IPTables

iptables 방화벽은 패킷 필터링 및 NAT(DNAT + SNAT) 규칙을 관리하는 데 사용된다.

DNAT(Destination Network Address Translation)

SNAT(Source Network Address Translation)

iptables는 모든 Linux 배포판과 함께 제공된다.
상위 수준의(high-level) iptables는 여러 테이블(tables)을 포함할 수 있다. 그리고 테이블에는 여러 체인(chains)이 포함될 수 있다.
체인들은 내장되거나 사용자가 정의할 수 있다.
체인은 들어오는 패킷과 나가는 패킷에 대한 규칙(rules)을 여러개 포함할 수 있다.

따라서 구조는 IPTables -> Tables -> Chains -> Rules

iptables(and ip6tables)의 구성파일

iptables 서비스는 /etc/sysconfig/iptables, /etc/sysconfig/ip6tables에 구성을 저장하고,
반면에 firewalld는 /usr/lib/firewalld/, /etc/firewalld/에 다양한 XML 파일에 구성을 저장한다.

/etc/sysconfig/iptables 파일은 Red Hat Enterprise Linux에 기본적으로 firewalld가 설치되어 있으므로 존재하지 않습니다.

iptables와 firewalld 둘 다 iptables 도구를 사용하여 커널 패킷 필터(Netfilter)와 통신한다.
firewalld 대신 iptables(또는 ip6tables) 서비스를 사용하려면 루트(root)로 다음 명령을 실행하여 firewalld를 비활성화한다.

```
# systemctl disable firewalld
# systemctl stop firewalld
```

Netfilter Hooks

Netfilter는 사용자 공간의 애플리케이션들이 패킷을 처리할 때 커널 네트워크 스택에 의해 적용되는 처리 규칙(Rules)을 등록할 수 있게 한다. (forwarding과 filtering이 가능하게 한다.)

모든 패킷은 networking 시스템으로 들어오는데, 들어올 때 이러한 Hook(Netfilter)를 Trigger 하게 되고, 패킷들은 그 때 스택을 통해 통과된다.

많은 일반적인 호스트 방화벽 애플리케이션과 Kubernetes 서비스 포워딩이 iptables로 구현된다.

Netfilter Hooks	
Hook	***설명***
NF_IP_PRE_ROUTING	***패킷이 네트워크 스택에 들어오고 난 바로 직 후, 초기 시점에 incoming 트래픽에 의해 트리거 될 것인데, 이 Hook은 이후 라우팅 결정 이전에 처리 된다.***
NF_IP_LOCAL_IN	***만약 패킷이 local system으로 목적지를 가지게 될 경우 incoming 패킷이 해당 목적지로 라우팅 되고 난 이후에 트리거 된다.***
NF_IP_FORWARD	***만약 패킷이 다른 host로 포워딩 될 경우, incoming 패킷이 해당 호스트로 라우팅 되고 난 이후 트리거 된다.***
NF_IP_LOCAL_OUT	***네트워크 스택을 만나자 마자, 로컬에 생성되어진 Outbound 트래픽에 의해 트리거 된다.***
NF_IP_POST_ROUTING	***라우팅이 발생하고 난 이후와 실제 네트워크에 놓여지기 바로 직전 어떤 outgoing이나 포워딩 된 트래픽에 의해 트리거 된다.***

Tables와 Chains

테이블은 사용자가 익숙한 형태에 따라 규칙(Rules)을 구분한다. 예들 들어 만약 하나의 규칙이 network 주소 변환을 다룬다면 그것은 nat 테이블로 놓여질 것이고, 규칙이 패킷을 목적지로 허용하는데 사용된다면 그것은 filter 테이블에 추가 될 것이다.
이러한 각각의 iptables 테이블 내에서 규칙들은 체인(chain)이라 불리는 그룹으로 묶여서 관리된다.
iptables에 내장된 대표적인 빌트인 체인(Built-in Chains)들은 앞서 소개한 netfilter hook과 1:1 대응 관계에 있다. 따라서, netfilter에서 특정 훅이 실행될 때, iptables에서 해당 훅에 대응되는 체인에 등록된 룰이 실행된다.

- PREROUTING : NF_IP_PRE_ROUTING Hook에 의해 트리거 된다.
  (이 체인의 규칙(Rules)은 패킷이 네트워크 인터페이스에 막 도착할 때 패킷에 적용 -> 커널에서 라우팅 결정을 내리기 전에 취해야 할 액션을 정의)
- INPUT : NF_IP_LOCAL_IN Hook에 의해 트리거 된다.
  (이 체인의 규칙(Rules)은 로컬 프로세스에 전달되기 직전에 패킷에 적용)
- FORWARD : NF_IP_FORWARD Hook에 의해 트리거 된다.
  (현재 호스트를 통해 라우팅되는 모든 패킷에 적용)

- OUTPUT : NF_IP_LOCAL_OUT Hook에 의해 트리거 된다.
  (이 체인의 규칙(Rules)은 로컬 프로세스에서 패킷이 생성된 직후 패킷에 적용)
- POSTROUTING : NF_IP_POST_ROUTING Hook에 의해 트리거 된다.
  (이 체인의 규칙(Rules)은 패킷이 네트워크 인터페이스를 떠날 때 적용 -> 커널이 취한 라우팅 결정 후에 취해야 하는 액션을 정의)

각 테이블에 구현된 체인

테이블	*설명*
filter	*필터 테이블은 패킷 필터링에만 사용해야 합니다. 다른 테이블에서와 마찬가지로 문제 없이 패킷을 DROP, LOG, ACCEPT 또는 REJECT할 수 있습니다. filter 테이블에는 3개의 빌트인 체인이 내장되어 있다. INPUT 체인은 로컬 호스트(방화벽)로 향하는 모든 패킷에 사용된다. FORWARD 체인은 로컬 호스트로 향하지도 않고, 로컬에서 생성되지도 않은 모든 패킷에 사용된다. OUTPUT 체인은 최종적으로 로컬에서 생성된 모든 패킷에 사용된다.*
nat	*nat 테이블은 주로 NAT(Network Address Translation : 네트워크 주소 변환)에 사용된다. NAT 처리된 패킷들은 Rule에 따라 IP 주소가 변경된다. 스트림(Stream)에 있는 패킷들은 이 테이블을 한 번 만 통과한다. 스트림의 첫 번째 패킷이 허용된다고 가정할 때, 같은 스트림의 나머지 패킷은 자동으로 NAT 처리되거나 마스커레이딩 등으로 처리되며 첫 번째 패킷과 동일한 작업이 적용된다. 스트림의 패킷들은 nat 테이블을 다시 통과하지 않지만 그럼에도 불구하고 첫 번째 패킷처럼 취급된다. 이것이 지금 보고있는 이 표에서 필터링을 수행하지 말아야 하는 주된 이유이며, 이에 대해서는 더 자세히 논의할 것이다. PREROUTING 체인은 패킷이 방화벽에 들어가자마자 패킷을 변경하는 데 사용된다. OUTPUT 체인은 라우팅 결정에 도달하기 전에 로컬에서 생성된 패킷을 변경하는데 사용된다. POSTROUTING 체인은 패킷이 방화벽을 떠나려고 할 때 패킷을 변경하는 데 사용된다.*
mangle	*패킷을 mangling? 하는데 사용한다. 서로 다른 패킷들의 내용(contents)과 헤더(header)를 변경할 수 있다. 예를 들어 패킷의 TTL(Time To Live), TOS(Type Of Service) 또는 MARK를 변경하는 것이다.(MARK는 실제로 패킷에 대한 변경은 아니지만 패킷에 대한 mark값은 커널 공간에서 설정된다) 다른 Rule 또는 프로그램들 더 나아가 방화벽에서 이 mark값을 사용하여 필터링하거나 고급 라우팅을 수행할 수 있다. tc가 그 예이다. mangle 테이블은 5개의 빌트인 체인으로 구성되어 있다. PREROUTING 체인은 패킷이 방화벽에 들어가자마자 패킷을 변경하거나 라우팅 결정에 도달하기 전에 패킷을 변경하는 데 사용된다. POSTROUTING 체인은 모든 라우팅 결정이 내려진 직후 패킷을 mangling하는데 사용된다. OUTPUT 체인은 라우팅 결정에 들어간 후 로컬에서 생성된 패킷을 변경하는 데 사용된다. INPUT 체인은 패킷이 로컬 컴퓨터(자기 자신)로 라우팅된 후 사용자 공간 응용 프로그램이 실제로 데이터를 보기 전에 패킷을 변경하는 데 사용된다. FORWARD 체인은 첫 번째 라우팅 결정에 도달한 후 실제로 마지막 라우팅 결정에 도달하기 전에 패킷을 mangling하는데 사용된다.*
raw	*raw테이블과 raw테이블의 체인들은 netfilter의 다른 테이블보다 먼저 사용된다. NOTRACK target을 사용하기 위해 도입되었다. raw테이블은 다소 새로운 테이블이고, 커널 컴파일을 했다면 2.6 커널 이상에서만 사용할 수 있다. raw테이블은 두 개의 체인(PREROUTING, OUTPUT)이 있고 이 체인들은 netfilter 하위 시스템에 도달하기 전에 패킷을 처리한다. PREROUTING 체인은 이 시스템으로 들어오는 모든 패킷 또는 전달되는(forwarded) 모든 패킷에 사용된다. OUTPUT 체인은 netfilter 하위 시스템에 도달하기 전에 로컬에서 생성된 패킷을 변경하는 데 사용된다.*

Table / Chains	*PREROUTING*	*INPUT*	*FORWARD*	*OUTPUT*	*POSTROUTING*
(routing decision)				✔	
raw	✔			✔	
(connection tracking enabled)	✔			✔	
mangle	✔	✔	✔	✔	✔
nat (DNAT)	✔			✔	
(routing decision)	✔			✔	
filter		✔	✔	✔	
security		✔	✔	✔	
nat (SNAT)		✔			✔

체인 순회 순서 - Chain Traversal Order

패킷이 처음 방화벽에 들어가면 하드웨어에 도달한 다음 커널의 적절한 장치 드라이버로 전달됩니다.
그런 다음 패킷은 올바른 응용 프로그램(로컬)으로 전송되거나 다른 호스트로 전달되기 전에 커널에서 일련의 단계를 거치기 시작합니다.

로컬 시스템으로 향하는 수신 패킷

STEP	Table	Chain	설명
1			On the wire (e.g., Internet) 패킷이 아직 인터넷상에 있음
2			Comes in on the interface (e.g., eth0) 패킷이 네트워크 인터페이스에 들어옴
3	raw	PREROUTING	이 체인은 연결 추적(Connection Tracking)이 발생하기 전에 패킷을 처리하는 데 사용된다. 예를 들어 연결 추적 코드(Connection Tracking Code)에 의해 처리되지 않도록 특정 연결을 설정하는 데 사용할 수 있다.
4			이 때 연결 추적 코드가 발생합니다.
5	mangle	PREROUTING	패킷을 mangling 한다 예를 들어 TOS 변경 등
6	nat	PREROUTING	이 체인은 주로 DNAT에 사용됩니다. 경우에 따라 우회되므로 이 체인에서 필터링하지 마십시오.
7			라우팅 결정(Routing decision), 즉 로컬 호스트로 향하거나 어딘가로 포워딩 될지 패킷의 운명이 결정된다
8	mangle	INPUT	이 체인을 사용하여 라우팅된 후 머신의 프로세스로 실제로 전송되기 전에 패킷을 mangling 한다.
9	filter	INPUT	여기서 로컬 호스트로 향하는 모든 수신 트래픽을 필터링한다. 이 호스트로 향하는 모든 수신 패킷들은 어디서(destination) 왔는지, 어떤 인터페이스(interface)로부터 왔는지에 관계없이 이 체인을 통과한다
10			Local process or application (i.e., server or client program) 로컬 프로세스 또는 애플리케이션(즉, 서버 또는 클라이언트 프로그램)

로컬에서 생성된 패킷

STEP	Table	Chain	설명
1			Local process/application (i.e., server/client program) 로컬 프로세스/애플리케이션(즉, 서버/클라이언트 프로그램)
2			라우팅 결정(Routing decision)을 한다. 사용할 출발지 주소(Source Addr), 사용할 발신 인터페이스(outgoing interface) 및 수집해야 하는 기타 필수 정보
3	raw	OUTPUT	이 체인에서 로컬로 생성된 패킷에 대한 연결 추적이 발생하기 전에 작업을 수행한다. 예를 들어 연결이 추적되지 않도록 연결을 mark 할 수 있다.
4			여기에서 로컬로 생성된 패킷에 대한 연결 추적이 수행됩니다.
5	mangle	OUTPUT	패킷을 mangling 한다. 부작용이 있을 수 있으므로 이 체인에서 필터링하지 않는 것이 좋습니다.
6	nat	OUTPUT	이 체인은 방화벽 자기 자신으로부터 나가는(outgoing) 패킷을 NAT처리 하는데 사용할 수 있다.
7			라우팅 결정을 이미 했지만 그 이후에 mangle 및 nat 변경으로 인해 패킷 라우팅 방법이 변경되었을 수 있으므로 라우팅 결정(Routing decision)을 또 한다.
8	filter	OUTPUT	로컬 호스트에서 나가는 패킷을 필터링한다.
9	mangle	POSTROUTING	mangle 테이블의 POSTROUTING 체인은 주로 패킷이 호스트를 떠나기 전에 패킷을 mangling 하는데 사용된다. 그러나 실제 라우팅 결정 후에. 이 체인은 로컬에서 생성된 패킷, 단지 방화벽을 통과하기만 하는 패킷(just traversing the firewall) 모두에게 적중된다.
10	nat	POSTROUTING	앞서 설명한 대로 SNAT를 수행하는 곳이다. 부작용(side effects)이 있을 수 있고 기본 정책인(default policy) DROP을 설정하더라도 특정 패킷이 통과할 수 있으므로 여기서 필터링을 하지 않는 것이 좋다.
11			Goes out on some interface (e.g., eth0) 네트워크 인터페이스를 통해 밖으로 나간다
12			On the wire (e.g., Internet) 인터넷에 돌아다닌다

다른 호스트로 향하는 수신 패킷

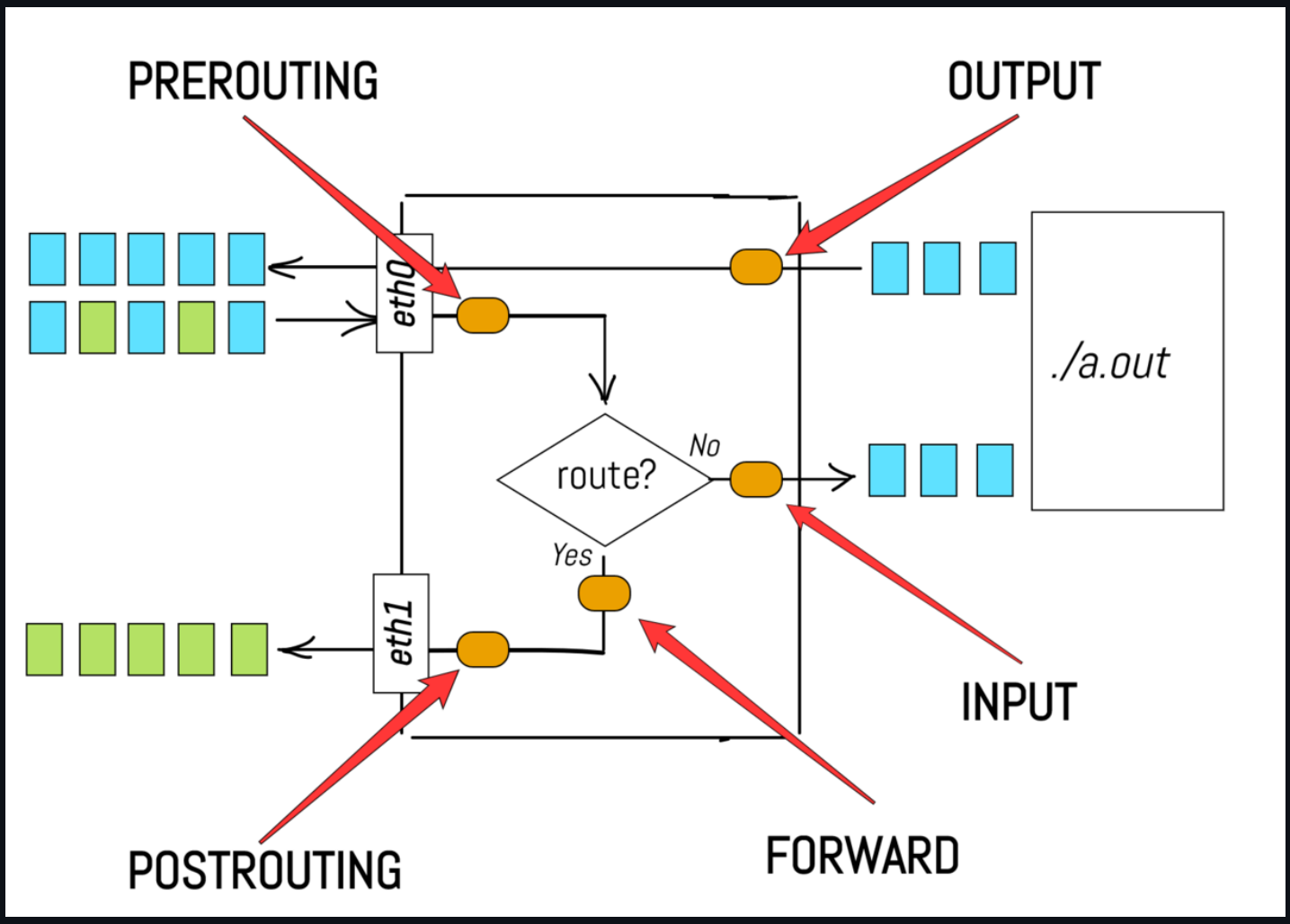
STEP	Table	Chain	설명
1			On the wire (e.g., Internet) 패킷이 아직 인터넷상에 있음
2			Comes in on the interface (e.g., eth0) 패킷이 네트워크 인터페이스에 들어옴
3	raw	PREROUTING	이 체인은 연결 추적(Connection Tracking)이 발생하기 전에 패킷을 처리하는 데 사용된다. 예를 들어 연결 추적 코드(Connection Tracking Code)에 의해 처리되지 않도록 특정 연결을 설정하는 데 사용할 수 있다.
4			여기서 로컬에서 생성되지 않은 연결 추적이 발생한다.
5	mangle	PREROUTING	이 체인은 패킷을 mangling 한다 예를 들어 TOS 변경 등
6	nat	PREROUTING	이 체인은 주로 DNAT에 사용됩니다. 경우에 따라 우회되므로 이 체인에서 필터링하지 마십시오.
7			라우팅 결정(Routing decision), 즉 로컬 호스트로 향하거나 어딘가로 포워딩 될지 패킷의 운명이 결정된다
8	mangle	FORWARD	패킷이 mangle 테이블의 FORWARD 체인으로 전달되면 이것은 초기 라우팅 결정 후, 마지막 라우팅 결정 전에 패킷을 mangling 할 수 있다.
9	filter	FORWARD	패킷은 FORWARD 체인으로 라우팅된다. 여기서 모든 필터링을 수행한다. 이 체인으로 전달되는 모든 트래픽이 다 같은 방향으로 FORWARD 되는 것은 아니기 때문에 (not only one direction) rule-set을 작성할 때 이를 고려해야 한다.
10	mangle	POSTROUTING	패킷이 호스트를 떠나기 전에 패킷을 mangling 하는데 사용된다. 이 체인은 모든 종류의 라우팅 결정이 완료된 후에 사용된다.
11	nat	POSTROUTING	앞서 설명한 대로 SNAT를 수행하는 곳이다. 특정 패킷이 통과할 수 있으므로 여기서 필터링을 하지 않는 것이 좋다. 여기서로 마스커레이딩(Masquerading)이 수행된다.
12			Goes out on the outgoing interface (i.e., eth1). 발신 네트워크 인터페이스에서 나간다.

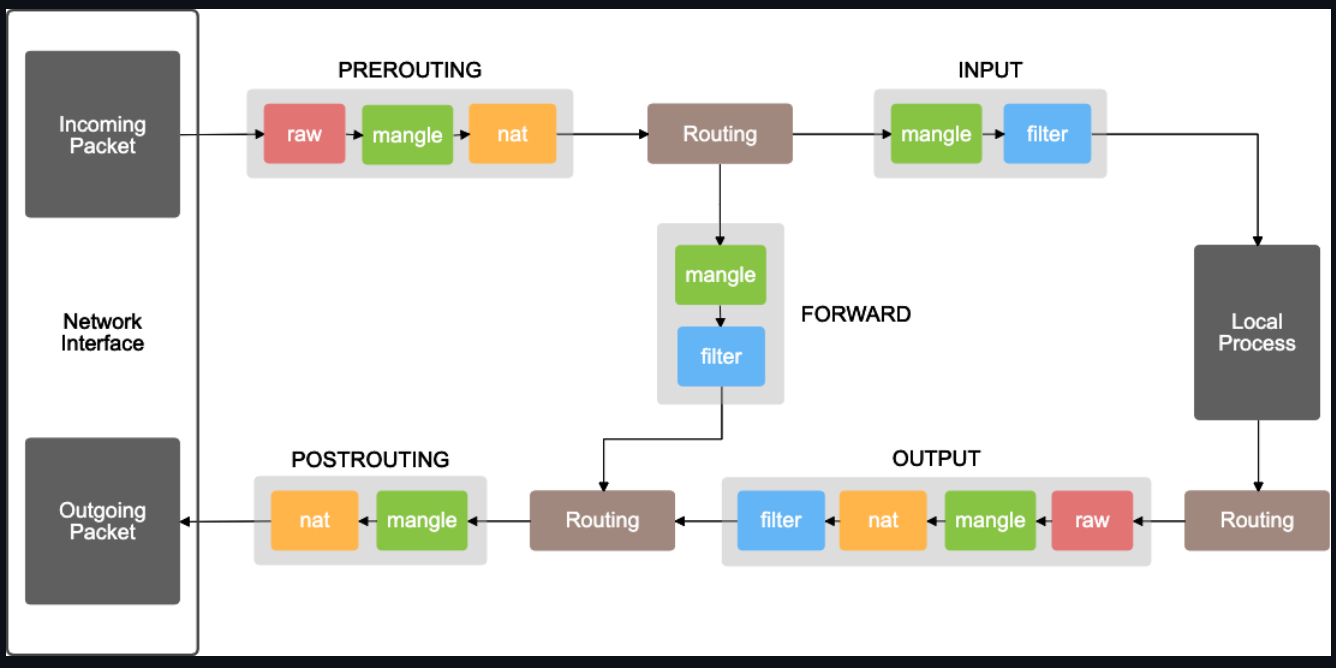
STEP	Table	Chain	설명
13			*Out on the wire again (i.e., LAN).* *패킷이 LAN으로 나가 돌아다닌다*

- 로컬 시스템으로 향하는 수신 패킷 PREROUTING -> INPUT
- 다른 호스트로 향하는 수신 패킷 PREROUTING -> FORWARD -> POSTROUTING
- 로컬에서 생성된 패킷 OUTPUT -> POSTROUTING

위 표에서 로컬 시스템으로 향하는 수신 패킷이 먼저 raw, mangle, nat 테이블의 PREROUTING 체인에 대해 평가됨을 알 수 있다.

그런 다음 최종적으로 로컬 소켓에 전달되기 전에 mangle, filter, security, nat 테이블의 INPUT 체인을 통과한다.





Rules

- 규칙(Rules)은 특정 테이블의 특정 체인 내에 배치된다.
- 각 체인이 호출되면 해당 패킷이 체인 내의 각 규칙에 대해 순서대로 확인됩니다.

- 각각의 규칙(Rules)은 프로토콜 유형, 목적지/출발지 주소, 목적지/출발지 포트, 목적지/출발지 네트워크, 입/출력 인터페이스, 헤더, 또는 다른 기준 중에서 연결 상태에 따라 일치하도록 규칙을 구성할 수 있다.
- 이것들을 결합하여 서로 다른 트래픽을 구별하는 상당히 복잡한 규칙 세트를 생성할 수 있다.

Matching

패킷이 충족해야 하는 "기준"이다.(Rule를 적용하기 위한 조건에 해당하는 부분)
특정 패킷이 현재 Rule에 부합한 경우, target에 등록된 구체적인 행동이 적용된다.

Targets

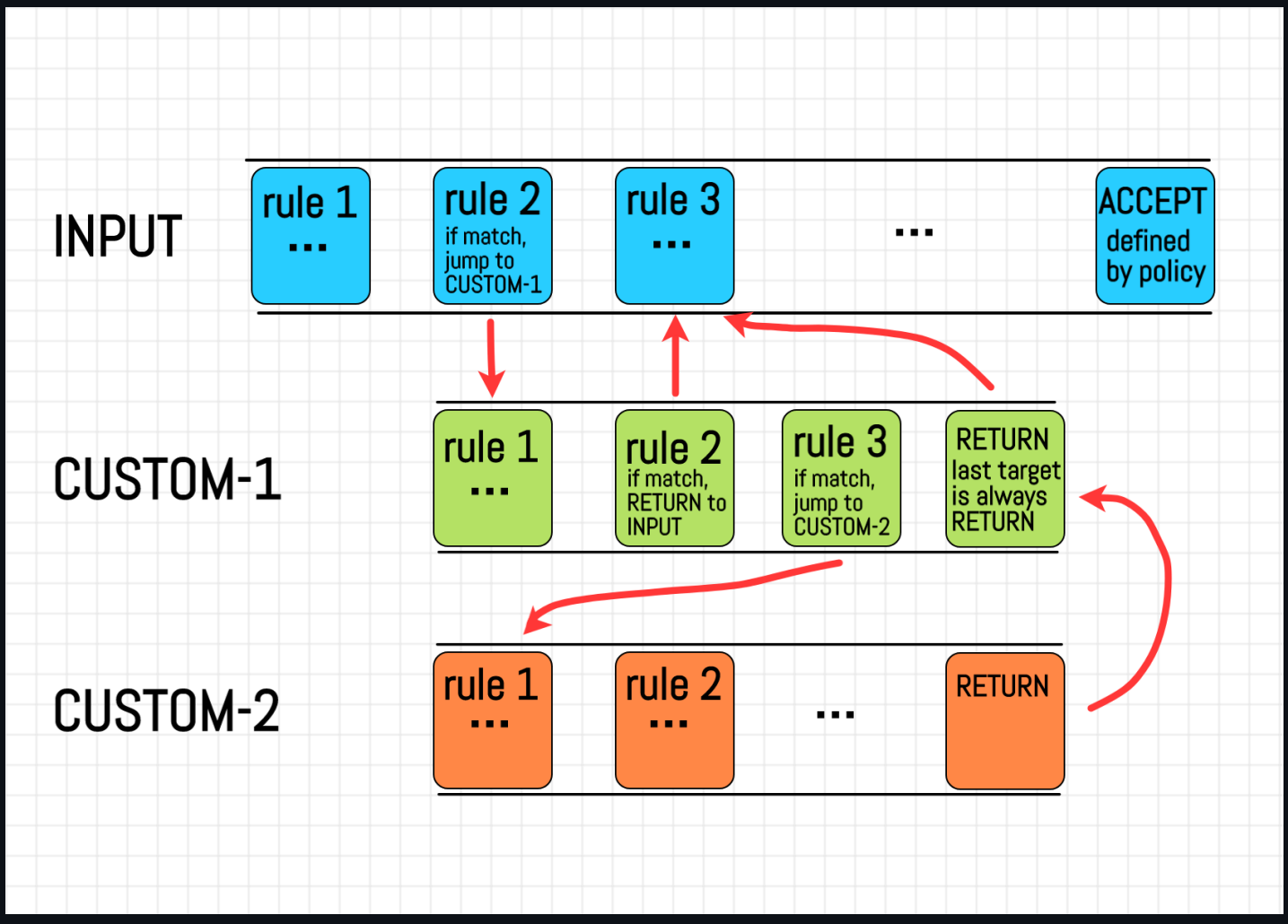
패킷이 Rule의 일치 기준(matching criteria)을 충족할 때 패킷을 어떻게 처리할지를 결정하는 실질적인 행위(action)에 해당하는 파트이다.
===== 주요 Target 예 =====

- ACCEPT : 패킷을 수신하고 해당 요청을 허용
- DROP : 패킷의 승인을 보내지 않고 요청을 삭제
- REJECT : 패킷을 거부하며 승인도 보냄
- LOG
- RETURN : 패킷을 원래 체인으로 다시 보내므로 다른 규칙(Rules)과 일치시킬 수 있다.

Custom Chains(사용자 정의 체인)

사용자가 직접 자체 체인을 만들 수도 있다.
규칙(Rules)들은 빌트인 체인과 마찬가지로 사용자 정의 체인에도 동일한 방법으로 배치할 수 있다.
차이점은 커스텀 체인은 규칙에서 "점프"하는 방식으로 도달할 수 있다.
커스텀 체인은 이를 호출한 체인의 단순한 확장 역할을 한다.
커스텀 체인 안에서 규칙 목록의 끝에 도달하거나, 규칙 목록 중에서 특정 rule에 매칭되어 return 대상이 활성화되면 평가(Evaluation)결과는 커스텀 체인을 호출한 체인으로 다시 전달된다.
평가는 커스텀 체인 안에서 다시 다른 커스텀 체인으로 이동할 수도 있다.

1. 실행할 체인의 규칙을 가리키는 포인터를 생각하면, 포인터의 체인 순회(Traversal)는 특정 Rule에 매칭되거나 FORWARD, INPUT과 같은 built-in 체인에 의해 끝날 때까지 규칙에서 규칙으로, 위에서 아래로 순회를 계속한다.
2. 동일한 테이블 내에서 다른 체인에 대한 점프 규칙을 지정할 수 있다.
3. built-in 체인과 다르게 커스텀 체인은 체인 끝에 기본정책(default policy)을 가질 수 없다. (일치하는 규칙이 없다면 기본 동작은 다시 원래 체인으로 점프하는 것이다)



Connection Tracking(연결 추적)

Netfilter 프레임워크가 특정 연결 상태를 알 수 있도록 연결 추적이 수행된다. 이를 구현하는 방화벽을 상태 저장 방화벽(stateful firewall)이라고 한다. 모든 연결 추적은 conntrack이라는 커널 내의 특수 프레임워크에 의해 수행된다.
아래의 상태(states)들은 주로 상태 일치(state match)에 사용되며, 연결 추적(Connection Tracking)에 필요하다.
OUTPUT 체인에서 처리되는 로컬 생성 패킷을 제외하고 모든 연결 추적은 PREROUTING 체인에서 처리됩니다.
OUTPUT 체인에서 초기 패킷을 보내면 상태가 NEW로 설정되고,
PREROUTING 체인에서 리턴 패킷을 수신하면 상태가 ESTABLISHED로 변경되는 식이다.
모든 상태 변경 및 계산은 nat 테이블의 PREROUTING, OUTPUT 체인 내에서 수행된다.

iptables에서 추적한 연결은 다음 상태중 하나이다.

- NEW : 기존 연결과 연결되지 않았지만 첫 번째 패킷으로 유효하지 않은 패킷이 도착하면, 새로운 커넥션이 "NEW" 레이블로 추가된다.
  특정 연결 내에서 conntrack 모듈이 보는 첫 번째 패킷이 일치함을 의미한다.
  예를 들어 SYN 패킷을 보고 그건이 우리가 보는 연결의 첫 번째 패킷이라면 일치할 것이다. 그래서 NEW로 간주된다.
- ESTABLISHED : NEW 상태는 응답 패킷을 수신하거나 방화벽을 통해 ESTABLISHED 상태로 변경된다.
  여기서 응답은 TCP 연결의 경우 SYN/ACK를 의미하고 UDP 및 ICMP 트래픽의 경우 원본 패킷의 source와 Destination이 전환되는 응답을 의미합니다.
  ESTABLISHED 상태는 양방향 트래픽을 확인한 다음 패킷을 지속적으로 일치시킨다.
  ESTABLISHED 상태가 되기 위한 유일한 요구 사항은 한 호스트가 패킷을 보내고 나중에 다른 호스트로부터 응답을 받는 것
- RELATED : 패킷이 기존 연결의 일부가 아니지만, 시스템에 이미 있는 연결과 관련된 패킷은 "RELATED" 레이블이 지정된다. (연결이 RELATED로 간주되려면 먼저 ESTABLISHED로 간주되는 연결이 있어야 한다.)
  이것은 FTP 데이터 전송 연결의 경우처럼 도우미 연결을 의미하거나 또는 다른 프로토콜로 연결 시도에 대한 ICMP 응답일 수 있다.
- INVALID : 패킷이 기존 연결과 관련이 없고, 새 연결을 여는 데 적합하지 않거나, 식별할 수 없거나, 다른 이유로 라우팅할 수 없는 경우 패킷을 INVALID로 표시
- UNTRACKED : 패킷이 raw 테이블 내에서 target이 NOTRACK으로 표시되는 경우 해당 패킷은 UNTRACKED로 표시된다.
  NOTRACK target은 이 규칙과 일치하는 모든 패킷에 대한 연결 추적을 끄는 데 사용된다.(패킷 추적을 우회하기 위해 사용한다.)
- SNAT : NAT 작업에 의해 패킷의 source address가 변경되었을 때 설정된 가상상태.
  이것은 응답 패킷이 도착했을 때 source address를 다시 원래대로 되돌려 놓기 위함
- DNAT : NAT 작업에 의해 패킷의 destination address가 변경되었을 때 설정되는 가상상태.
  이것은 응답 패킷이 도착했을 때 destination address를 다시 원래대로 되돌려 놓기 위함