 사설 ip 서버의 25번으로 포워딩 하고, 80번으로 오는 패킷은 두 번째 사설 ip 서버로 포워딩 해 주고, udp 53번으로 오는 패킷은 세 번째 사설ip 서버로 포워딩 해 주도록 설정하게 되면 충분히 사설ip로도 포트의 개수만큼 많은 서비스를 제공할 수 있게 되는 것이다. masquerading은 패킷의 소스 주소를 변경(사설->공인)하므로 Source NAT(SNAT)

라고 하며 포트 포워딩은 패킷의 목적지 주소를 변경(공인->사설)하므로 Destination NAT(DNAT) 이라고 한다.

2. NAT 룰 설정하기

이제 본격적으로 NAT를 설정하여 보도록 하자. 앞에서 살펴보았던 filter 테이블의 INPUT chain에서는 방화벽 서버 자체를 목적지로 하는 패킷에 대한 필터링을 담당하는데 반해 방화벽 내부의 서버에 대한 필터링은 들어오든 나가든 관계없이 FORWARD chain에서만 담당한다. 그리고, NAT를 하기 위해서는 NAT 테이블에 있는 PREROUTING과 POSTROUTING chain을 이용하면 된다. 그리고 방화벽 내부에서 외부로 나가는 패킷이나 외부에서 내부로 들어오는 패킷은 방화벽을 통과하면서 패킷을 포워딩(일종의 라우팅)해 주어야 하므로 커널에서 ip 패킷 포워딩 기능이 지원되어야 한다.

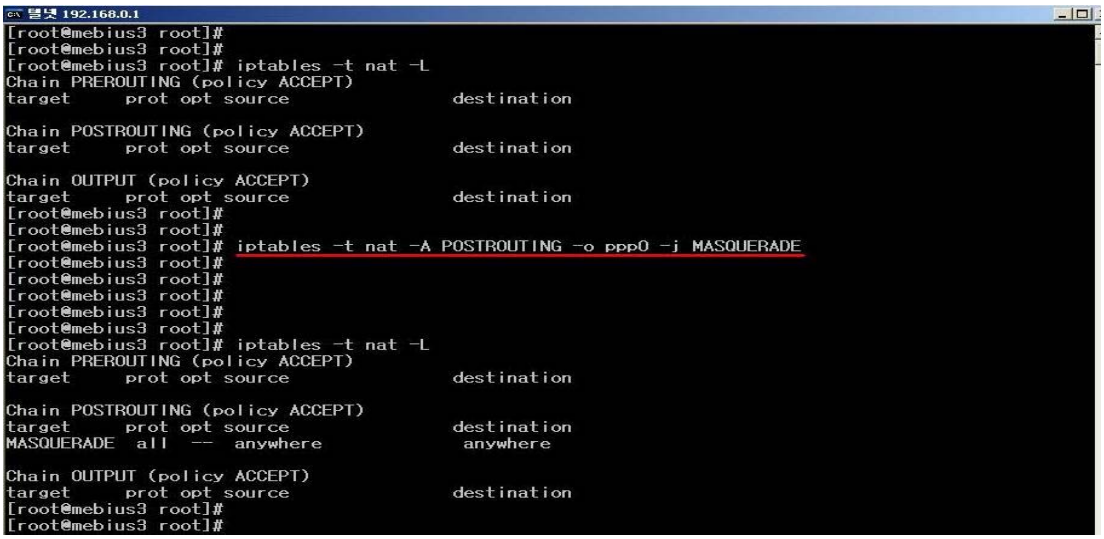
STEP 1. 패킷 포워딩 기능 설정



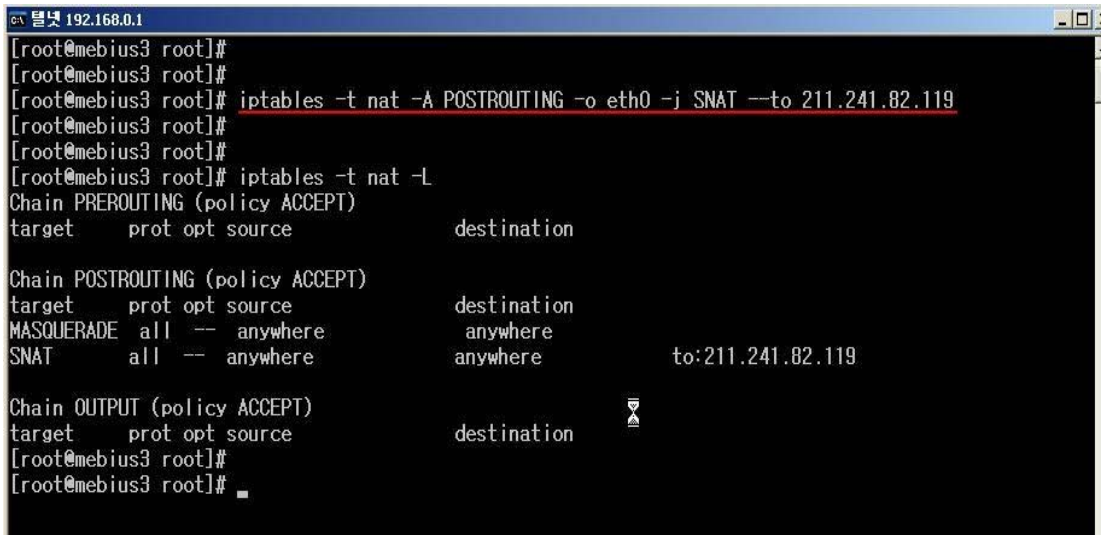
cat /etc/sysctl.conf : /etc/sysctl.conf 파일의 내용을 확인한다.
sysctl -w net.ipv4.ip_forward=1 : net.ipv4.ip_forward=1로 수정하여 포워딩 기능을 활성화 한다.

위와 같이 패킷포워딩 기능을 설정한다. 만일 nat가 영구적으로 패킷 포워딩이 지원되게 하려면 etc/sysctl.conf에서 net.ipv4.ip_forward = 1로 수정해 주어야 한다. (SNAT이든 DNAT이든 관계없이 NAT를 사용한다면 설정값은 반드시 1이어야 한다.)

STEP 2. SNAT (Source NAT) 구현하기



iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE



iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 211.241.82.119

사무실 등에서 서버를 운영하지는 않고 단지 인터넷만 사용할 수 있도록 하기 위해 소위 인터넷 공유를 이용하여 여러 pc에서 인터넷을 사용하는 경우에 대해 알아보자. 만약 ADSL과 같이 유동ip인 경우, 방화벽의 ppp0 인터페이스에는 DHCP 등을 이용한 공인 ip를 설정하고 내부의 네트워크와 연결된 eth1 인터페이스에는 사설 ip의 게이트웨이인 192.168.1.1 로 설정하기로 한다. 위의 룰은 SNAT 설정으로 여기에서 별도로 공인 ip가 설정되지 않은 이유는 DHCP를 이용한 유동ip 이므로 ip를 알 수 없기 때문이다. 만약 공인 ip가 고정 ip 라면 대신 위와 같이 실행하면 된다. ppp0 인터페이스를 통해 나가는 모든 트래픽은 방화벽에서 마스커레이드 되며 이때 ip는 ppp0 인터페이스에 설정된 ip가 된다. 만약 FORWARD chain이 기본 정책

으로 ACCEPT라면 위의 한 줄만으로 masquerading은 끝난다.
. 사실상 내부 네트워크는 사설 ip를 사용하여 외부에서 내부로 직접 접근할 수 없으므로 FORWARD의 기본 정책을 DROP으로 할 필요는 없으며 ACCEPT로 해도 무방하다.

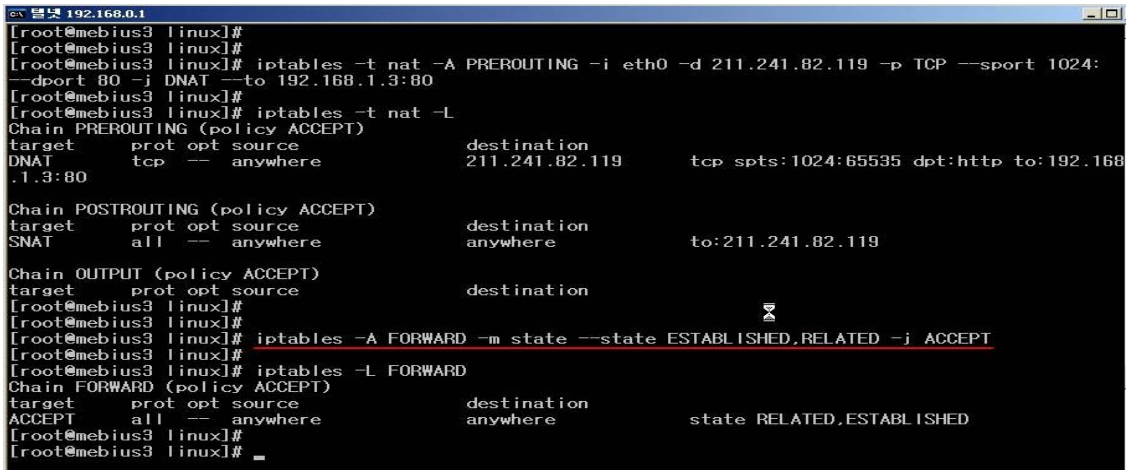
STEP 3. DNAT (Destination NAT) 구현하기

이번에는 DNAT를 이용한 포트 포워딩에 대해 알아본다. DANT의 작동방식은 다음과 같다. 외부에서의 접속 요청이 방화벽에 도착하면 방화벽은 PREROUTING chain을 통해 목적지 주소를 사설 ip로 변경하여 사설 네트워크가 연동된 인터페이스를 통해 패킷을 내부의 서버에 포워딩한다. FORWARD chain 을 통과한 이 패킷을 받은 서버가 응답하게 되면 역으로 방화벽에서 소스 주소를 원래의 외부 인터페이스 주소로 변경하여 원격지 클라이언트에 패킷을 포워딩 해주게 된다. 그럼 여기에서 http 트래픽을 포워딩 하는 예를 보면 DNAT를 설정할 때에는 NAT 테이블을 설정한 후 filter 테이블의 FORWARD 설정을 하면 된다.
그 다음은 filter 테이블을 설정할 차례이다. 여기에서 filter 테이블의 FORWARD chain 이 기본 정책(-P)으로 ACCEPT 라면 추가적인 룰이 필요 없다. 만약, DROP 인 경우에는 상태추적을 이용하여 이미 연결된 접속에 대해서는 허용하도록 한다.

iptables -P FORWARD ACCEPT

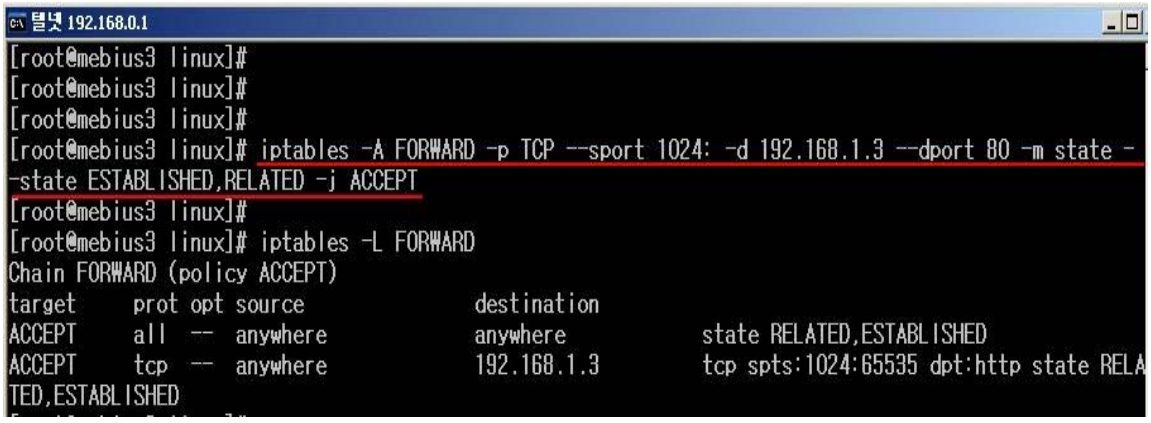
추가적으로 웹 접속 요청을 허용하여야 하므로 그림과 같이 소스 포트가 1024 이후의 포트이면서 목적지 주소가 192.168.1.3 이고 목적지 포트가 80인 패킷을 허용하면 된다.

iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT



그리고 외부에서의 요청을 허용하여야 하므로 상태는 NEW가 된다. 이후의 패킷 교환은 위의 룰을 통해 허용된다.

iptables -A FORWARD -p TCP --sport 1024: -d 192.168.1.3 --dport 80 -m state --state ESTABLISHED,RELATED -j ACCEPT



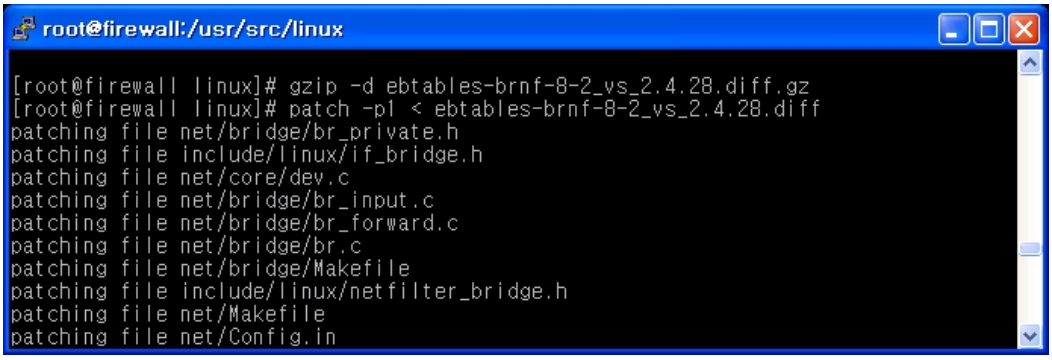
NAT를 이용한 방화벽 룰 예제는 <별첨 #3>을 참고 한다.

Ⅳ. 브리지(Bridge) 방화벽 설정하기

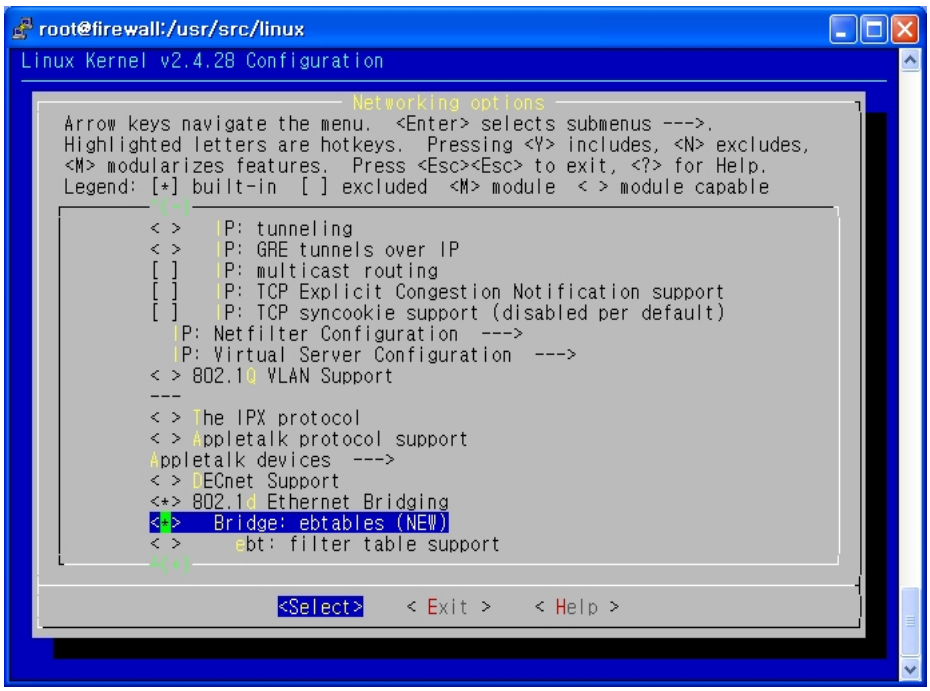
브리지 방화벽은 NAT 방식과 구조는 동일하지만 방화벽 내부의 서버가 사설 ip를 사용하지 않고 공인 ip를 그대로 사용한다는 차이가 있다. NAT 방화벽은 라우터라고 생각하면 되고, 브리지 방화벽은 스위치라고 생각하면 된다. NAT 방식의 경우 만약 방화벽에 장애가 발생했을 경우에는 대책이 없지만, 브리지 방식의 경우 장애가 발생하면 내부의 케이블만 상위 스위치에 연결하면 바로 서비스 재개가 가능하다는 장점이 있다. 이러한 장점 때문에 최근에 사용이 늘고 있는 브리지 방화벽을 설정하고 이용하는 방법에 대해 알아본다.

STEP 1. 브리지 구성

브리지 방화벽은 NAT와 같이 **이더넷 카드가 2장** 있어야 하는데, 이 두 장의 카드는 **0.0.0.0** 으로 설정하고 이 두 장의 이더넷 카드를 합쳐서 하나의 가상 인터페이스로 설정하게 된다. 따라서 방화벽을 기준으로 내부와 외부 케이블이 eth0 이나 eth1등 어떤 인터페이스에 연결되어도 관계없다. 참고로, NAT 의 경우 외부 케이블은 공인 ip가 할당된 인터페이스에, 내부 케이블은 사설 ip 가 할당된 인터페이스에 연결하여야 했다. 이렇듯 리눅스 시스템을 브리지 형태로 구성하려면 2.4 버전에서는 커널패치를 해준 뒤 커널 메뉴에서 **브리지 지원 메뉴**를 별도로 선택해 주고, 2.6 이후버전에서는 커널 자체에서 지원함으로 별도의 커널패치를 하지 않아도 된다.



2.4 커널의 패치를 위해서는 커널 소스 디렉토리인 **/usr/src/linux/** 디렉토리로 이동한 후 http://ebtables.sourceforge.net 에서 자신의 커널 버전에 맞는 패치 파일을 다운로드하도록 한다. 이후 이 파일을 **gzip -d** 로 압축해제 후 패치하면 된다. 이후 make menuconfig를 실행한 후 Networking options에서 확인해 보면 아래와 같이 802.1d Ethernet Bridging 를 선택하면 Bridge: ebtables 라는 것이 추가된 것을 알 수 있으며 이 메뉴를 선택하면 된다.

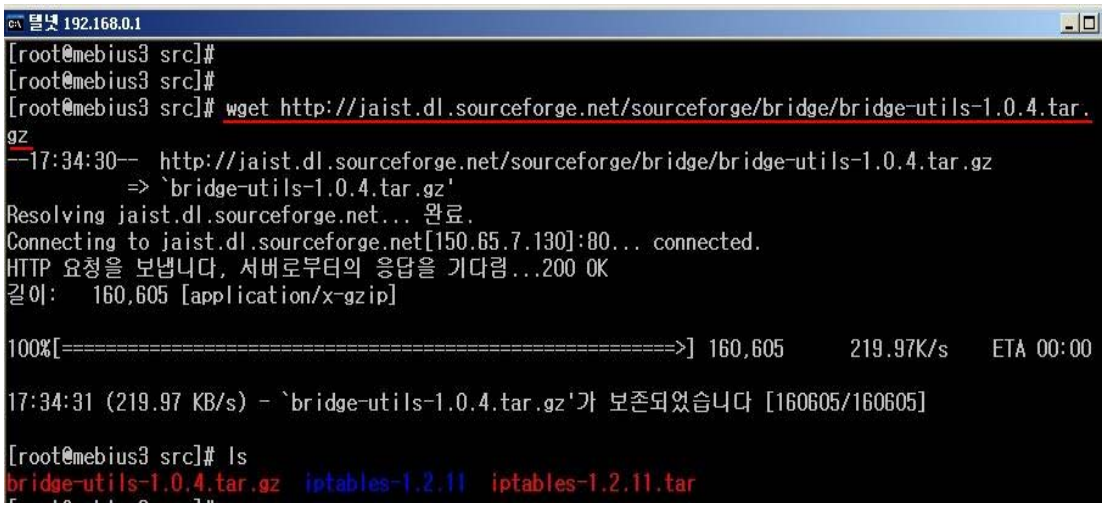


[그림 6] 커널 설정 화면

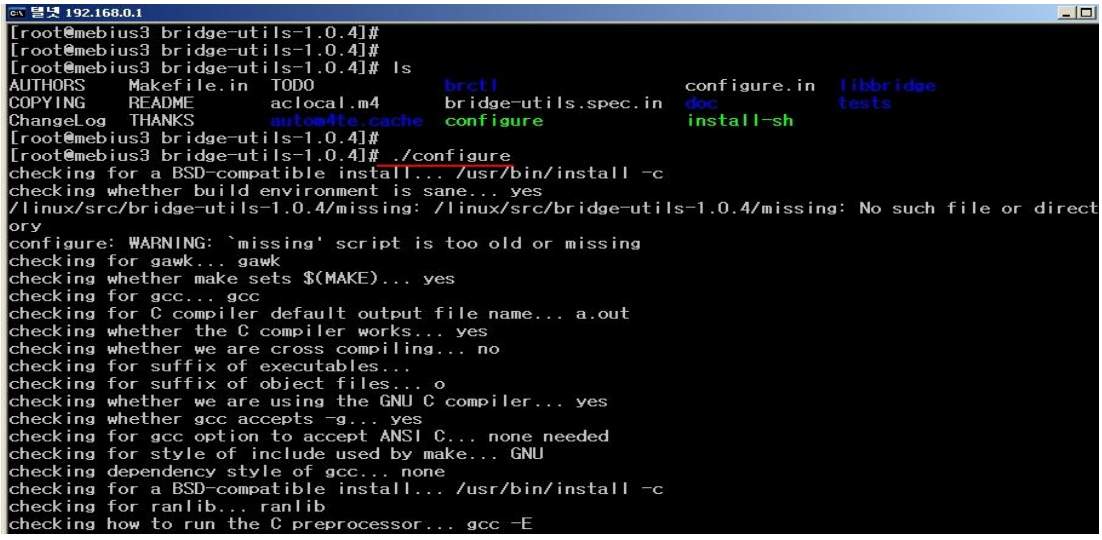
이후 커널컴파일 및 재부팅하면 사용할 준비가 되었다. 다음으로는 브리지를 설정하기 위한 관리 프로그램인 bridge-utils를 설치하도록 한다.

☞ bridge-utils 다운로드 : http://bridge.sourceforge.net

이후 아래와 같이 압축 해제 후 ./configure; make 로 컴파일만 하면 된다. 이후 생성된 **brctl** 실행파일을 /sbin/으로 옮기도록 한다.



[그림 7] 브리지 유틸리티 압축 해제



[그림 8] 브리지 유틸리티 환경 설정

```
[root@firewall bridge-utils-x.x.x]# make
[root@firewall bridge-utils-x.x.x]# mv brctl/brctl /sbin/
```

STEP 2. brctl 설정하기

bridge-utils인 brctl을 설치한 후 실행하면 사용할 수 있는 옵션이 많이 있는데, 브리징이나 스패닝트리(stp)등에 대한 자세한 설명은 네트워크 관련 서적을 참고하기 바라며 실제 브리지 방화벽을 운영할 때는 아래와 같은 옵션만 알고 있으면 된다. 실제 아래와 같은 스크립트를 만들어 br.sh와 같은 파일로 작성 후 부팅할 때 자동으로 실행하도록 설정한다.

```
#!/bin/sh

SERVICE_IP="221.1.2.3"
#- 방화벽에서 사용할 공인 ip를 지정한다. 각자의 공인 ip를 설정하면 된다.
GATEWAY_IP="221.1.2.1"
#- 방화벽에서 사용할 게이트웨이 ip를 지정한다. 각자의 게이트웨이 ip를 설정하면 된다.

/sbin/brctl addbr bridge
/sbin/brctl stp bridge on
/sbin/brctl addif bridge eth0
/sbin/brctl addif bridge eth1
/sbin/ifconfig eth0 down
/sbin/ifconfig eth1 down
```

```
/sbin/ifconfig eth0 0.0.0.0 promisc up
/sbin/ifconfig eth1 0.0.0.0 promisc up

#- 이후 아래 부분은 각자의 상황에 따라 설정한다.

/sbin/ifconfig lo 127.0.0.1 up
#- 루프백 인터페이스에 ip 설정.
/sbin/ifconfig bridge $SERVICE_IP promisc up
#- br0 인터페이스의 ip 를 정의한다.
/sbin/route add -host 127.0.0.1 dev lo
#- 이후 127.0.0.1을 라우팅 테이블에 추가한다.
/sbin/route add default gw $GATEWAY_IP
#- 각자의 공인 게이트웨이 ip를 라우팅 테이블에 정의한다.
```

위와 같이 설정 후 ifconfig를 실행하면 다음과 같이 보일 것이다.

```
[root@firewall root]# ifconfig
bridge    Link encap:Ethernet  HWaddr 00:02:B3:23:46:A2
          inet addr:221.1.2.3  Bcast:221.1.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:78976119 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1280633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19496170 (1163.0 Mb)  TX bytes:139611162 (133.1 Mb)

eth0      Link encap:Ethernet  HWaddr 00:02:B3:23:46:A2
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1433779958 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1244370418 errors:0 dropped:0 overruns:10 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3970825670 (3786.8 Mb)  TX bytes:3927975228 (3746.0 Mb)
          Interrupt:7

eth1      Link encap:Ethernet  HWaddr 00:02:B3:23:46:A3
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1245227725 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1278748119 errors:0 dropped:0 overruns:8325 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:154061880 (146.9 Mb)  TX bytes:2212479690 (2109.9 Mb)
          Interrupt:7 Base address:0x2000
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12390 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12390 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16320338 (15.5 Mb)  TX bytes:16320338 (15.5 Mb)
```

route -n 이나 netstat -r 을 실행하면 아래와 같이 bridge 인터페이스가 보일 것이다.

```
[root@firewall root]# route -n
127.0.0.1       0.0.0.0         255.255.255.255 UH    0      0        0 lo
221.1.2.0       0.0.0.0         255.255.255.0   U     0      0        0 bridge
0.0.0.0         221.1.2.1       0.0.0.0         UG    0      0        0 bridge
```

현재 브리지 상태를 조회하려면 다음과 같이 brctl show를 실행하면 된다.

```
[root@firewall root]# brctl show
bridge name     bridge id               STP enabled     interfaces
bridge          8000.00d0b79a2b6c       yes             eth0
                                                        eth1
```

여기에서 만약 브리지를 해지하고 재설정하려면 먼저 해당 브리지에 설정되어 있는 인터페이스를 삭제한 후 브리지를 삭제하면 된다. 물론 원격접속을 한 상태라면 네트워크가 끊기게 되므로 주의하여야 한다. 아래와 같이 스크립트 파일을 생성한 뒤 원격접속이 아닌 리눅스 콘솔상에서 실행한다.

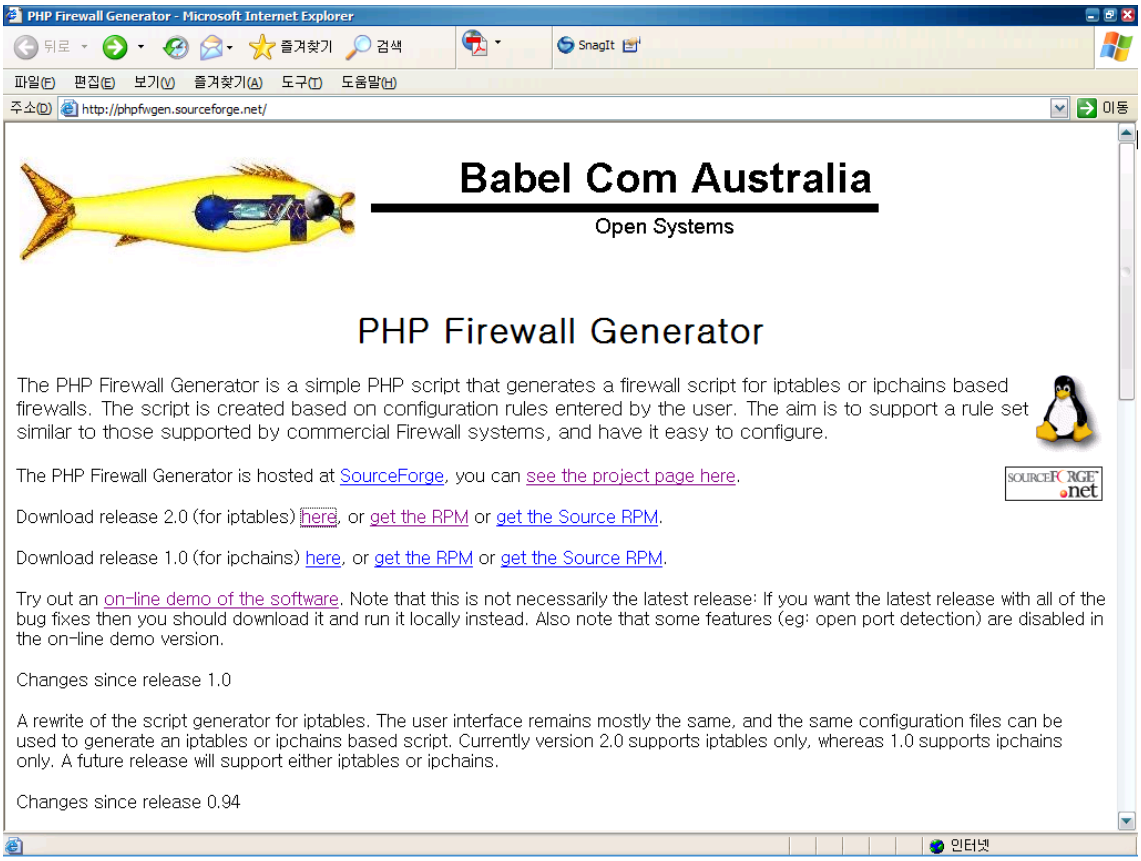
```
#!/bin/sh

ifconfig bridge down
brctl stp bridge off
brctl delif bridge eth0
brctl delif bridge eth1
brctl delbr bridge
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth0 221.1.2.3 promisc up
ifconfig eth1 192.168.1.1 promisc up
```

브리지 방화벽 룰은 <별첨 #4>를 참고한다.

Ⅳ. phpfwgen을 이용한 iptables 사용하기

지금까지 iptables를 이용하여 보안정책을 설정하는 법을 알아보았지만, 수많은 보안정책을 수립하는데는 많은 시간과 노력이 필요함을 느끼는 사용자들이 많다. 또한 수많은 정책을 설정하다보면, 일관성이 없어지고 많은 부분이 실행이 제대로 안되어 문제가 발생할 수 있다. 이런 사용자를 위해 좀 더 손쉽게 iptables script 생성을 위한 툴이 많이 존재하고 있으며, 그 중에 웹 환경에서 사용할 수 있는 phpfwgen에 대해 알아보도록 하겠다.

☞ phpfwgen 홈페이지 : http://phpfwgen.sourceforge.net



1. phpfwgen의 기능 및 특징

phpfwgen는 오픈소스 프로젝트 중 하나로서 iptables를 이용한 보안 정책을 손쉽게 수립할 수 있게 해주는 툴로써 아래와 같은 기능 및 특징이 있다.

가. 웹 상에서 GUI 형태로 구성되어 있어 간편하게 사용할 수 있다.
나. 기본적인 서비스명이나 포트 번호 등에 대해 정의되어 있다.

다. 기본으로 제공되는 서비스명이나 포트 번호 이외에 사용자 정의로 추가할 수 있다.
라. RPM 등을 이용할 수 있으며, 설치가 용이하다.

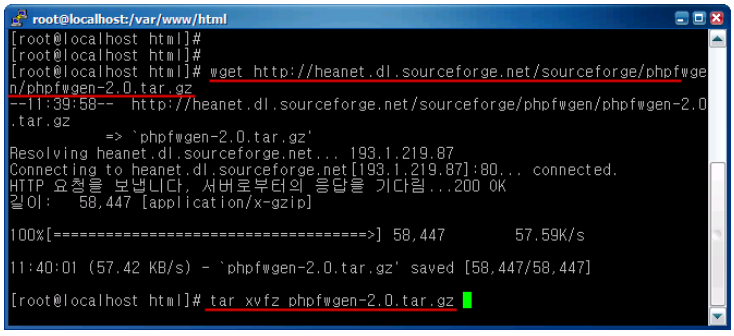
2. phpfwgen 설치 및 사용하기

phpfwgen를 사용하기 위해서는 php를 사용할 수 있는 아파치 웹 서버 환경이 마련되어야 한다. 설치나 사용하는 부분이 별 어려움이 없으므로, 간략하게 살펴보도록 하자. 아래의 순서로 설치하고, 사용하면 별 문제 없이 사용할 수 있다.

STEP 1. phpfwgen 다운로드 및 설치하기

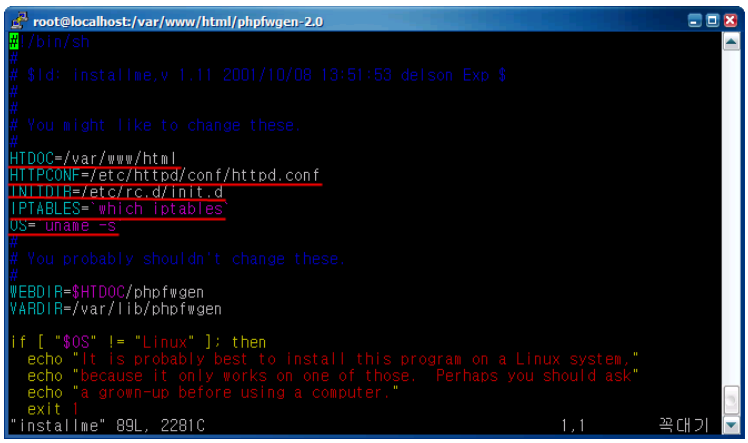
① phpfwgen 다운로드
홈페이지에서 다운로드 받기.

wget http://heanet.dl.sourceforge.net/sourceforge/phpfwgen/phpfwgen-2.0.tar.gz (wget을 이용한 다운로드)
tar xvfz phpfwgen-2.0.tar.gz (압축 풀기)



② 환경 설정
installme 파일을 시스템의 환경에 맞게 수정한다.

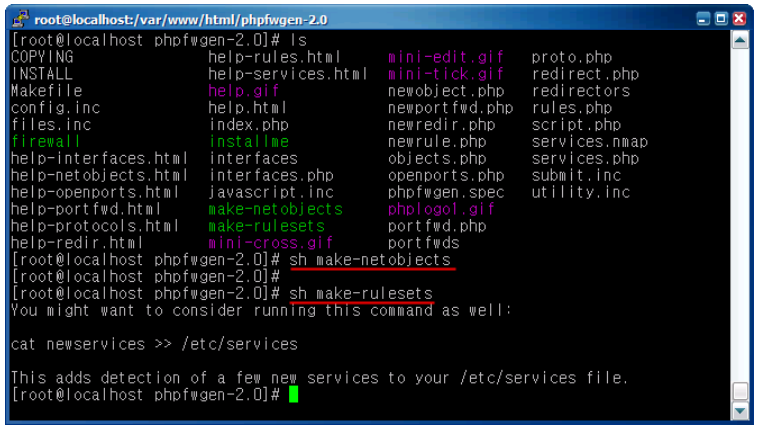
cd phpfwgen-2.0 (디렉토리 이동)
vi installme (환경 설정)



③ 설치하기

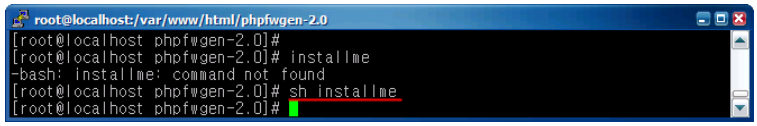
sh make-netobjects (netobjects 생성하기)

sh make-rulesets (rulesets 생성하기)



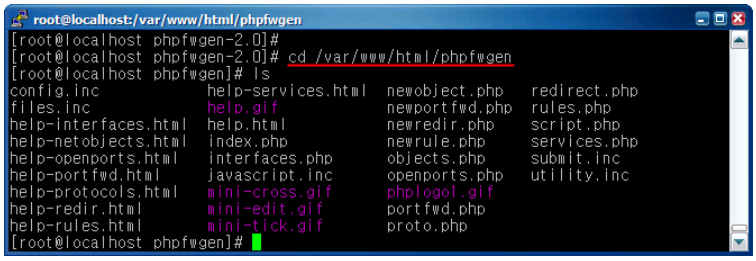
상기의 명령어를 이용하여 netobjects 파일 및 rulesets 파일을 필히 생성해야 한다.

sh installme (설치 하기)



④ 설치 종료 및 확인하기

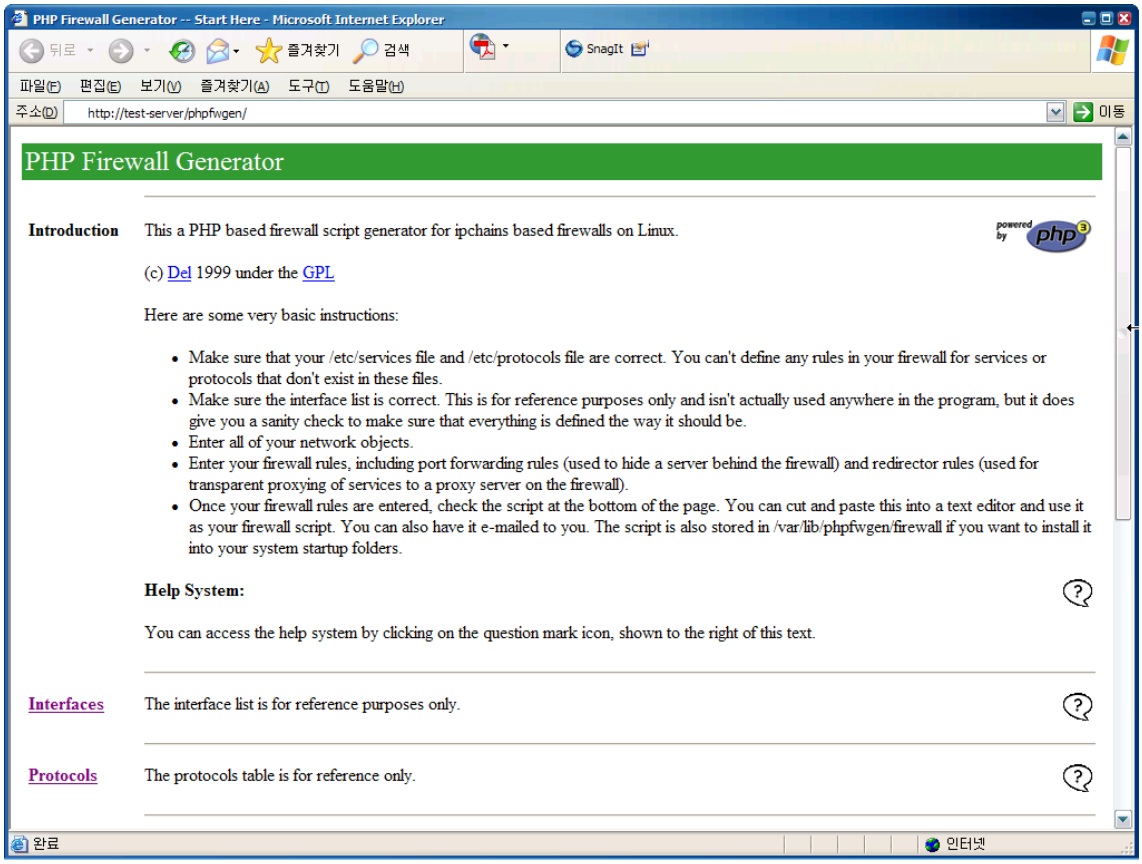
cd /var/www/html/phpfwgen (설치디렉토리로 이동하기)



설치 디렉토리는 자신이 설정한 디렉토리이며, 확인하도록하자.

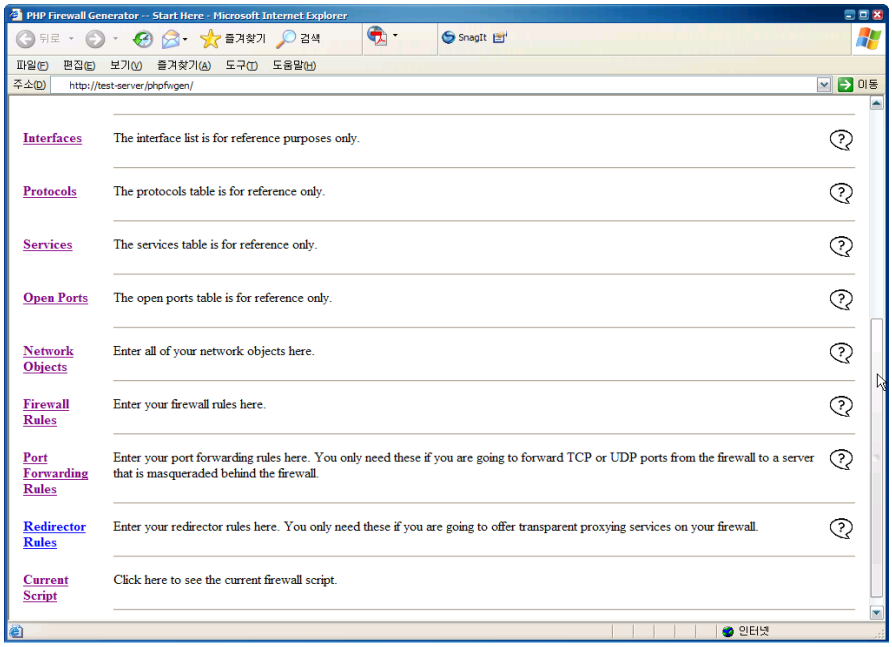
⑤ 사용하기

웹 브라우저로 http://test-server/phpfwgen/으로 접근하였을 때, 아래와 같은 페이지가 확인이 된다면, 설치가 제대로 되었다고 할 수 있다.



STEP 3. iptables script 만들기

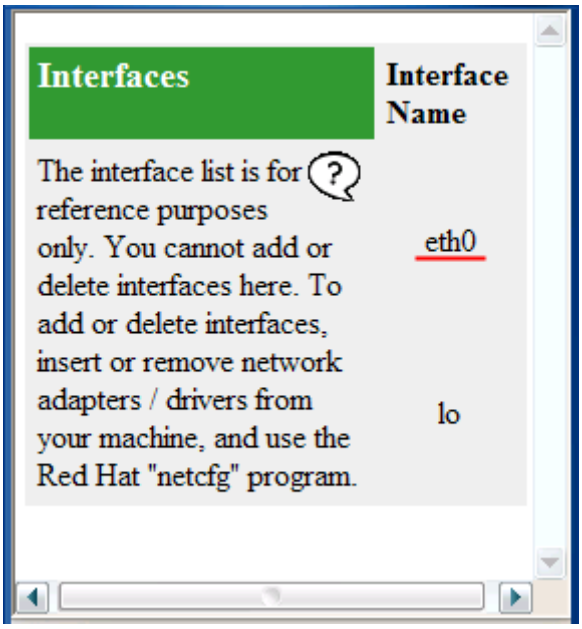
설치가 끝났으면, 이제 사용하는 법에 대해 알아보도록 하자. 사용하는 방법도 설치만큼이나 쉬우므로, 별 어려움이 없을 것이다. 메뉴 화면은 아래와 같다.



메뉴에 대해 아래와 같이 확인 및 사용을 하도록 하자.

① Interface 확인하기

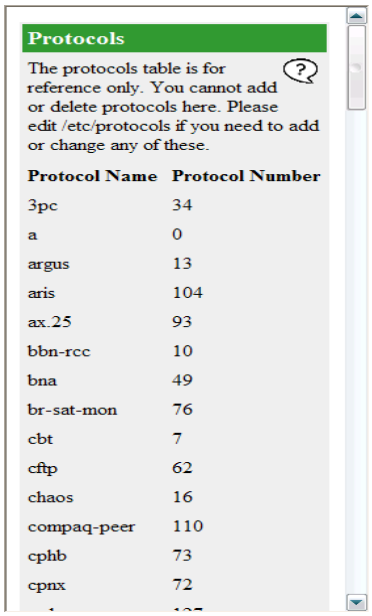
메뉴 중 Interfaces 라는 부분을 클릭하게되면 아래와 같이 현재 시스템에서 사용하고 있는 Interface 정보를 확인할 수 있다.



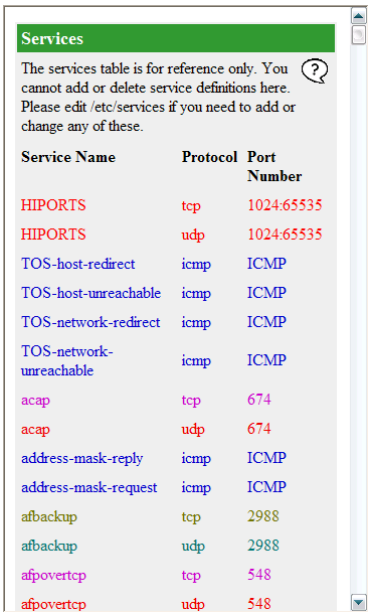
② Protocols, Services, Open Ports 확인하기

Protocols, Services, Open Ports 메뉴는 기본적인 Reference 정보를 제공하는 것으로 참고 자료로 확인하자.

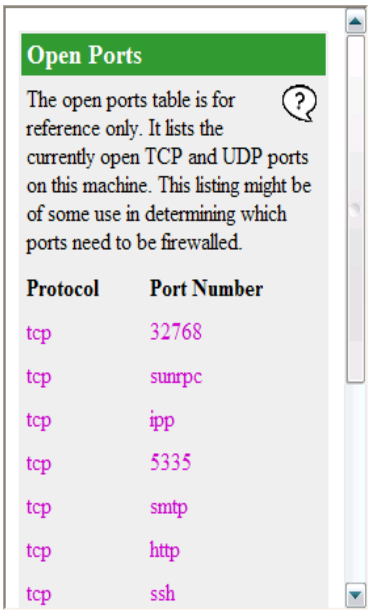
Protocols



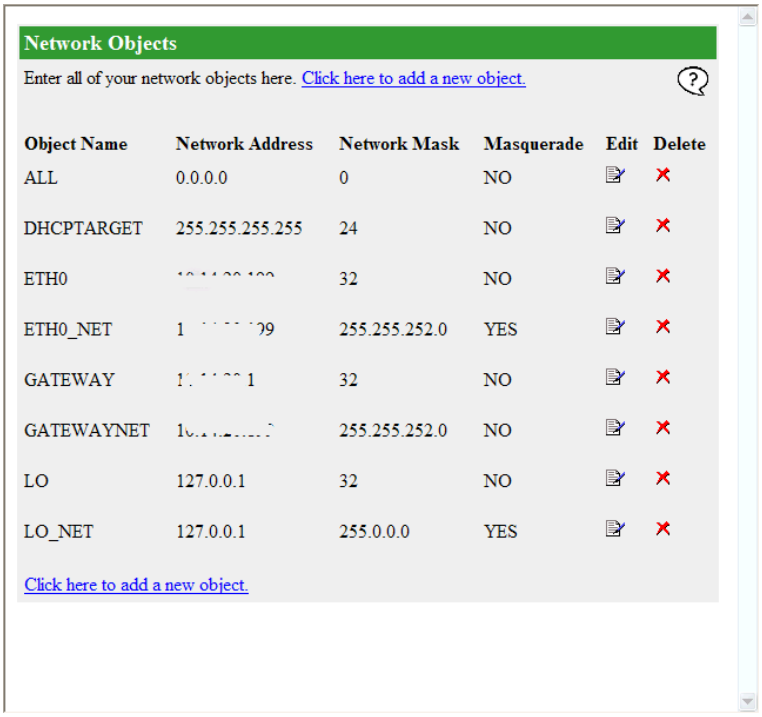
Services



Open Ports



③ Network Objects 관리하기



Network Objects

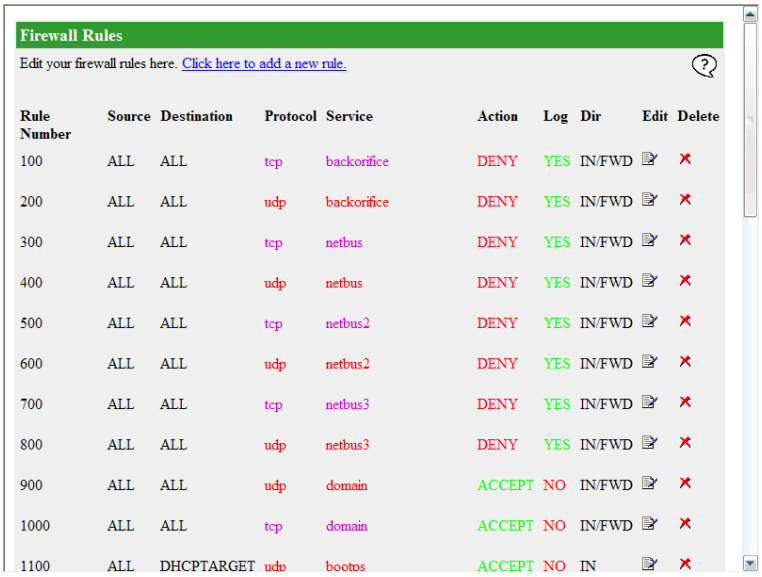
Enter all of your network objects here. Click here to add a new object.

Object Name	Network Address	Network Mask	Masquerade	Edit	Delete
ALL	0.0.0.0	0	NO		
DHCPTARGET	255.255.255.255	24	NO		
ETH0	[redacted]	32	NO		
ETH0_NET	1 [redacted] 99	255.255.252.0	YES		
GATEWAY	1 [redacted] 1	32	NO		
GATEWAYNET	1 [redacted]	255.255.252.0	NO		
LO	127.0.0.1	32	NO		
LO_NET	127.0.0.1	255.0.0.0	YES		

Click here to add a new object.

Network Objects 메뉴를 클릭하면 다음과 같이 설정된 Object에 대한 정보를 확인하거나, “Click here to add a new object"를 클릭하여 object를 추가할 수 있다.

④ Rules 확인 및 설정하기



Firewall Rules

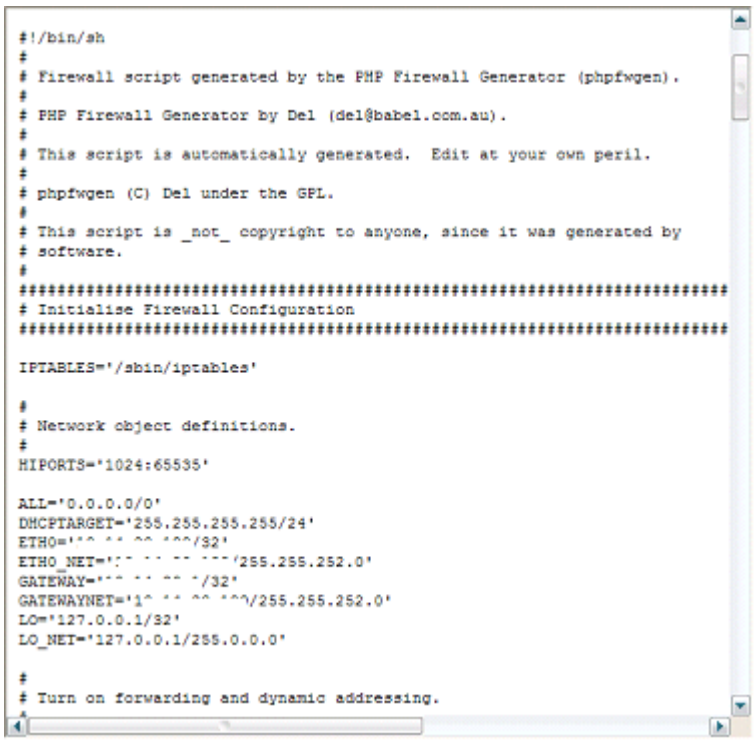
Edit your firewall rules here. Click here to add a new rule.

Rule Number	Source	Destination	Protocol	Service	Action	Log	Dir	Edit	Delete
100	ALL	ALL	tcp	backorifice	DENY	YES	IN/FWD		
200	ALL	ALL	udp	backorifice	DENY	YES	IN/FWD		
300	ALL	ALL	tcp	netbus	DENY	YES	IN/FWD		
400	ALL	ALL	udp	netbus	DENY	YES	IN/FWD		
500	ALL	ALL	tcp	netbus2	DENY	YES	IN/FWD		
600	ALL	ALL	udp	netbus2	DENY	YES	IN/FWD		
700	ALL	ALL	tcp	netbus3	DENY	YES	IN/FWD		
800	ALL	ALL	udp	netbus3	DENY	YES	IN/FWD		
900	ALL	ALL	udp	domain	ACCEPT	NO	IN/FWD		
1000	ALL	ALL	tcp	domain	ACCEPT	NO	IN/FWD		
1100	ALL	DHCPTARGET	udp	bootps	ACCEPT	NO	IN		

Firewall Rules 메뉴를 클릭하면 보안 정책을 확인 및 수정할 수 있다. 일반적인 보안 정책이 기본적으로 설정 되어 있으므로, 자신의 환경에 맞게 수정하거나 삭제하여 사용하도록 하자.

“Port Forwarding Rules”, “Redirector Rules” 메뉴는 해당 사용자만 설정하여 사용하도록 하자.

⑤ Current Script 메뉴를 이용한 iptables 스크립트 만들기



```
#!/bin/sh
#
# Firewall script generated by the PHP Firewall Generator (phpfwgen).
#
# PHP Firewall Generator by Del (del@babel.com.au).
#
# This script is automatically generated.  Edit at your own peril.
#
# phpfwgen (C) Del under the GPL.
#
# This script is _not_ copyright to anyone, since it was generated by
# software.
#
################################################################################
# Initialise Firewall Configuration
################################################################################

IPTABLES='/sbin/iptables'

#
# Network object definitions.
#
HIPORTS='1024:65535'

ALL='0.0.0.0/0'
DHCPTARGET='255.255.255.255/24'
ETH0='[illegible]/32'
ETH0_NET='[illegible]/255.255.252.0'
GATEWAY='[illegible]/32'
GATEWAYNET='[illegible]/255.255.252.0'
LO='127.0.0.1/32'
LO_NET='127.0.0.1/255.0.0.0'

#
# Turn on forwarding and dynamic addressing.
```

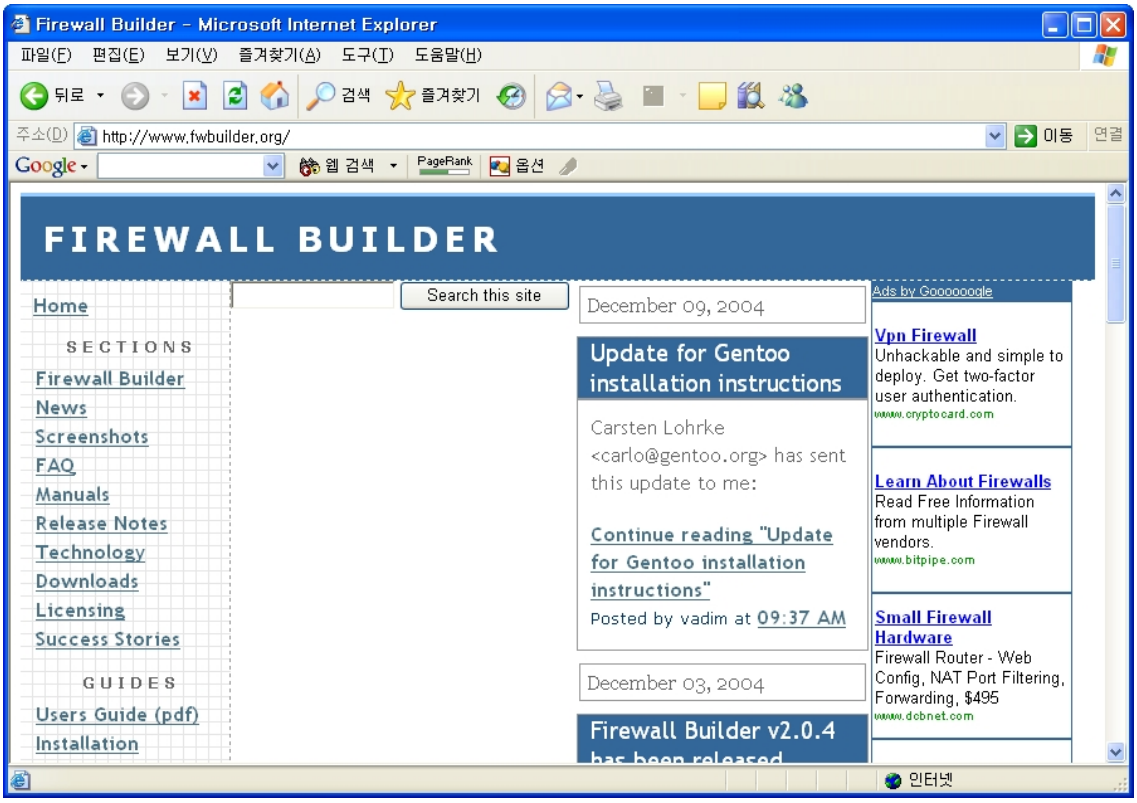
“Current Script" 메뉴를 클릭하여, 위에서 설정한 모든 내용이 포함된 iptables 스크립트를얻을 수 있다. 확인하여 얻은 스크립트를 파일로 만들어서 iptables 스크립트로 사용하도록하자.

이상으로 iptables의 스크립트를 좀 더 손쉽게 만드는 법을 알아보았다. 많은 보안정책을 iptables를 이용하여 생성할 때 유용한 편이므로, 테스트 후 사용해보도록 하자.

Ⅵ. GUI를 활용한 방화벽 스크립트 구축

지금까지는 직접 시스템에 SSH 로그인하여 일일이 룰을 생성하고 수정하는 방법에 대해 알아보았다. 그러나 이 작업이 번거롭고 쉽지 않은 경우에는 GUI 기반의 프로그램을 이용하여 룰을 생성하는 작업을 할 수 있다. 이러한 대표적인 프로그램으로는 firewall builder라는 프로그램이 있는데 리눅스 등 오픈소스에서는 자유롭게 이용이 가능하고 Windows나 MAC OS와 같이 상용 OS 버전은 30일동안 한시적으로 사용이 가능하다.

☞ firewall builder 홈페이지 : http://www.fwbuilder.org



1. fwbuilder의 기능 및 특징

fwbuilder는 다음과 같은 특징을 가지고 있다.

가. 가장 많이 사용되는 100개 이상의 미리 정의된 룰이 있어 기존의 룰을 그대로 가져다 쓰거나 기존의 룰을 참고하여 수정해서 간단히 이용할 수 있다.

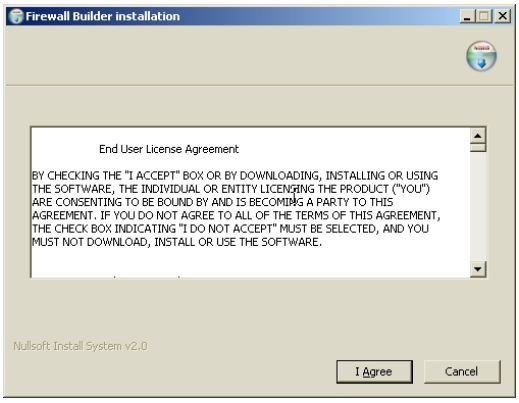
나. 기존의 룰 외에도 서비스명이나 포트 번호 등에 대해 개인적으로 정의하여 사용할 수 있다.

다. 마법사 형식으로 제공되어 초보자도 쉽게 룰 생성 및 수정을 할 수 있다.
라. snmp등을 이용하여 시스템 정보나 인터페이스 정보 등을 자동 인식할 수 있다.
마. 작성된 룰은 스크립트 파일로 저장할 수도 있고 방화벽 시스템에 로그인하여 자동 설치할 수도 있다.
바. 일반적인 windows 환경에서 지원되는 것처럼 오른쪽 마우스 클릭 후 메뉴나 copy & paste 등이 지원된다.
사. iptables 뿐만 아니라 cisco의 PIX, ipfilter, pf 등도 지원한다.

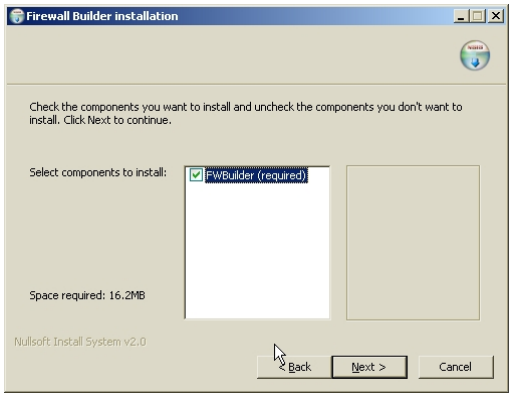
2. fwbuilder의 설치 및 활용

이제 fwbuilder를 설치해 보도록 한다. fwbuilder는 여러 OS에서 제공하는 버전을 제공하는데, 여기에서는 쉽게 사용이 가능한 Windows 기반의 프로그램을 다운로드 하여 설치해 보도록 한다.
먼저 홈페이지에 접속 후 downloads 메뉴를 클릭 후 Windows 버전의 프로그램을 다운로드하면 된다. 이후 다운로드 받은 실행 파일을 실행하여 안내에 따라 설치하면 된다. 설치가 완료된 후 **시작 -> 프로그램 -> Firewall Builder**에서 FWBuilder를 선택하면 된다.

① 인스톨하기 선택
"I Agree" 클릭



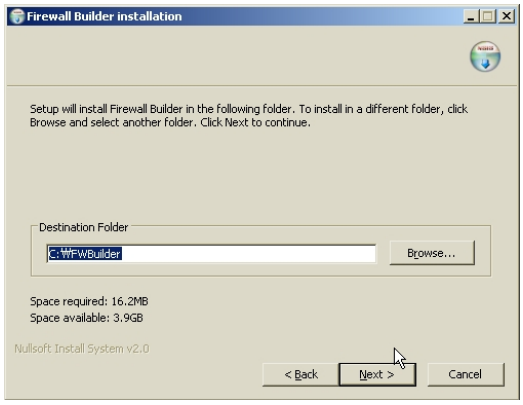
② 구성요소 선택하기
필요한 구성요소를 마우스로 클릭 하여 선택 -> "Next" 클릭



③ 저장하기

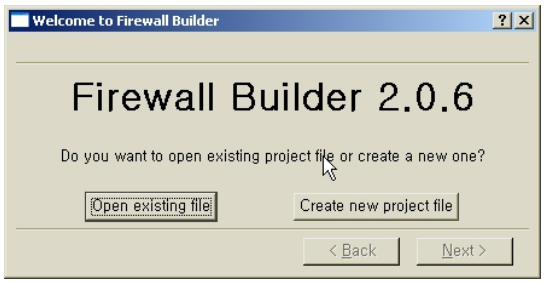
저장할 폴더 선택 (변경하려면 Browse를 선택하여 원하는 위치에 폴더 생성)

-> “Next” 클릭



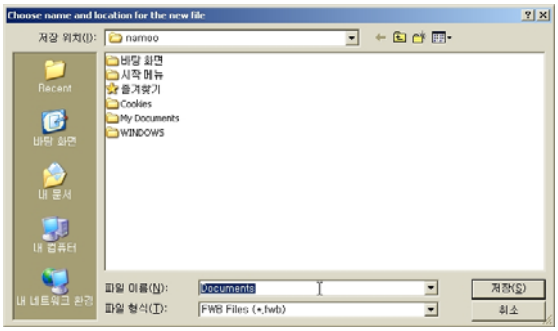
④ 프로젝트 파일 생성

“Open existing file” 클릭-> “Next” 클릭



⑤ 파일명 입력

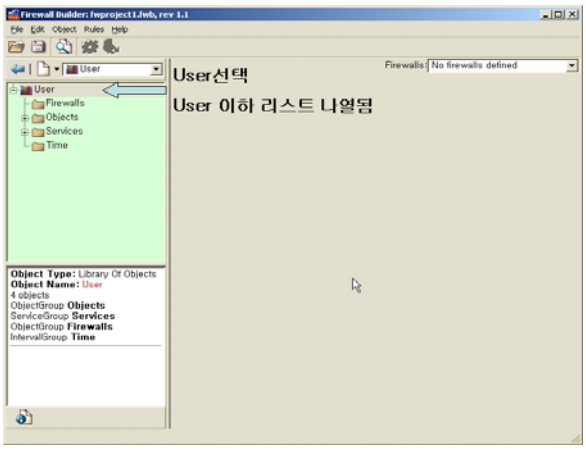
새로운 파일을 생성 후 차후에 생성했던 파일을 선택하여 사용 -> “Next” 클릭



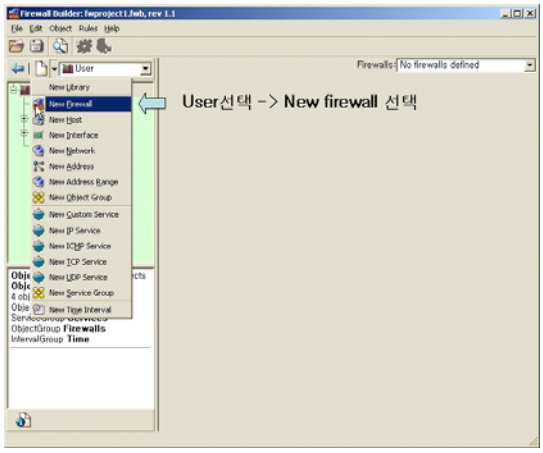
이후 Finish를 선택하면 드디어 프로그램이 로딩된 것이다. 초기에는 아무런 데이터가 없이 초기화된 상태이다.

⑥ Object 생성

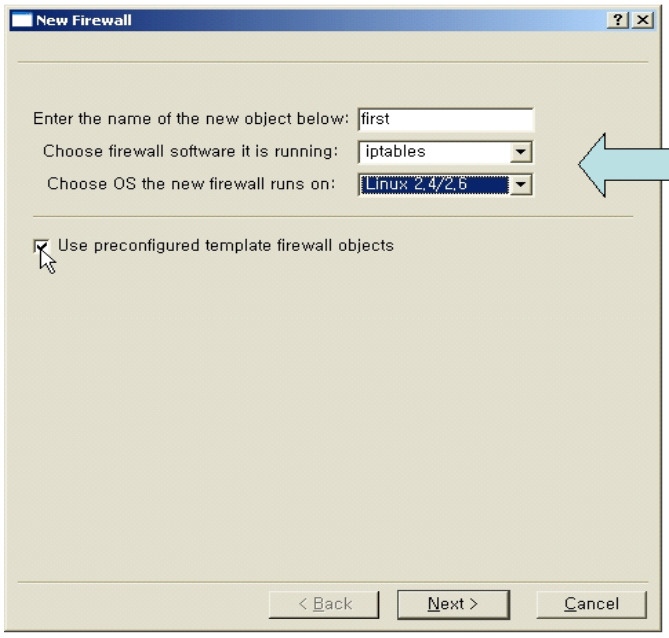
User선택 -> **New firewall** 선택 -> 새로운 Object 이름 지정 -> 방화벽 SW와 OS 지정 -> fwbuilder에서는 자체적으로 사전에 정의된 룰 파일들을 제공하는데 이 룰을 사용하기 위해서 "**Use preconfigured template firewall objects**" 메뉴를 선택 -> Next 선택 -> **Web Server** 선택(방화벽 이외의 타입도 사용이 가능) -> **Finish** 선택



⑥-1 **User**선택



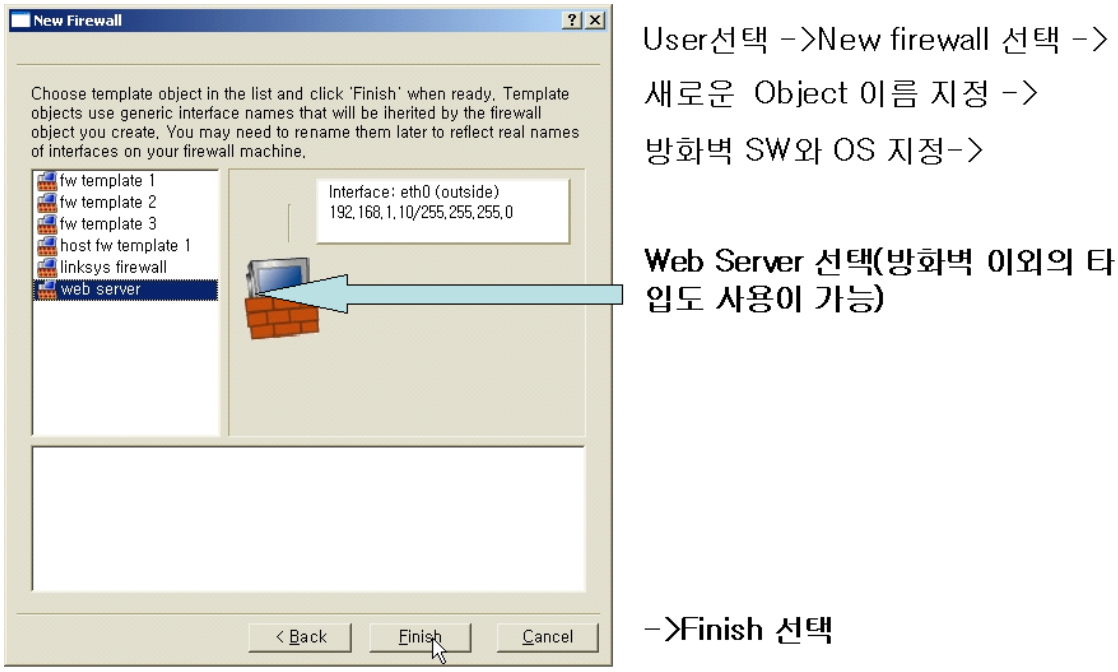
⑥-2 **New firewall** 선택



User선택 -> New firewall 선택 -> **새로운 Object 이름 지정 방화벽 SW와 OS 지정 -> Next 선택**

#fwbuilder에서는 자체적으로 사전에 정의된 룰 파일들을 제공하는데 이 룰을 사용하기 위해서 "Use preconfigured template firewall objects" 메뉴를 선택

⑥-3 새로운 Object 이름 지정 -> 방화벽 SW와 OS 지정



User선택 ->New firewall 선택 ->
새로운 Object 이름 지정 ->
방화벽 SW와 OS 지정->

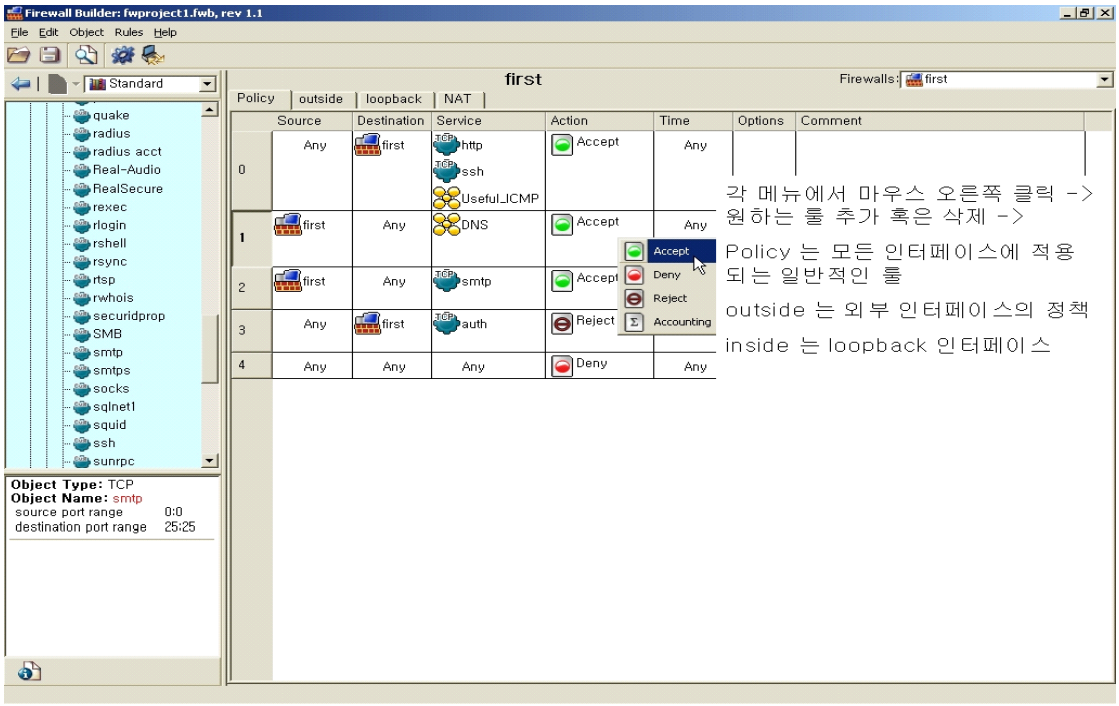
Web Server 선택(방화벽 이외의 타입도 사용이 가능)

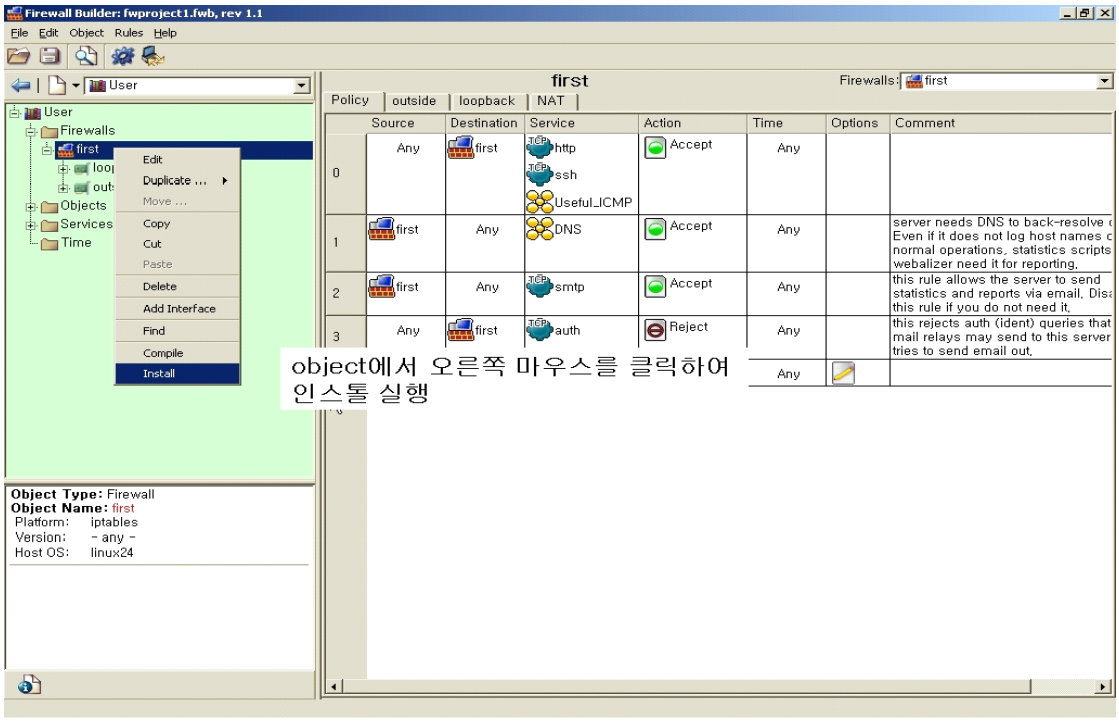
->Finish 선택

⑥-4 **Web Server** 선택(방화벽 이외의 타입도 사용이 가능) -> **Finish** 선택

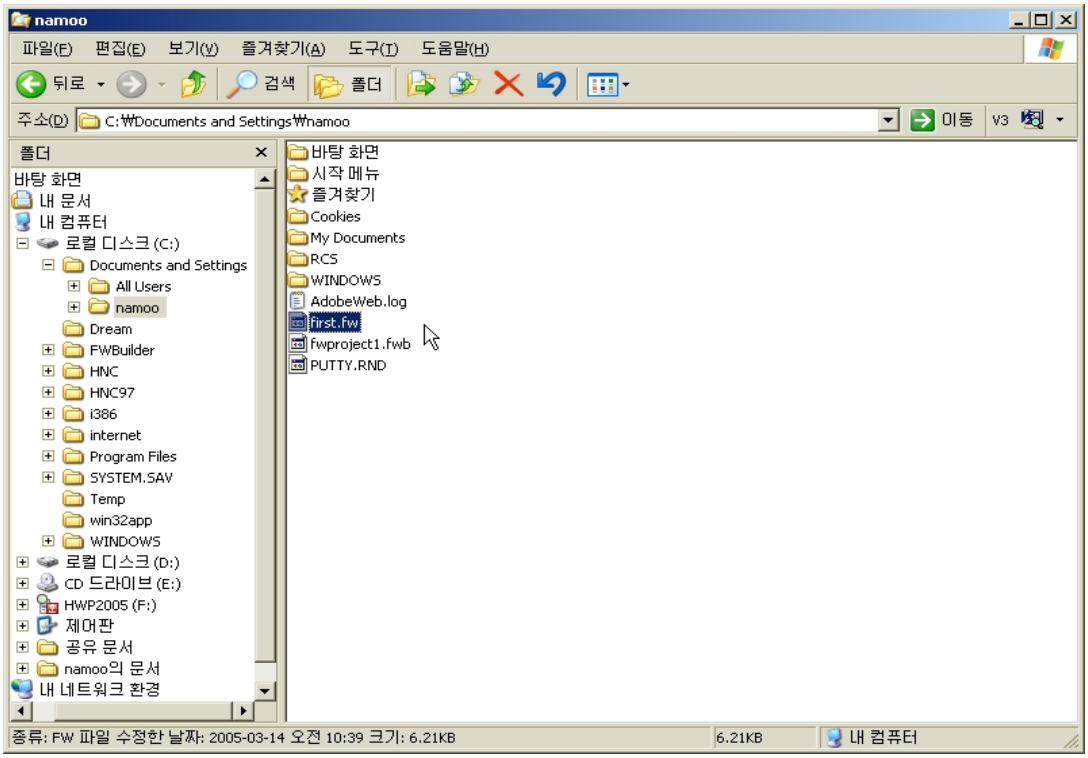
⑦ 정책 설정하기

각 메뉴에서 마우스 오른쪽 클릭 -> 원하는 룰 추가 혹은 삭제 -> Policy 는 모든 인터페이스에 적용되는 일반적인 룰을, outside 는 외부 인터페이스의 정책을, inside 는 loopback 인터페이스 -> object에서 오른쪽 마우스를 클릭하여 인스톨 실행





⑧ 해당 생성 파일 확인



fwbuilder가 제공하는 기능과 옵션은 매우 다양한데, 여기에서는 간략하게만 살펴보았다. 좀 더 많은 옵션과 기능에 대해서는 홈페이지에서 제공하는 매뉴얼을 참고하기 바라며 방화벽이나 iptables에 대한 사전 이해없이 GUI 화면만으로 룰을 생성하려면 당장 생성하는 것은 가능할지 몰라도 문제 발생시 수정하거나 해결하는 데에는 어려움이 있을 수 있으므로 초기에는 직접 설정하면서 테스트하는 것을 추천한다.

<별첨 #1> DNS 트래픽 및 ICMP 트래픽 허용

(1) DNS 트래픽 허용

DNS(Domain Name Service)는 호스트이름을 ip 주소로 또는 ip 주소를 호스트 이름으로 변환해 주는 역할을 하며 기본적으로 udp/tcp 53번을 사용한다. DNS 서비스의 경우 질의를 요청하는 클라이언트 포트가 1024:65535 일수도 있고 53번일 수도 있으니 소스 포트를 명시할 때 주의하도록 하여야 한다. 만약 DNS 서비스를 정확히 허용하지 않을 경우에는 서비스에 직접적인 장애가 될 수 있기 때문이다.

DNS 서비스 운영시 트래픽 허용
방화벽이 운영중인 서버에서 직접 bind 등의 데몬을 실행하여 외부에 DNS 서비스를 제공할 경우에는 모든 곳에서 접근할 수 있도록 -s 0/0 으로 허용하거나 아예 -s 부분을 명시하지 않으면 된다. 그리고 외부에서의 DNS 요청시 소스 포트는 1024:65535 일수도 있고 53 일수도 있으므로 아래 그림과 같이 각각에 대해 명시해 주면된다. 아니면 아예 --sport 부분을 언급하지 않아도 된다. 아울러 DNS 트래픽은 패킷의 특성에 따라 udp 일 수도 있고 tcp 일 수도 있으므로 두 가지를 모두 고려해 주어야 한다.

```
root@firewall:/usr/src
[root@firewall src]#
[root@firewall src]# iptables -A INPUT -p UDP --sport 1024: --dport 53 -m state --state NEW -j ACCEPT
[root@firewall src]# iptables -A INPUT -p TCP --sport 1024: --dport 53 -m state --state NEW -j ACCEPT
[root@firewall src]# iptables -A INPUT -p UDP --sport 53 --dport 53 -m state --state NEW -j ACCEPT
[root@firewall src]# iptables -A INPUT -p TCP --sport 53 --dport 53 -m state --state NEW -j ACCEPT
[root@firewall src]#
```

외부의 DNS를 사용하고자 할 때
대부분 자체적으로 DNS 서버를 운영하기 보다는 ISP에서 제공하는 DNS를 사용하는 경우가 많은데, 이러한 경우 서버 내부에서 외부로 요청하는 것이므로 이 트래픽 역시 먼저 OUTPUT 을 고려한 후 이에 대한 응답인 INPUT을 생각하면 된다. OUTPUT 의 경우 기본적으로 허용되어 있고 이에 대한 응답인 INPUT 의 경우 상태추적인 ESTABLISHED 에 의해 허용되므로 이 트래픽은 별도로 언급하지 않아도 된다.

(2) ICMP 트래픽 허용

ICMP (Internet Control Message Protocol)에 대해 허용 여부를 설정할 차례이다. 언급한 바와 같이 icmp 는 tcp나 udp와는 달리 포트(port)라는 개념이 없이 icmp-type 과 code 가 사용되는데, 허용해 주어야 할 특정한 타입만을 허용해 주면된다. icmp의 타입과 코드에 대해서는 http://www.iana.org/assignment/icmp-parameters를 참고하기 바란다.

① ping(icmp echo request) 허용하기

icmp의 가장 대표적인 ping 은 icmp echo request 에 대하여 icmp echo reply 패킷을 받게 된다. 만약 외부에서 방화벽 서버로의 ping을 허용하려면 INPUT에서 echo request를 허용하면 될 것이다. 그리고 이에 대한 응답인 echo reply 는 OUTPUT을 통해 나가게 되는데. 기본 정책이 허용이므로 별도로 언급하지 않아도 된다.

위 그림은 외부에서의 ping 요청을 허용하기 위한 룰인데, 물론 좀 더 정확하게 하기 위해 -m state --state NEW를 추가할 수 있으나 ICMP는 통상적으로 상태추적을 사용하지 않아도 무방하다. 만약 방화벽에서 외부 호스트 또는 네트워크로의 ping을 허용하기 위해서는 먼저 OUPUT을 생각하면 나가는 트래픽 중 icmp type이 echo-request인 패킷을 허용하면 되는데, OUTPUT은 기본적으로 허용이므로 언급하지 않아도 된다. 그리고 이에 대한 응답인 echo-reply를 INPUT에서 허용하면 되지만, 이는 상태추적의 관점에서 ESTABLISHED이므로 별도로 언급하지 않아도 된다.

② source-quench 허용하기

source-quench는 자주 사용되지는 않지만 수신자의 버퍼가 꽉 차서 더 이상 자료를 받을 수 없을 때 자료를 보내는 소스 호스트에게 보내는 메시지인데, 필요하므로 허용하는 것이 좋다. OUTPUT 은 기본적으로 허용이고, 상태추적에 의해 요청에 대한 응답도 허용된다.

③ destination-unreachable 허용하기

destination-unreachable 은 traceroute를 허용하고자 할 때 사용되는데, OUTPUT 은 기본적으로 허용이므로 INPUT에서만 언급하면 된다.

④ fragmentation-needed 허용하기

A와 B 이렇게 양 구간에서의 통신시 중간을 경유하는 라우터의 MTU 값보다 큰 사이즈의 패킷이 전송될 경우에는 패킷이 **단편화**된다. 그러나 만약 단편화될 수 없는 경우에는 패킷을 drop 하고 해당 패킷의 소스 ip에게 단편화가 필요하다는 icmp 패킷을 발송하게 된다. 따라서 이 패킷을 허용하여야 중간에 패킷이 drop 되었다는 것을 알기 때문에 허용설정을 하여야 한다. OUTPUT은 기본적으로 허용이므로 INPUT 에만 설정하면 된다.

⑤ time-exceeded 허용하기

내부에서 외부의 네트워크로 traceroute를 할 때 TTL이 초과되었다는 icmp 메시지를 받고자 할 때 필요하다. OUTPUT 은 기본적으로 허용이므로 INPUT 에서만 허용하면 된다.

⑥ parameter-problem 허용하기

마지막으로 parameter-problem 은 수신한 패킷 헤더에 비정상적이거나 예상치 못한 데이터가 있을 경우 송신자에게 전송하는 icmp 에러 메시지로서 이 패킷도 허용하여야 한다. OUTPUT 은 기본적으로 허용이므로 INPUT 에서만 허용하도록 한다.

<별첨 #2> iptables 설정 스크립트 예제

```
#!/bin/sh
# 자체 서버 방화벽 룰 설정 파일

iptables -F -t filter
iptables -F -t nat
iptables -F -t mangle
# 기존에 설정되어 있을지 모를 룰을 모두 초기화한다.

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
# 모든 룰에 매칭되지 않았을 때 실행될 기본 정책을 설정하는 부분이다.
# 자체 서버형태에서는 패킷을 다른 서버로 포워딩하지 않으므로 FORWARD chains 은
# 사용할 필요가 없다. 따라서 기본 정책으로 DROP 하고, OUTPUT은 가급적 허용하도록
# 한다. 단, INPUT에서는 반드시 DROP 하도록 한다.

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# 루프백 인터페이스를 통해 오가는 트래픽은 허용한다.



iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 255.255.255.255/32 -j DROP
iptables -A INPUT -i eth0 -s 0.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 169.254.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A INPUT -i eth0 -s 192.0.2.0/24 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 224.0.0.0/4 -j DROP
iptables -A INPUT -i eth0 -s 240.0.0.0/5 -j DROP
iptables -A INPUT -i eth0 -s 248.0.0.0/5 -j DROP
# INPUT 패킷중 RFC 1918 에 정의된 사설 ip 및 일반적인 인터넷에서는 사용될 수
# 없는 ip 또는 ip 대역을 소스로 한 패킷을 차단한다.
# 위의 경우 외부 네트워크의 인터페이스로 eth0을 사용하는 경우이다.

iptables -A OUTPUT -d 10.0.0.0/8 -j DROP
iptables -A OUTPUT -d 255.255.255.255/32 -j DROP
iptables -A OUTPUT -d 0.0.0.0/8 -j DROP
```

```
iptables -A OUTPUT -d 169.254.0.0/16 -j DROP
iptables -A OUTPUT -d 172.16.0.0/12 -j DROP
iptables -A OUTPUT -d 192.0.2.0/24 -j DROP
iptables -A OUTPUT -d 192.168.0.0/16 -j DROP
iptables -A OUTPUT -d 224.0.0.0/4 -j DROP
iptables -A OUTPUT -d 240.0.0.0/5 -j DROP
iptables -A OUTPUT -d 248.0.0.0/5 -j DROP
# OUTPUT chain 에 대한 설정으로 비정상적인 ip 또는 ip 대역을 목적지로 한 패킷을
# 거부한다. FORWARD 는 어떠한 트래픽도 허용하지 않으므로 별도로 언급하지 않아도
# 된다.

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# 상태 추적에 따라 허용되어 이미 세션을 맺은 후 뒤따라오는 패킷은 허용하도록 한다.

iptables -A INPUT -i eth0 -p TCP ! --syn -m state --state NEW -j DROP
# tcp 패킷중 상태추적에는 NEW 이면서 syn 패킷이 아닌 패킷을 필터링한다.
# 이외의 패킷은 모두 위조된 패킷이기 때문이다.

iptables -A INPUT -i eth0 -p ALL  -m state --state INVALID -j DROP
# 상태추적 테이블에서 INVALID 인 패킷은 차단한다.



iptables -A INPUT -i eth0 -p TCP --sport 1024: --dport 80 -m state --state NEW -j
ACCEPT
# 웹 서비스를 제공할 때 목적지 포트가 80번으로 향하는 초기(NEW) 패킷은 허용한다.



iptables -A INPUT -i eth0 -p TCP -s 192.168.1.0/24 --sport 1024: --dport 21 -m state
--state NEW -j ACCEPT
# 192.168.1.0/24 대역에서의 ftp 접속을 허용한다.

iptables -A INPUT -i eth0 -p TCP -s 192.168.1.3 --sport 1024: --dport 22 -m state --state
NEW -j ACCEPT
# 192.168.1.3 에서의 ssh 접속을 허용한다.

iptables -A INPUT -i eth0 -p TCP --sport 1024: --dport 25 -m state --state NEW -j
ACCEPT
# SMTP 서비스를 제공할 때 외부에서 오는 메일을 받아 서버에 저장하거나 다른 서버로
# 보내고자 할 때 필요하므로 25/tcp를 허용한다.
```

```
iptables -A INPUT -i eth0 -p TCP -s 192.168.1.0/24 --sport 1024: --dport 110 -m state --state NEW -j ACCEPT
# pop3 서비스를 제공할 때 110/tcp 번으로 향하는 초기(NEW)  패킷을 허용한다.


iptables -A INPUT -p ICMP --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type network-unreachable -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type host-unreachable -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type port-unreachable -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type fragmentation-needed -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type time-exceeded -j ACCEPT
# icmp 와 관련된 패킷을 허용한다.
# 허용하고자 하는 icmp type을 지정하면 된다.
```

<별첨 #3> NAT를 이용한 방화벽 정책 설정 예제

아래는 NAT 방화벽에 설정하여 사용할 수 있는 룰 예제이다. 룰을 볼 때는 먼저 각 룰의 개개 의미를 먼저 살펴보고, 전체적인 흐름에서 다시 살펴보기 바란다. 이 예에서는 eth0에 공인 ip로 221.1.2.3이 할당되어 있고, 내부의 사설 대역에는 192.168.1.0/24 대역을 사용하며 내부의 게이트웨이는 192.168.1.1 과 같은 eth1의 ip 주소를 사용한다고 가정한다. NAT 방화벽 자체의 게이트웨이는 공인 ip의 게이트웨이로 설정되어 있어야 한다.

```
#!/bin/sh

SERVICE_IP="221.1.2.3"
# 방화벽에서 사용할 공인 ip를 지정한다. 각자의 공인 ip를 설정하면 된다.

/bin/echo "1" >/proc/sys/net/ipv4/ip_forward
# NAT를 사용하려면 ip_forward 이 반드시 1 이어야 한다.

iptables -t nat -F
iptables-t mangle -F
iptables -t filter -F
# 각 테이블에서 기존의 룰을 모두 초기화(flush)한다.

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
# filter 테이블의 기본정책을 설정한다. 여기에서 INPUT 과 FORWARD 는 DROP 하고
# OUTPUT 은 ACCEPT 로 하였다.

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# 루프백 인터페이스를 통한 트래픽은 허용한다.

iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 255.255.255.255/32 -j DROP
iptables -A INPUT -i eth0 -s 0.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 169.254.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A INPUT -i eth0 -s 192.0.2.0/24 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 224.0.0.0/4 -j DROP
iptables -A INPUT -i eth0 -s 240.0.0.0/5 -j DROP
```

```
iptables -A INPUT -i eth0 -s 248.0.0.0/5 -j DROP
# 외부에서 NAT 방화벽 자체를 향하는 트래픽에 대한 제어를 한다. 이는 INPUT chain
# 에서 필터링 설정하면 된다. 사설ip 및 공인 네트워크에서 사용되지 않는 ip 대역을
# 소스로 한 패킷을 차단한다.

iptables -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 255.255.255.255/32 -j DROP
iptables -A FORWARD -i eth0 -s 0.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 169.254.0.0/16 -j DROP
iptables -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A FORWARD -i eth0 -s 192.0.2.0/24 -j DROP
iptables -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A FORWARD -i eth0 -s 224.0.0.0/4 -j DROP
iptables -A FORWARD -i eth0 -s 240.0.0.0/5 -j DROP
iptables -A FORWARD -i eth0 -s 248.0.0.0/5 -j DROP
# 외부에서 NAT 방화벽을 통과하여 내부의 서버를 향하는 트래픽에 대한 제어를 한다.
# 이는 FORWARD chain 에서 필터링 설정하면 된다.

iptables -A OUTPUT -d 10.0.0.0/8 -j DROP
iptables -A OUTPUT -d 255.255.255.255/32 -j DROP
iptables -A OUTPUT -d 0.0.0.0/8 -j DROP
iptables -A OUTPUT -d 169.254.0.0/16 -j DROP
iptables -A OUTPUT -d 172.16.0.0/12 -j DROP
iptables -A OUTPUT -d 192.0.2.0/24 -j DROP
iptables -A OUTPUT -d 192.168.0.0/16 -j DROP
iptables -A OUTPUT -d 224.0.0.0/4 -j DROP
iptables -A OUTPUT -d 240.0.0.0/5 -j DROP
iptables -A OUTPUT -d 248.0.0.0/5 -j DROP
# 방화벽에서 외부로 나가는 트래픽에 대한 제어를 한다.

iptables -A FORWARD -i eth1 -s 192.168.1.0/24 -j ACCEPT
# 내부의 사설 대역에서 사설 인터페이스인 eth1을 통해 들어오는 패킷은
# 내부 네트워크에서의 정상적인 트래픽이므로 허용한다.

iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -j SNAT --to $SERVICE_IP
# 내부에서 192.168.1.0/24 대역을 사용할 경우 eth0을 통과해 나갈 때 SERVICE_IP
# 에서 지정한 공인ip 로 변환되어 나가도록 한다.

iptables -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD  -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
# 이미 세션을 맺어 상태추적 테이블 목록에 있는 ESTABLISHED,RELATED 패킷은 허용
# 한다.

iptables -t nat -A PREROUTING -d $SERVICE_IP -p TCP --dport 21 -j DNAT --to
192.168.1.1:21
# 앞에서 지정한 SERVICE_IP 의 21 번 포트로 향하는 트래픽은 192.168.1.1 의 21번으
# 로 포워딩한다. 따라서 외부에서 221.1.2.3 의 21번으로 접속하면 192.168.1.1 의 21번 #
포트가 응답하게 된다.

iptables -t nat -A PREROUTING -d $SERVICE_IP -p TCP --dport 747 -j DNAT --to
192.168.1.12:80
# SERVICE_IP 의 747 번으로 향하는 트래픽은 192.168.1.12 의 80 번으로 포워딩한다.

iptables -A INPUT -p TCP ! --syn -m state --state NEW -j DROP
iptables -A FORWARD -p TCP ! --syn -m state --state NEW -j DROP
# 상태추적 테이블에 NEW이면서 syn 비트가 설정되지 않은 tcp 패킷은 차단한다.

iptables -A INPUT -p ALL -m state --state INVALID -j DROP
iptables -A FORWARD -p ALL -m state --state INVALID -j DROP
iptables -A OUTPUT -p ALL -m state --state INVALID -j DROP
# 상태추적 테이블에서 INVALID 인 패킷은 차단한다.

iptables -A FORWARD -p TCP --sport 1024: --dport 747 -m state --state NEW -j
ACCEPT
# 공인 ip의 747/tcp 포트를 향하는 트래픽 즉, 192.168.1.12 의 80번으로 향하는 트래픽
# 을 허용해 주는 설정이다.

iptables -A FORWARD  -p TCP  --sport 1024: --dport 21 -m state --state NEW -j
ACCEPT
# 앞에서 NAT 설정한 21/tcp 에 대해 포트 포워딩을 허용한다.

iptables -A INPUT  -p TCP -s 192.168.1.3 --sport 1024: --dport 22 -m state --state NEW
-j ACCEPT
# 관리를 위해 192.168.1.3 에서 방화벽으로의 ssh 로그인을 허용한다.

iptables -N ICMP_HANDLE
iptables -F ICMP_HANDLE
# icmp 트래픽에 대한 설정으로 INPUT, FORWARD 에 대해 각각 동일한 룰을
# 반복적으로 설정하여야 하는데, 이를 간소화하기 위해 ICMP_HANDLE 이라는 별도의
```

```
# chain을 생성하도록 한다.

iptables -A ICMP_HANDLE -p ICMP --icmp-type echo-reply -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type network-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type host-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type port-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type fragmentation-needed -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type time-exceeded -j ACCEPT
# 허용해 주어야 할 몇 가지 ICMP type을 설정해 준다.

iptables -A INPUT -p ICMP -j ICMP_HANDLE
iptables -A FORWARD -p ICMP -j ICMP_HANDLE
# INPUT, FORWARD 되는 패킷 중 ICMP 패킷을 ICMP_HANDLE chain 에 보낸다.
```

<별첨 #4> 브리지 방화벽 룰 설정 예제

실제 iptables를 이용하여 브리지 방화벽 룰을 설정할 차례이다. 브리지 방식이라고 해서 특별할 것이 없으며, NAT 와 매우 유사하거나 오히려 더 쉽고 간단하다. 브리지는 NAT와 관련된 룰을 고려할 필요가 없고, 단지 FORWARD chain 만으로 허용 또는 거부만 설정하면 된다. 브리지 방화벽 룰은 앞에서 살펴보았던 NAT 와 관련된 룰을 아래와 같이 수정하여 사용한다.

```
#!/bin/sh

SERVICE_IP="221.1.2.3"
#- 방화벽에서 사용할 공인 ip를 지정한다. 각자의 공인 ip를 설정하면 된다.

/bin/echo "1" >/proc/sys/net/ipv4/ip_forward

iptables -t nat -F
iptables-t mangle -F
iptables -t filter -F

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 255.255.255.255/32 -j DROP
iptables -A INPUT -i eth0 -s 0.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 169.254.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A INPUT -i eth0 -s 192.0.2.0/24 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A INPUT -i eth0 -s 224.0.0.0/4 -j DROP
iptables -A INPUT -i eth0 -s 240.0.0.0/5 -j DROP
iptables -A INPUT -i eth0 -s 248.0.0.0/5 -j DROP

iptables -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 255.255.255.255/32 -j DROP
iptables -A FORWARD -i eth0 -s 0.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 169.254.0.0/16 -j DROP
```

```
iptables -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A FORWARD -i eth0 -s 192.0.2.0/24 -j DROP
iptables -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A FORWARD -i eth0 -s 224.0.0.0/4 -j DROP
iptables -A FORWARD -i eth0 -s 240.0.0.0/5 -j DROP
iptables -A FORWARD -i eth0 -s 248.0.0.0/5 -j DROP

iptables -A OUTPUT -d 10.0.0.0/8 -j DROP
iptables -A OUTPUT -d 255.255.255.255/32 -j DROP
iptables -A OUTPUT -d 0.0.0.0/8 -j DROP
iptables -A OUTPUT -d 169.254.0.0/16 -j DROP
iptables -A OUTPUT -d 172.16.0.0/12 -j DROP
iptables -A OUTPUT -d 192.0.2.0/24 -j DROP
iptables -A OUTPUT -d 192.168.0.0/16 -j DROP
iptables -A OUTPUT -d 224.0.0.0/4 -j DROP
iptables -A OUTPUT -d 240.0.0.0/5 -j DROP
iptables -A OUTPUT -d 248.0.0.0/5 -j DROP

iptables -A FORWARD -i eth1 -s 192.168.1.0/24 -j ACCEPT

iptables -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p TCP ! --syn -m state --state NEW -j DROP
iptables -A FORWARD -p TCP ! --syn -m state --state NEW -j DROP

iptables -A INPUT -p ALL -m state --state INVALID -j DROP
iptables -A FORWARD -p ALL -m state --state INVALID -j DROP
iptables -A OUTPUT -p ALL -m state --state INVALID -j DROP

iptables -A FORWARD -p TCP --sport 1024: --dport 747 -m state --state NEW -j
ACCEPT

iptables -A FORWARD  -p TCP  --sport 1024: --dport 21 -m state --state NEW -j
ACCEPT

iptables -A INPUT  -p TCP -s $SERVICE_IP --sport 1024: --dport 22 -m state --state
NEW -j ACCEPT
```

```
iptables -N ICMP_HANDLE
iptables -F ICMP_HANDLE

iptables -A ICMP_HANDLE -p ICMP --icmp-type echo-reply -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type network-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type host-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type port-unreachable -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type fragmentation-needed -j ACCEPT
iptables -A ICMP_HANDLE -p ICMP --icmp-type time-exceeded -j ACCEPT

iptables -A INPUT -p ICMP -j ICMP_HANDLE
iptables -A FORWARD -p ICMP -j ICMP_HANDLE
```