Technical Guide

iptables을 활용한 가상 서버 네트워크 제어

Netfilter

May 2021

SAMSUNG SDS

Contents

1. 개요

이 문서에서는 iptables 동작 원리의 바탕이 되는 Netfilter 라는 프레임워크와 기본적인 iptables 사용법에 대해 다뤄보겠습니다.

iptables 를 이용해서 리눅스의 방화벽 기능이 동작되게 하는 것은 실제로는 iptables 가 아니라, Netfilter 라는 리눅스 커널에 이미 내장되어 있는 프레임워크 덕분입니다. Netfilter 를 주로 패킷 필터링 프레임워크(Packet Filtering Framework)라고 부르는데, 말 그대로 패킷을 필터링할 수 있는 뼈대를 제공한다고 이해하면 됩니다.

즉, Netfilter 자체가 패킷 필터링을 구현해서 모든 규칙을 다 가지고 있는 것은 아니고, iptables 라는 User-Space 에서 실행되는 툴을 제공하고, 이 툴을 사용해 룰을 추가하면, 패킷이 리눅스 커널에 들어와서 Netfilter 가 정해놓은 각 지점을 통과할 때 미리 정해진 룰들이 있으면, Netfilter 가 패킷이 그 룰들을 거치게 만들어서 drop 또는 accept 등으로 동작하게 합니다.

2. Netfilter가 제공하는 Hook point

Netfilter 는 5 개의 Hook point 를 제공합니다. 패킷들이 각 지점을 통과하면서 각 Hook point 에 등록된 커널 모듈을 트리거하게 됩니다.

동일 Hook point 에 복수 개의 커널 모듈을 등록해서 사용할 수 있으므로, 커널 모듈이 Hook point 를 잡아서 쓰겠다고 등록할 때 Priority Number 를 입력해서 그 우선 순위대로 커널 모듈을 트리거하게 됩니다.

1. NF_IP_PRE_ROUTING: 외부에서 온 패킷이 리눅스 커널의 네트워크 스택에 들어온 직후에 발생하는 Hook입니다. 패킷을 라우팅하기 전에 발생합니다.
2. NF_IP_LOCAL_IN: 패킷이 라우팅된 후 목적지가 자신일 경우, 패킷을 로컬 프로세스에 전달하기 전에 발생하는 Hook입니다.
3. NF_IP_FORWARD: 패킷이 라우팅된 후 목적지가 자신이 아닐 경우, 패킷을 다른 곳으로 Forwarding하는 경우 발생하는 Hook입니다.
4. NF_IP_LOCAL_OUT: 패킷이 프로세스에서 나와 네트워크 스택을 통과하기 전에 발생하는 Hook입니다.
5. NF_IP_POST_ROUTING: 패킷이 네트워크 스택을 통과한 후 밖으로 보내기 전 발생하는 Hook입니다.

위 5 개의 Hook point 를 그림으로 나타내면 다음과 같습니다.

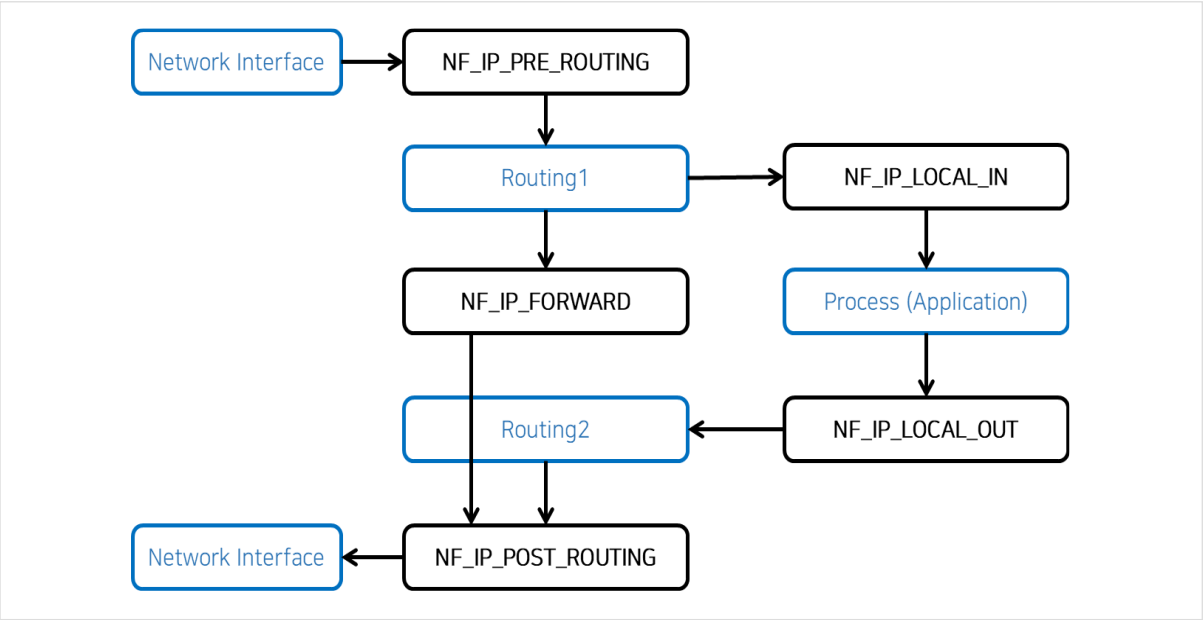

Figure 1. Netfilter 패킷 경로

위 그림은 Netfilter 에서 패킷의 경로를 나타내고 있습니다.

Routing1 은 NIC(Network Interface Card)에서 전달받은 패킷이 로컬 프로세스가 받아야 하는 패킷인지, 아니면 다른 호스트가 받아야 하는 패킷인지 구분하여 라우팅하는 과정을 나타냅니다.
Routing2 는 프로세스에서 전송한 패킷이 NF_IP_LOCAL_OUT 에서 DNAT 될 경우 다시 라우팅하는 과정을 나타냅니다.

정리하면, 패킷 경로는 아래의 3 가지로 구분할 수 있습니다.

1. 외부에서 온 패킷의 목적지가 자신인 경우: NF_IP_PRE_ROUTING -> NF_IP_LOCAL_IN -> Process
2. 외부에서 온 패킷의 목적지가 자신이 아닌 경우: NF_IP_PRE_ROUTING -> NF_IP_FORWARD -> NF_IP_POST_ROUTING -> Network Interface
3. 로컬 프로세스에서 전송하는 패킷의 경우: NF_IP_LOCAL_OUT -> NF_IP_POST_ROUTING -> Network Interface

3. iptables 동작 원리

iptables 에 의한 패킷 필터링 메커니즘을 이해하기 위해서는 아래 네 가지 용어 또는 구조를 이해해야 합니다.

- tables, chains, rules, targets

테이블(tables)은 이미 커널에 만들어져 있고 5 개가 있습니다. 각 테이블에는 미리 정의된 체인(chain)들이 있고, 이 체인에 룰(rule)들이 들어가게 됩니다. 룰에는 타겟(target)을 정하게 되어 있는데, 어떤 패킷이 해당 룰에 일치(match)하면 그 타겟으로 가서 다시 다른 룰에 일치하는지 검사하거나, drop 이나 accept 등의 동작을 할 수 있습니다.

그럼 각 구조에 대해서 좀 더 알아보겠습니다.

4. 테이블

다음과 같이 총 5 가지의 테이블을 제공합니다.

1. Filter Table: 패킷 필터링을 위한 테이블입니다. 패킷을 원래 목적지까지 전달할지, 아니면 drop할지를 결정합니다. Firewall 기능은 Filter Table을 통해 구축할 수 있습니다.
2. Mangle Table: 패킷의 IP 헤더를 바꾸는데 사용됩니다. 예를 들어, 패킷의 TTL(Time to Live)을 변경하거나 패킷을 Marking하여 다른 iptables의 테이블이나 네트워크 툴에서 패킷을 구분할 수 있도록 할 수 있습니다.
3. NAT Table: 패킷 NAT(Network Address Translation)를 위한 테이블입니다. 패킷의 Source Address나 Destination Address를 변경합니다.
4. Raw Table: Netfilter 프레임워크는 Hook 뿐만 아니라 Connection Tracking 기능을 제공합니다. 이전에 도착한 패킷들을 바탕으로 방금 도착한 패킷의 Connection을 추적합니다. Raw Table은 특정 패킷이 Connection Tracking에서 제외되도록 설정합니다.
5. Security Table: SELinux에서 패킷을 어떻게 처리할지 결정하기 위한 테이블입니다.

5. 체인

각 테이블에는 빌트인(built-in) 체인들이 있고, 이 체인들의 이름은 Netfilter Hook 이름을 따와서 만들었습니다.

1. PREROUTING: NF_IP_PRE_ROUTING Hook이 트리거합니다.
2. INPUT: NF_IP_LOCAL_IN Hook이 트리거합니다.
3. FORWARD: NF_IP_FORWARD Hook이 트리거합니다.
4. OUTPUT: NF_IP_LOCAL_OUT Hook이 트리거합니다.
5. POSTROUTING: NF_IP_POST_ROUTING Hook이 트리거합니다.

6. 패킷 흐름도

iptables 에서 패킷의 경로는 다음과 같습니다.

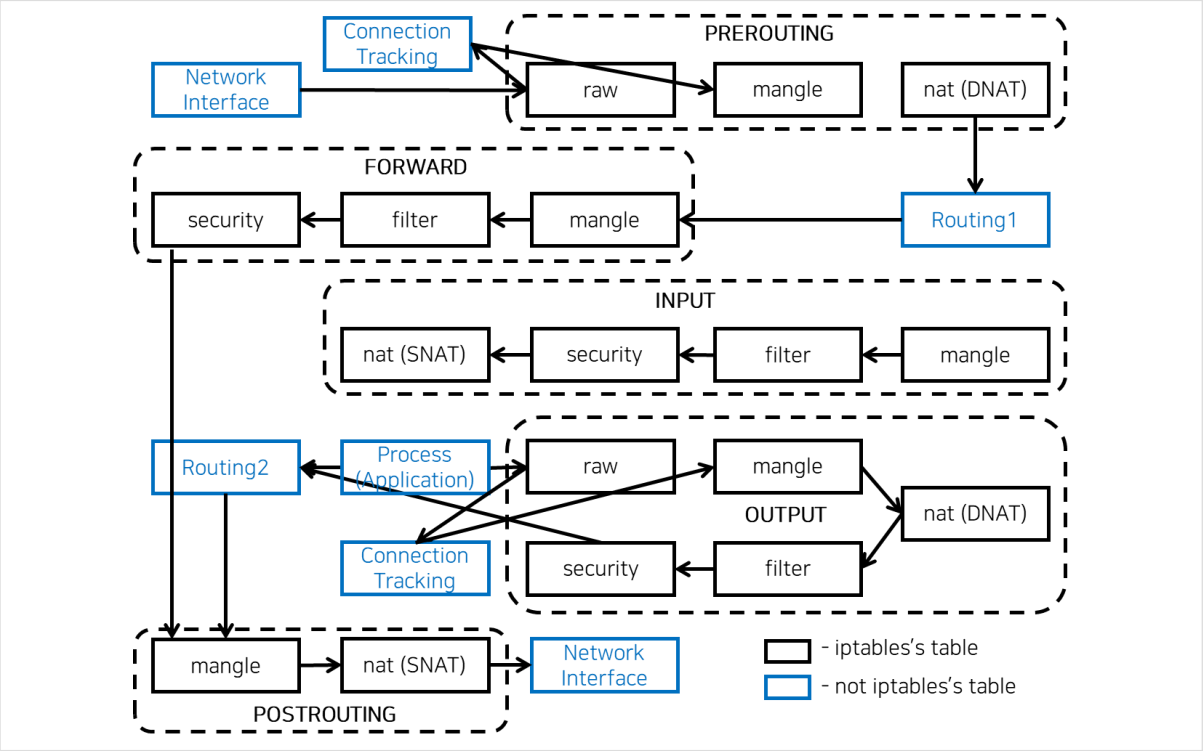

Figure 2. iptables 패킷 경로

위 그림처럼 패킷들은 각 Hook 에서 여러 테이블들을 통과하면서 처리됩니다. 물론 각 테이블에는 여러 체인들이 있고, 각 체인에 룰이 등록되는데, 패킷이 어떤 룰에 매칭되어서 그 운명이 결정된다면(예를 들어, accept 또는 drop), 다음 체인 또는 테이블로 진행하지 않습니다.

7. iptables 룰

룰은 특정 테이블의 특정 체인에 등록되어 있고, 각 체인들이 호출될 때 패킷이 등록된 룰과의 매치 여부를 확인한 후 액션을 결정합니다.

룰의 매치 기준으로는 프로토콜 타입, 출발지 주소/포트, 목적지 주소/포트, input/output interface 등의 조합이 될 수 있습니다.

8. 타겟

타겟은 패킷이 등록한 룰과 매치되었을 때 트리거되는 액션입니다.

타겟은 -j 또는 --jump 옵션을 사용합니다. 타겟은 사용자 정의 체인(User-defined Chain)이거나, 빌트인 타겟 중의 하나이거나, 타겟 확장(extension)일 수 있습니다.

사용자 정의 체인인 경우에는 그 지정된 사용자 정의 체인으로 점프해서 룰 매칭 여부를 계속 시도합니다.

빌트인 타겟은 ACCEPT, DROP, QUEUE, RETURN 이 있는데 다음과 같습니다.

1. ACCEPT - iptables가 패킷을 받아들입니다.
2. DROP - iptables가 패킷을 버립니다. 시스템에 연결하려는 사람에게는 시스템이 존재하지 않는 것처럼 보입니다.
3. QUEUE - iptables는 패킷을 userspace로 전달합니다.
4. RETURN - iptables는 이 패킷에 대한 현재 체인의 다음 규칙 세트 실행을 중지합니다. 컨트롤은 호출 체인으로 반환됩니다.

타겟 확장으로는 REJECT, LOG 가 있습니다.

1. REJECT: iptables는 패킷을 "거부"합니다. TCP의 경우 "connection reset" 패킷을, UDP 또는 ICMP의 경우 "destination host unreachable" 패킷을 보냅니다. (위 빌트인 타겟인 DROP 과 차이가 있음)

만약, 타겟이 빌트인 타겟이면 패킷의 운명이 바로 결정되기 때문에 현재 테이블에서의 패킷 처리는 더 이상 진행하지 않습니다.

만약, 타겟이 사용자 정의 체인이고 패킷의 운명이 이 사용자 정의 체인에 의해 결정되지 않는다면, 다시 원래 체인으로 돌아와서 나머지 룰과의 매칭을 시도하게 됩니다.

9. Policy Chain Default Behavior

위에서 언급한 각 체인들에 룰들이 추가된다고 했을 때, 만약 패킷이 어떤 룰에도 매칭되지 않으면 어떻게 처리할 것인지를 정해야 합니다.

Options	Description
-N (--new-chain)	새로운 체인 추가
-F(--flush)	해당 체인의 모든 rule 삭제
-P (--policy)	정책 변경

-R (--replace)	Rule 을 교체
-A (--append)	새로운 rule 추가
-I (--insert)	새로운 rule 삽입
-X(--delete-chain)	빈 체인 삭제
-L (--list)	테이블에 정의된 rule 확인
-D (--delete)	Rule 삭제

현재 정책을 보려면 아래와 같이 수행합니다.

```
$ sudo iptables -L | grep policy
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

iptables 명령어에서 -t 옵션으로 테이블을 지정하지 않으면 디폴트 값이 filter table 이므로 다른 테이블(예: nat)의 체인들에 대한 디폴트 정책을 알고 싶으면 아래와 같이 수행합니다.

```
$ sudo iptables  -t nat  -L | grep policy
```

만약, 테이블의 각 체인들에 대한 디폴트 정책을 변경하고 싶으면 아래와 같이 -P 옵션을 사용합니다.

다음은 디폴트 값으로 모든 패킷에 대해 accept 일 경우입니다.

```
$ sudo iptables -P INPUT ACCEPT
$ sudo iptables -P OUTPUT ACCEPT
$ sudo iptables -P FORWARD ACCEPT
```

다음은 디폴트 값으로 모든 패킷에 대해 drop 일 경우입니다.

```
$ sudo iptables -P INPUT DROP
$ sudo iptables -P OUTPUT DROP
$ sudo iptables -P FORWARD DROP
```

현재 룰을 확인하기 위해서는 다음의 명령을 수행합니다.

```
$ sudo iptables –L
$ sudo iptables -S
```

체인의 끝에 룰을 추가하기(-A (--append)) 위해서는 다음의 명령을 수행합니다.

```
$ sudo iptables -A
```

체인의 특정 위치에 룰을 삽입(-I(--insert)) 하려면 다음의 명령을 수행합니다.

```
$ sudo iptables -I
```

룰을 교체(-R(--replace))하려면 다음의 명령을 수행합니다.

```
$ sudo iptables -R
```

룰을 삭제(-D(--delete))하려면 다음의 명령을 수행합니다.

```
$ sudo iptables -D
```

10. iptables 사용 예제

자주 사용되는 옵션에 대해 먼저 살펴보면 아래와 같습니다.

Options	Description
-s (--source)	출발지에 대한 IP설정
-d (--destination)	목적지에 대한 IP 설정
-j (--jump)	ACCEPT, DROP 등 Target 으로 JUMP
-p (--protocol)	TCP, UDP 등 Protocol 을 설정
-t (--table)	위에 언급한 다섯 가지 테이블을 지정
--sport	출발지 port
--dport	도착지 port
--m(--match)	특정 모듈을 사용할 경우
--state	연결상태 지정(FIN_WAIT 등)

--string	Application의 string과 매칭
-!	특정 IP 를 제외할 경우
-i (--in-interface)	들어오는(in) 인터페이스 지정
-o (--out-interface)	나가는(out) 인터페이스 지정

몇 가지 상황에 대한 iptables 사용 예제를 보여드리겠습니다.

10.1 특정 포트로 들어오는 트래픽 허용하기

기본 ssh 포트인 22 번에 대한 TCP 접속을 허용하기 위해 다음과 같이 명령을 수행합니다.

```
$ sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
  또는
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

위와 같이 룰을 추가한 후에 조회하면 아래와 비슷한 결과가 출력됩니다.

```
$ sudo iptables -L
Chain INPUT
(policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp --  anywhere             anywhere            tcp dpt:ssh
```

웹 관련 트래픽을 허용해보겠습니다.

```
$ sudo iptables -A INPUT -p tcp --dport http -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport https -j ACCEPT
```

위 명령은 아래와 같이 수행할 수도 있습니다.

```
$ sudo iptables -A INPUT -p tcp -m multiport --dports http,https -j ACCEPT
```

룰을 추가한 후에 조회하면 아래와 비슷한 결과가 출력됩니다.

```
$ sudo iptables -L
Chain INPUT
(policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:http
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:https
```

10.2 들어오는 트래픽 막기

위와 같이 ssh, http, https 에 대해서만 허용하고, 나머지 모든 들어오는 트래픽에 대해서 막고자 한다면 다음과 같이 명령을 수행할 수 있습니다.

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -L

Chain INPUT
(policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:http
ACCEPT     tcp --  anywhere             anywhere             tcp dpt:https
DROP       all --  anywhere             anywhere
```

-A INPUT 옵션으로 룰을 추가했기 때문에 INPUT 체인의 끝에 룰이 추가되고, 패킷이 들어오면 위에서부터 룰 매칭을 시도하기 때문에, 매치가 되면 바로 액션이 적용되고 다음 룰로 진행하지 않습니다.
즉, ssh 등은 여전히 접속이 됩니다.

10.3 특정 IP에서 들어오는 트래픽 막기

-s 옵션에 IP 를 입력합니다.

```
$ sudo iptables -A INPUT -s 59.45.175.62 -j REJECT
  또는
$ sudo iptables -A INPUT -s 59.45.175.62 -j DROP
```

소스 IP 주소에 CIDR(Classless Inter-Domain Routing) 표현도 가능합니다. 따라서, 59.45.17.0 ~ 59.45.17.255 범위의 모든 IP 를 막고 싶으면 다음과 같이 명령을 수행할 수 있습니다.

```
$ sudo iptables -A INPUT -s 59.45.175.0/24 -j REJECT
```

특정 IP 로 나가는 트래픽을 막고 싶다면, 이것은 INPUT 체인이 아니라 OUTPUT 체인에서 룰을 등록해야 하고 목적지 IP 이므로 -d 옵션을 사용합니다.

```
$ sudo iptables -A OUTPUT -d 31.13.78.35 -j DROP
```

10.4 줄 번호와 같이 룰 조회하기

룰 조회 시에 --line-numbers 옵션을 같이 입력하면 각 체인과 룰을 보여줄 때 라인 앞에 줄 번호도 같이 보여주는데, 이것이 도움이 될 때가 있습니다. (특히, 룰 삭제 시)

```
$ sudo iptables -L --line-numbers
```

결과는 아래와 같이 각 체인 별로 줄 번호가 출력됩니다.

```
Chain INPUT (policy ACCEPT)
num target prot opt source destination

1 DROP all -- 59.45.175.0/24 anywhere
2 DROP all -- 221.194.47.0/24 anywhere
3 DROP all -- 91.197.232.104/29 anywhere

Chain FORWARD (policy ACCEPT)
num target prot opt source destination

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 DROP all -- anywhere 31.13.78.0/24
```

또한, iptables 가 룰에 등록된 IP 들에 대해서 reverse DNS lookup 을 하기 때문에 이게 불필요하거나, 좀 더 빠른 출력을 원한다면 다음과 같이 -n 옵션을 사용합니다.

```
$ sudo iptables -L -n --line-numbers
```

10.5 줄 번호와 같이 룰 삭제하기

추가한 룰을 삭제하고자 한다면, 다음과 같이 -D 옵션을 사용합니다. 즉, 추가할 때 -A 옵션을 사용해서 작성한 커맨드 라인에서 -A 만 -D 로 바꾸면 됩니다.

```
$ sudo iptables -D INPUT -s 221.194.47.0/24 -j REJECT
```

또는 iptables -L --line-numbers 를 이용해서 각 체인에 등록된 룰에 대한 줄 번호가 나오게 한 다음, 아래 명령어로 삭제할 수 있습니다.

```
$ sudo iptables -D INPUT 2
```

줄 번호를 가지고 삭제할 때에는 주의할 점이 룰의 줄 번호가 바뀔 수 있다는 점이다. 예를 들어, 줄 번호 9 와 12, 두 개의 룰을 삭제하고 싶은데, 9 를 먼저 삭제하면 12 번 룰은 11 이 되기 때문입니다. 이 때 다시 iptables -L --line-numbers 명령을 수행해서 최신 줄 번호를 확인해도 되지만, 매번 확인하는 게 번거롭다면 아래와 같이 줄 번호가 큰 것(즉, 제일 아래에 있는 룰)부터 삭제하는 것이 편할 수도 있습니다.

```
$ sudo iptables -D INPUT 12
$ sudo iptables -D INPUT 9
```

특정 테이블의 특정 체인에 있는 모든 룰을 삭제하고 싶다면 아래와 같이 명령을 수행합니다.

```
$ sudo iptables -F INPUT
```

위 명령어는 -t 옵션으로 특정 테이블을 지정하지 않으면 filter table 이 디폴트이기 때문에 모든 룰이 삭제됩니다.

10.6 룰을 특정 위치에 넣기

룰을 INPUT 체인의 첫 번째 줄에 넣고자 한다면, 다음과 같이 명령을 수행합니다.

```
$ sudo iptables -I INPUT 1 -s 59.45.175.10 -j ACCEPT
```

10.7 룰 변경하기

INPUT 체인의 첫 번째 줄에 넣은 룰에 있는 소스 IP 가 실수로 59.45.175.12 로 입력되어 이 IP를 59.45.17.10으로 변경하고자 한다면, 다음과 같이 명령을 수행합니다.

```
$ sudo iptables -R INPUT 1 -s 59.45.175.10 -j ACCEPT
```

10.8 프로토콜 기반 매칭

모든 들어오는 TCP 트래픽을 막고자 한다면, 다음과 같이 명령을 수행합니다.

```
$ sudo iptables -A INPUT -p tcp -j DROP
```

Source IP Range 를 가지고 ssh 접근을 막고자 한다면, 아래와 같이 명령을 수행할 수 있습니다.

```
$ sudo iptables -A INPUT -p tcp -m tcp --dport 22 -s 59.45.175.0/24 -j DROP
```

ssh, vnc 에 대한 접속을 하나의 룰로 막고자 한다면, -m multiport 옵션을 사용해서 아래와 같이 명령을 수행할 수 있습니다.

```
$ sudo iptables -A INPUT -p tcp -m tcp --dport 22 -s 59.45.175.0/24 -j DROP
```

ICMP 프로토콜에 대해서 ICMP address mask requests (type 17)를 막고 싶다면, 아래와 같이 명령을 수행할 수 있습니다.

```
$ sudo iptables -A INPUT -p icmp --icmp-type 17 -j DROP
```

10.9 특정 네트워크 인터페이스 선택하기

Loopback Interface(일반적으로, lo)를 통해서 들어오는 패킷을 허용하고 싶다면, -i 옵션을 사용해서 아래와 같이 명령을 수행합니다. -i 옵션은 input interface 를 나타냅니다.

```
$ sudo iptables -A INPUT -i lo -j ACCEPT
```

OUTPUT 체인에 대해서는 -o 옵션을 사용해야 합니다. -o 옵션은 output interface 를 나타냅니다.

```
$ sudo iptables -A OUTPUT -o wlan0 -d 121.18.238.0/29 -j DROP
```

10.10 사용자 정의 체인

어떤 상황 때문에 아래와 같이 룰들을 입력했다고 가정해보겠습니다.

```
$ sudo iptables -A INPUT -p tcp -m tcp --dport 22 -s 18.130.0.0/16 -j ACCEPT
$ suoo iptables -A INPUT -p tcp -m tcp --dport 22 -s 18.11.0.0/16 -j ACCEPT
$ sudo iptables -A INPUT -p tcp -m tcp --dport 22 -j DROP
```

위의 룰들이 정상적으로 동작하기는 하지만, 뭔가 비효율적이고 깔끔해 보이지 않습니다. 이 룰들을 사용자 정의 체인을 하나 만들어서 다음과 같이 정리할 수 있습니다.
-N 옵션을 사용해서 아래와 같이 새로운 체인을 만들 수 있습니다.

```
$ sudo iptables -N ssh-rules
```

그런 다음, 새로 만든 ssh-rules 체인에 아래와 같이 룰들을 추가할 수 있습니다.

```
$ sudo iptables -A ssh-rules -s 18.130.0.0/16 -j ACCEPT
$ sudo iptables -A ssh-rules -s 18.11.0.0/16 -j ACCEPT
$ sudo iptables -A ssh-rules -j DROP
```

일단, 여기까지 새로운 체인과 그 체인에 들어가는 룰들이 만들어졌습니다. 하지만, 여기서 끝나면 어떤 패킷도 이 체인에 들어가서 매칭 여부를 검사할 수 없습니다. 왜냐하면, 패킷들은 기본적으로 빌트인 체인인 INPUT, OUPUT 등을 타고 흐르기 때문입니다. 따라서, 빌트인 체인인 INPUT 에서 특정 조건이 되면 우리가 만든 체인으로 점프하게 해야 합니다.

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j ssh-rules
```

사용자 정의 체인을 지우고 싶다면, 아래와 같이 명령을 수행합니다. 주의할 점은 체인을 지우기 전에 이 체인을 참조하는 룰들을 모두 삭제해야 한다는 것입니다.

```
$ sudo iptables -X ssh-rules
```

10.11 iptables-save와 iptables-restore

iptables 에 적용된 모든 룰들을 알고 싶다면 iptables -L 명령을 수행해도 되지만, iptables-save 명령어의 출력을 리다이렉트해서 텍스트 파일을 만들 수 있습니다.

```
$ sudo iptables-save > iptables.rules
```

생성한 텍스트 파일을 vi 와 같은 편집기로 열어서 룰 등을 수정/추가/삭제한 다음, 다시 아래 명령어로 파일의 내용을 반영할 수 있습니다.

```
$ sudo iptables-restore < iptables.rules
```

10.12 iptables에 등록된 rule 영구 적용

iptables 명령어를 사용해서 등록한 룰들은 운영 중에 바로 적용 되지만, 재부팅하면 사라집니다. 따라서 재부팅과 상관없이 지속 적용하기 위해서는 각 리눅스 배포본에 따라 아래와 같이 적절한 패키지를 설치해주어야 합니다.

RHEL 계열

```
$ sudo yum install iptables-services
```

Debian 과 Ubuntu 계열

```
$ sudo apt install iptables-persistent
```

만약, 사용하는 배포본이 위와 비슷한 기능을 하는 패키지가 없다면, 직접 서비스 파일을 만들어서 iptables 룰 파일을 시작 시에 로드하고 종료 시에 저장하게 하면 됩니다.

이상으로 Netfilter 에 대한 개념과 iptables 를 활용한 다양한 사용방법에 대해 살펴봤습니다. 클라우드 환경에서 가상 서버의 완벽한 네트워크 라우팅 설정에 도움이 되길 바랍니다.