리눅스 2.4 패킷 필터링 하우투

Rusty Russell, mailing list netfilter@lists.samba.org v1.0 Tue Mar 21 23:29:42 EST 2000

이 문서는 2.4 리눅스 커널에서 잘못된 패킷을 필터링하기 위하여 어떻게 iptables 을 이용하는가를 기술한다.

목 차

제 1 절 소개 **2**

제 2 절 웹사이트와 리스트가 어디에 있나 ? **2**

제 3 절 그렇다면, 패킷 필터란 무었일까? **2**

3.1 왜 우리는 패킷을 필터할려고 하나 ? . . . 3

3.2 리눅스에서 패킷 필터는 어떻게 하나 ? . . . 3

3.2.1 iptables . . . 4

3.2.2 규칙들을 영속시키기 . . . 4

제 4 절 당신은 누구며 왜 나의 커널을 가지고 놀려고 하나 ? **4**

제 5 절 Rusty's 의 패킷 필터링에 대한 총알 가이드 **4**

제 6 절 패킷이 필터를 어떻게 지나는가 ? **5**

제 7 절 iptables 사용하기 **5**

7.1 컴퓨터가 부팅될때 여러분이 보게 되는 것 . . . 6

7.2 하나의 규칙으로 작동하기 . . . 6

7.3 필터링 지정 . . . 7

7.3.1 출처와 목적지 지정 . . . 7

7.3.2 '역'의 경우 지정 . . . 8

7.3.3 프로토콜 지정 . . . 8

7.3.4 인터페이서 지정 . . . 8

7.3.5 분절 (Fragments) 지정 . . . 8

7.3.6 iptables 의 확장 : 새로운 대상(Matches) . . . 9

7.4 타겟 지정 . . . 12

7.4.1 사용자 지정의 체인들 . . . 12

7.4.2 iptables로의 확장 : 새로운 타겟 . . . 13

7.4.3 특별한 미리 만들어진 타겟 14
7.5 전체 체인에 대한 작용. 14
7.5.1 새오룬 체인 생성 14
7.5.2 체인 제거 14
7.5.3 체인 비우기 14
7.5.4 체인 규칙 나열하기 15
7.5.5 카운트 리셋트 ('0'으로 만들기) 15
7.5.6 정책 설정 15
7.5.7 ipchains와 ipfwadm을 사용하기 15

제 8 절 iptables와 ipchains의 차이점 16

제 1 절 소개

여러분 환영합니다.

이 문서는 여러분이 IP address, network address, netmask, routing, DNS 가 무었인지 알고 있다고 가정합니다. 그렇지 않다면 '네트워크 개념 하우투' 를 읽기를 권유합니다.

이 하우투는 상냥한 소개(이게 여러분을 열받게 하고 지금은 흐리멍텅하고 그러나 무방비인)와 가공되지않은 완전 노출(which would leave all but the hardiest souls confused, paranoid and seeking heavy weaponry)사이를 넘 나들 것이다.

여러분의 네트워크는 **안전하지 않다**. 빠르고, 편안하면서 그 사용이 좋은 쪽으로만 하도록하고 악한 시도를 허락하지 않도록 하려는 것은 복잡한 영 화관에서 자유로운 대화는 허락하면서 "불이야"하고 외치는 것은 불허하는 것처럼 거의 해결불능의 문제와 같다. 이것에대한 해답은 이 하우투에서 구할 수 없을 것이다.

여러분이 할 수있는 것은 그 절충점을 결정하는 일이다. 나는 이러한 목적 으로 사용할 수 있는 몇몇 도구와 경계하여야할 약점에 대하여 여러분이 좋은 목적으로 사용하고 악의있는 목적으로 사용하지 않기를 바라며, 알려 주려고 한다. 또다른 어려운 문제이다.

제 2 절 웹사이트와 리스트가 어디에 있나 ?

세개의 웹사이트가 있다.

- Thanks to *Penguin Computing* <http://antarctica.penguincomputing.com/~netfilter/>
- Thanks to *The Samba Team and SGI* <http://www.samba.org/netfilter>.
- Thanks to *Jim Pick* <http://netfilter.kernelnotes.org>.

netfilter의 메일링 리스트를 위해서는 *삼바의 리스트 서버* <http://list.samba.org>를 보라.

제 3 절 그렇다면, 패킷 필터란 무었일까?

패킷필터는 지나가는 패킷의 *헤더*를 살펴보고 그 전체 패킷의 운명을 결정하는 소프트웨어의 일부이다. 이것은 패킷을 **'DROP'**(즉, 마치 전혀 전달되지도 못 했던것 처럼 패킷을 거부) 하던가, **'ACCEPT'**(즉, 패킷이 지나가도록 내버려 둠) 하던가 또는 다른 더욱 복잡한 무엇을 할 것인가를 결정할 것이다.

리눅스에서 패킷 필터링은 커널 내부에 구성되고(커널의 모듈로서 또는 그 내부에 포함 되는 형태이다), 우리가 패킷으로 해야할 몇몇 복잡한 것이 있다. 그러나, 그 패킷의 헤더를 관찰하고 그 패킷의 운명을 결정하는 기본 원칙은 여전히 적용 된다.

3.1 왜 우리는 패킷을 필터할려고 하나 ?

제어, 보안, 관찰가능성

제어:

여러분이 내부 네트워크에서 다른 네트워크로 리눅스 박스를 이용하여 접속을 하고자 할때(소위, 인터넷) 여러분은 어떤형태의 전송은 가능하게 하고 다른것은 불가능하게 할 기회를 가진다. 예를 들어, 패킷 헤더에는 목적지의 주소를 포함하고 있고 이것으로 패킷이 바깥 네트워그의 다른곳 으로 가지 않도록 한다. 다른 예로, 나는 Dilbert archives를 호출하기 위하여 넷스케잎을 이용한다. 그곳의 웹페이지에는 doubleclick.net으로 부터의 광고가 있고 넷스케잎은 그 광고를 받기위하여 나의 시간을 소비한 다. doubleclick.net의 주소로 가거나 또는 그곳에서 오는 어떠한 패킷도 허락하지 않도록 패킷필터에게 이야기 해 놓음으로 이 문제를 해결할 수 있다. (이렇게 하는 더 좋은 방법도 있다 : Junkbuster를 보세요)

보안:

여러분의 멋지고, 잘 정돈된 네트워크와 인터넷의 혼돈사이에 리눅스 박스 만이 있다면, 여러분의 네트워크로 들어오려는 것을 억제할 수 있다는 것은 근사한 일이다. 예를들어, 여러분은 여러분의 네트워크로부터 나가는 모든 것을 허용하고 싶지만, 반면에 밖으로부터 들어오는, "죽음의 핑" 같은, 악의있는 것에 대하여는 것정이 될 것이다. 다른 예로, 아무리 여러분 리눅스 박스의 모든 계정사용자가 암호를 가지고 있다고 하더도 바깥으로부터의 텔넷시도는 바라지 않을 것이다. 대부분의 사람들처럼 인터넷에서 구경꾼 이 되고 싶고 제공자는 되고싶지 않을 것이다. 간단히 말해서, 접속중에 모든 들어오려는 패킷을 패킷 필터를 이용하거 거부할려고 할 것이다.

관찰가능성:

가끔 잘못 설정된 지역네트워크는 패킷을 바깥세상으로 토해놓는다. 패킷 필터에게 어떠한 이상한 일이라도 일어나면 여러분에게 알려 주도록 말해 두는 것은 근사한 일이다. 여러분은 이런 일에대하여 무엇인가를 할 수도 있고 그냥 '이상한 일이네'하고 넘길 수도 있다.

3.2 리눅스에서 패킷 필터는 어떻게 하나 ?

1.1 시리즈 부터 리눅스 커널은 패킷 필터링을 포함하기 시작했다. 제 1세대는 BSD의 ipfw를 기본으로 하였고 1994년 후반기에 Alan Cox에 의해서 포트 되었다. 이것은 리눅스 2.0에서 Jos Vos와 다른이들에 의해서 개선되었고 커널의 필터링 규칙을 제어하는 사용자 툴로는 'ipfwadm'이 사용되었다. 1998년 중반에 리눅스 2.2를 위하여 나는 Michael Neuling의 도움으로 커널에 대하여 열심히 일하였고 사용자 툴료는 'ipchains'를 내놓았다. 마지막으로, 제 4세대 툴이 'iptables'이고 리눅스 2.4를 위하여 1999년 중반에 커널을 제 작성 하였다. 이 하우투 문서가 촛점을 맞추고 있는 것이 이 iptables 에 대한 내용이다.

netfilter를 가지고있는 커널이 필요하다. netfilter는 다른 것들(iptables 모듈 같은)이 붙을수 있는 리눅스 커널의 일반적인 기본 구조이다. 이것은 2.3.15 이상 의 리눅스 커널에 들어있고 커널 설정에서 CONFIG_NETFILTER 에 'Y'로 대답하고 컴 파일한 것이어야 한다.

`iptables`라는 툴은 커널에게 어떤 패킷을 필터할 것인지를 알려준다. 여러분이 프 로그래머나 변태가 아니라면, 이것이 패킷 필터링을 제어하는 수단이다.

3.2.1 iptables

iptables 는 커널의 패킷 필터링 테이블에 필터링 규칙을 삽입하거나 삭제하는 도구 이다. 이것은 여러분이 무었을 설정했든지, 재부팅시에는 소실된다는 것을 의미한다. 다음번 리눅스가 다시 부팅되었을때 설정내용들이 재설치 되기를 바란다면 3.2.2 (규칙들을 영속시키기)를 보아라

iptables는 ipfwadm과 ipchains를 대치한다. 손실없이 iptables 의 사용을 피하고 싶 다면 7.5.7 (ipchains와 ipfwadm 사용하기)를 보아라.

3.2.2 규칙들을 영속시키기

여러분의 파이어월 설정은 커널에 저장되므로 재부팅시에 손실된다. iptables-save 와 iptables-restore을 구현하는 것은 나의 TODO 리스트에 있다. 이것들이 나오게 되면 정말 좋을 것이다. 약속한다.

그동안은 여러분의 규칙을 설정하는 명령들을 초기화 스크립트에 기록해야 한다. 명 령들중 하나가 실패하였을때를 대비하여 뭔가 이성적인것을 해두어야 한다. (보통 'exec /sbin/sulogin'을 사용한다.)

제 4 절 당신은 누구며 왜 나의 커널을 가지고 놀려고 하나 ?

나는 Rusty이다. 리눅스 IP 파이어월 유지하는 사람이고 적절한 시기에 적절한 장소에 있게된 그냥 워킹코더일 뿐이다. 나는 ipchains를 맹글었다. (실제적인 작업을 한 사람을 볼려면 "How Do I Packet Filter Under Linux?"라는 문서를 보라), 그리고 이번에 패킷 필터링을 구할 수 있을 정도의 많은 것을 배웠다.

WatchGuard <http://www.watchguard.com>는 정말 훌륭한 파이어월 회사이며, 정말 멋진 플럭인 파이어박스를 판매하며 아무것도 하지 않아도 나에게 월급을 준다. 덕분에 나는 이 작품을 만드는데 나의 모든 시간을 소모할 수 있었고, 이전의 작업도 그러하다. 나는 6개월이면 끝마치리라 짐작했지만 12개월이 걸렸다. 그러나 내가 바르게 했다는 생각이 든다. 수많은 수정과 하드디스크 박살과, 한번의 랩탑분실과 몇번의 파일시스템 박살과 한번의 모니터 박살의 결과물이 여기있다.

내가 여기에 있는 동안, 사람들의 잘못된 개념을 고쳐주고 싶다. 나는 커널대왕 이 아니다. 나도 내가 커널대왕이 아닌것을 안다. 왜냐하면 이런 커널에 대한 작 업중 진짜 커널대왕들(David S. Miller, Alexey Kuznetsov, Andi Kleen, Alan Cox 같은) 몇명과 접촉해 봤기 때문이다. 그러나 그들은 심오한 마술을 하느라 너무나 바빴고, 내가 안전한 물 가장자리에서 허우적 거리도록 내버려 두었다.

제 5 절 Rusty's 의 패킷 필터링에 대한 총알 가이드

대부분의 사람들은 단 하나의 PPP 접속만 사용하고 어떤 누구도 이것을 통해서 들어오는 것을 원하지 않는다.

```
## connection-tracking modules을 삽입한다. (not needed if built into kernel).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## 내부로부터 오는 것을 제외한 다른 새로운 접속을 막기위하여 새로운 체인을
## 만든다.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## 입력과 포워드 체인으로부터 그 체인으로 가도록 한다.
# iptables -A INPUT -j block
```

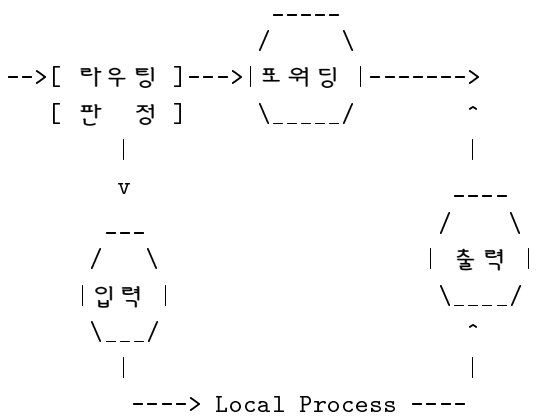
```
# iptables -A FORWARD -j block
```

제 6 절 패킷이 필터를 어떻게 지나는가 ?

커널은 '필터' 테이블에 세개의 규칙을 가지고 시작한다. 이것을 **파이어월 체인** 또는 그냥 **체인**이라고 한다. 그 세개의 체인은 **INPUT**, **OUTPUT**, **FORWARD** 이다.

이것은 2.0 이나 2.2 커널과는 아주 다르게 움직인다.

이 체인은 아래그림처럼 생겼다.



위 그림에서 세개의 원은 위에서 언급한 세개의 체인을 나타낸다. 패킷이 이 그림에서 동그라미로 나타낸 곳에 이르면 그 체인은 그 패킷의 운명을 결정하 기위하여 시험한다. 체인이 그 패킷을 DROP 하라고 하면 패킷은 그곳에서 삭 제된다. 그러나 그 체인이 ACCEPT 하고 하면 이 이 그림의 다음 부분으로 계 속 전달된다.

체인은 **규칙**의 점검표이다. 각 규칙은 '패킷의 헤더가 이렇게 되어있으면 이 곳에서 무엇을 하라'는 형태로 되어 있다. 규칙이 그 패킷에 맞지 않으면 다 음 규칙을 참고한다. 마지막으로 더이상 고려할 규칙이 없으면 커널은 무엇 을 할 것인가를 결정하기 위하여 그 체인의 **정책**을 확인한다. 보안을 생각하 는 시스템에서 이러한 정책은 보통 커널에게 그 패킷을 DROP 하도록 한다.

1. 패킷이 커널에 도탁하면 그 패킷의 목적지를 확인한다. 이것은 '라우팅' 이라고 한다.
2. 이것의 목적지가 이곳이면, 패킷은 위 그림에서 아래쪽 방향으로 전달 되어 입력 체인에 도달한다. 이것이 이 체인을 통과하면 패킷을 기다리 고있던 어떤 프로세서도 그것을 받게 된다.
3. 그렇지 않으면, 커널이 포워딩 불능으로 되어있던가, 패킷을 어떻게 포 워딩해야 하는가를 알지 못하면, 그 패킷은 DROP 된다. 포워딩이 가능하 게 되어있고 다른 곳이 목적지이면 패킷은 그림의 오른쪽 방향으로 전달 되어 포워딩 체인으로 간다. 이 체인이 ACCEPT 하게 되면 이것은 포워딩 할 네트워크로 보내진다.
4. 마지막으로, 이곳에서 돌아가던 프로그램은 네트워크 패킷을 전송할 수 있 게 된다. 이 패킷은 즉시 출력 체인에 보내진다. 이 체인이 ACCEPT 하게 되면 이 패킷은 그 목적지가 어디든지 보내진다.

제 7 절 iptables 사용하기

iptables는 상당히 자세한 메뉴얼 페이지(`man iptables`)를 가지고 있다. ipchains 에 익숙하다면 **??** (ipchains와 iptables의 다른점)을 보아라. 이 둘은 매우 비슷하다.

iptables로 할수 있는 일에는 몇가지 다른것이 있다. 첫번째 작동은 전체 체인을 조절한다. 처음 시작은 세 개의 미리 만들어진 체인으로 시작하는 데 이것은 제거될 수 없다.

1. 새로운 체인 만들기 (-N).
2. 비어있는 체인을 제거하기 (-X).
3. 미리 만들어진 체인의 정책을 바꾸기 (-P)
4. 어떤 체인의 규칙들을 나열하기 (-L)
5. 체인으로부터 규칙들을 지우기 (-F)
6. 체인내의 모든 규칙들의 패킷과 바이트의 카운드를 0 으로 만들기 (-Z)

체인 내부의 규칙을 조작하는 몇가지 방법이 있다.

1. 체인에 새로운 규칙을 추가하기 (-A)
2. 체인의 어떤 지점에 규칙을 삽입하기 (-I)
3. 체인의 어떤 지점의 규칙을 교환하기 (-R)
4. 체인의 어떤 지점의 규칙을 제거하기 (-D)
5. 체인에서 일치하는 첫번째 규칙을 제거하기 (-D)

7.1 컴퓨터가 부팅될때 여러분이 보게 되는 것

iptables는 모듈로 되어있을 것이다. 이것은 iptable_filter.o 이다. 이것은 처음으로 iptables를 실행할때 자동으로 로드될 것이다. 이것느 커널에 영구히 포함될 수도 있다.

iptables 명령이 실행되기 전에는 기본적으로 만들어져있는 체인(입력, 포워딩, 출력)에는 아무른 규칙도 없다. (주의 : 어떤 배포판에는 초기화 스크깁트에 iptables를 실행하는 것이 들어있을 수 있다.) 입력과 출력 체인의 정책은 ACCEPT이고 포워딩 체인의 정책은 DROP이다. (iptable_filter 모듈에 'forward=1' 옵션을 선택하여 이것을 고칠 수 있다.)

7.2 하나의 규칙으로 작동하기

이것은 패킷 필터링의 약방의 감초이다. 일반적으로 추가와 제거 명령을 사용할 것이다. 다른것은 이런 개념의 단순한 확장이다.

각 규칙은 패킷이 일치되어야할 상태를 설정하고, 일치되었을때 무었을 할 것인가 ('target')를 나타낸다. 예를들어, 여러분은 127.0.0.1로부터의 모든 ICMP 패킷을 DROP하려고 할 것이다. 그렇다면, 이경우 일치되어야할 상태는 'ICMP이면서 그 출처 가 127.0.0.1' 이다. 이 경우 'target' 은 DROP 이다.

127.0.0.1 은 'loopback' 인터페이서 이고 실제적인 네트워크 접속이 전혀 없더라도 이것은 가지고 있을 것이다. 이러한 패킷은 'ping' 프로그램을 이용하여 생성할 수 있다. (이것은 단순히 ICMP type 8 (반응요구)을 보내고 모든 협조적인 호스트는 친절하게 ICMP type 0 (반응요구에 대한 응답)을 대답한다. 이것은 테스트하는데 유용 하다.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms
```

```
--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

여기서 첫번째 ping 이 성공한 결과를 볼 수 있다. ('-c 1'은 하나의 패킷만 보내도록 ping에게 말하는 것이다.)

그리고, '입력' 체인에 127.0.0.1에서 오는 패킷('-s 127.0.0.1')으로 ICMP 프로토콜인것 ('-p icmp')은 DROP ('-j DROP')하라는 규칙을 추가(-A)하였다.

그리고는 다시 ping으로 규칙을 테스트하였다. ping이 오지않은 응답을 기다 리지 못하고 응답받기를 포기하기까지 약간의 시간이 걸릴것이다.

규칙을 제거하는데는 두가지 방법이 있다. 첫째, 입력체인에는 단 하나의 규칙 만이 있다는 것을 앎으로, 몇번을 지워라는 방식으로 할 수 있다.

```
# iptables -D INPUT 1
#
```

입력 체인으로부터 1번 규칙을 제거한다.

두번째 방법은 -A 명령을 이용한 이전의 명령에서 -A를 -D로 다꿔주면 된다. 이것은 복잡한 규칙을 가지고 있고 각 규칙이 몇번째 규칙인지를 외우고 다니기를 싫어 한다면, 아주 유용한 방법이다.

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

-D 명령은 -A 명령과 똑 같은 문법이다. (-I 나 -R 도 마찬가지이다.) 만약, 여러개의 똑 같은 규칙들이 같은 체인에 있다면, 첫번째 것만 제거 될 것이다.

7.3 필터링 지정

앞에서 프로토콜을 지정하기위하여 '-p'를 이용하였고, 출처를 지정하기 위하여 '-s'를 이용하였다. 그 외에도 패킷의 특징을 지정하는데 사용되 는 다른 옵션들이 있다. 아래는 이것들에 대한 완벽한 개요이다.

7.3.1 출처와 목적지 지정

출처('-s', '–source', '–src')와 목적지('-d', '–destination', '–dst') IP 주소를 지정하는데 4가지 방법이 있다. 가장 보편적인 방법은 'www.linuxhq.com', 'localhost' 처럼 이름을 이용하는 것이다. 두번째 방법은 '127.0.0.1' 처럼 IP 주소를 이용하는 것이다.

세번째와 네번째 방법은 IP 주소의 그룹을 지정하는 것으로 '199.95.207.0/24' 또는 '199.95.207.0/255.255.255.0' 같은 형태이다. 이 둘은 모두 199.95.207.0 부터 199.95.207.255 사이의 모든 IP 주소를 지정한다. '/' 다음의 숫자는 IP 주소의 어떤 부분이 의미있는가를 나타낸다. '/32' 나 '/255.255.255.255' 가 기본값이다.(IP 주소의 모든부분이 일치해야 한다.) 모든 IP 주소를 지정하는데 '/0' 가 사용된다.

```
# iptables -A INPUT -s 0/0 -j DROP
#
```

이것은 '-s' 옵션을 이용하지 않은것과 같은 효과를 나타내므로 잘 사용되지 않는다.

7.3.2 '역'의 경우 지정

많은 지시자들('-s'나 '-d' 같은)은 일치하지 않는 주소를 나타내기 위하여 '!'('not'을 의미한다)로 시작하는 설정을 할 수 있다. 예로, '-s ! localhost' 는 localhost로부터오는 패킷이 아닌경우를 나타낸다.

7.3.3 프로토콜 지정

프로토콜은 '-p' 지시자로 지정할 수 있다. 프로토콜을 숫자가 될수 있고 (IP의 프로토콜 번호를 알고 있다면) 'TCP', 'UDP', 'ICMP' 같은 이름이 될 수도 있다. 그리고 'tcp'는 'TCP'와 같은 역할을 한다.

프로토콜 이름 지정에도 '!'을 이용할 수 있다. '-p ! TCP'

7.3.4 인터페이서 지정

'-i'('–in-interface')와 '-o'('–out-interface')가 **인터페이서**를 저정 하는데 사용된다. 인터페이서는 패킷이 들어오고 나가는 물리적인 도구이다. `ifconfig` 명령을 사용하여 현재 활성화 되어있는 인터페이서를 알아볼 수 있다.

입력 체인을 지나는 패킷은 출력 인터페이서를 가지고 있지 않으므로 '-o' 설정에 일치하는 패킷이 없을 것이고 **출력** 체인을 지나는 패킷은 입력 인터 페이서를 가지고 있지 않으므로 '-i' 설정에 일치하는 패킷이 없을 것이다.

포워딩 체인을 지나는 패킷만이 입력과 출력 인터페이서를 모두 가질것이다.

현재 존재하지 않는 인터페이서를 지정하는 것도 아무런 문제없이 될 수 있 다. 이것은 인터페이서가 활성화 되기 전까지는 규칙에 일치하는 패킷이 있을수 없을 것이다. 이것은 dial-up PPP를 사용하는 경우 특히 유용하다.

특별한 경우로, 인터페이서 이름이 '+'로 끝날수 있는데 이것은 그 이름으로 시작하는 모든 인터페이서를 모두 지정한다(그것이 현재 존재하든 존재하지 않든). 예를들어, 모든 PPP 인터페이서와 일치하는 규칙을 지정하려면 `-i ppp+`와같이 하면 된다.

인터페이서 이름앞에 '!'도 이용할 수 있다.

7.3.5 분절 (Fragments) 지정

가끔 패킷은 한번에 다 전달되기에는 너무 큰 경우가 있다. 이런경우 패킷은 여러 **분절**로 나뉘어지고 다중 패킷의 형태로 전달된다. 목적지에서 이 분절들 은 재 구성되어 전체 패킷이 된다.

분절에서 문제점은 내부 패킷의 부분으로 IP 헤더 다음의 위치에서 프로토콜 헤더를 찾는데, 이것은 첫번째 분절에만 있기 때문에 찾을수가 없다.

만약 여러분이 접속추적이나 NAT를 한다면 모든 분절은 필터링 코드에 도달하 기 전에 뭉쳐지므로 분절에 대한 걱정은 할 필요가 없다.

그렇지 않다면, 분절들이 필터링 규칙에서 어떻게 처리되는가를 이해하는 것 은 중요하다. 우리가 가지고 있지 않은 정보를 요구하는 필터링 규칙에 부합될 수가 없다. 이것은 첫번째 패킷은 다른 패킷과 같이 처리되고 두번째 이후의 분절은 전달될 수 없음을 의미한다. 그러므로 `-p TCP --sport www` ('www' 를 출신 포트로 지정하는 경우)와 같은 규칙에 맞는 분절은 있을 수 없다(첫번째 분절을 제외하고). 그 반대의 규칙인 `-p TCP --sport ! www`도 분 절들을 처리할 수 없다.

그러나, 두번째 이상의 분절에 대하여 규칙을 지정하기위하여 '-f' ('–fragment')라는 지시자를 사용할 수 있다. 두번째 이상의 분절에는 적용 되지않는 규칙을 지정하기 위하여 '-f' 앞에 '!' 를 붙이는 것도 가능하다.

일반적으로 , 첫번째 분절에 필터링이 적용되어 DROP 되면 목적지 에서 다른 분절들의 재합성이 되지 않으므로, 두번째 이상의 분절이 그냥 지나가도록하는 것도 안전한 것으로 간주 된다. 그러나 단순히 분절들을 전달하는 것만으로 호스트를 크래쉬가 생기는 버그가 알려져 있다. 여러분이 결정할 일이다.

네트워크 헤더에 대한 주의 : 잘못 구성된 패킷들은 이러한 시험을 할때 DROP 되었다. (TCP, UDP, ICMP 패킷중 길이가 너무 짧아 파이어월 코드가 포트나 ICMP 코드와 형태를 읽을 수 없는 경우). 왜냐하면 TCP 분절은 8번째 위치부터 시작되기 때문이다.

예로, 다음과 같은 규칙은 192.168.1.1 로 향하는 분절을 DROP 시킨다.

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

7.3.6 iptables 의 확장 : 새로운 대상(Matches)

iptables는 **확장** 가능하다. 즉 새로운 형태를 제공하기 위하여 iptables와 커널 모두가 확장 가능하다는 의미이다.

이중 일부는 표준적이고 다른 것은 이색적이다. 다른사람에 의해서도 확장 은 만들어질 수도 있으며 독립적으로 배포될 수 있다.

정상적으로 커널 확장은 커널 모듈 하부 디렉토리에 존재한다. (/lib/modules/2.3.15/net). 이것은 요구에 의하여 적재된다. 그러므로 직 접 이들 모듈을 적재할 필요는 없다.

iptables의 확장들은 공유라이버러리 형태로 보통 /usr/local/lib/iptables 에 위치한다. 배포판은 이것을 /lib/iptables나 /usr/lib/iptables에 넣으 려 할 것이다.

확장은 두가지 형태이다. : 새로운 타겟(target), 새로운 적용(match) 아래에 새로운 타겟에 대하여 이야기할 것이다. 어떤 프로토콜은 자동으로 새로운 테스트를 제공하는데 현재로는 TCP, UDP, ICMP 에 대해서 아래에 보여 줄 것이다.

이것을 위해서 '-p' 옵션 뒤에 지정하는데 그러면 확장을 적제할 것이다. 명백히 할려면 '-m' 옵션으로 확장을 적재하고 확장 옵션을 사용가능하게 할 수 있다.

확장에 대한 도움을 얻으려면, 적제하는 옵션('-p', '-j', '-m')을 '-h'나 '–help' 다음에 지정하면 된다.

TCP 확장 TCP 확장은 '–protocol tcp' 가 지정되고 다른 적용이 지정되지 않으면 자동으로 적제된다. 이것은 다음과 같은 옵션을 제공한다.

–tcp-flags

'!' 옵션을 사용한다면 이것 뒤에 두개의 단어를 사용한다. 첫번째 것은 검사하고자 하는 지시자 리스트의 마스크이다. 두번째 단어는 지시자에게 어떤것이 설정 될 것인지를 말해준다. 예를들어,

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DENY
```

이것은 모든것이 검사되어야 함을 말한다.('ALL'은 'SYN,ACK,FIN,RST,URG,PSH' 와 같다.) 그러나 SYN 과 ACK 만 설정된다. 'NONE'는 지시자가 없음 을 말한다.

–syn

'!' 옵션이 선행될 수 있다. 이것은 '–tcp-flags SYN,RST,ACK,SYN'의 약어이다.

–source-port

'!' 옵션이 선행될 수 있다. 이후에 하나의 TCP 포트나 포트의 범위를 지정한다. /etc/services 에 기록된 것과 같은 포트 이름이 사용될 수 도 있고 숫자로 나타낼 수도 있다. 범위는 두개르 포트 이름을

'-' 으 로 연결해서 사용하거나 (커거나 같은경우를 위해서) 하나의 포트 뒤에 '-'를 사용하거나 (작거나 같은 경우를 위해서) 하나의 포트 앞에 '-' 를 덧붙일 수 있다.

–sport

이것은 '–source-port'와 동의어이다.

–destination-port

와

–dport

는 위의 내용과 같으나 목적 지를 지정한다.

–tcp-option

'!' 나 숫자가 옵션에 선행될 수 있는데 숫자가 앞에 올경우 그 숫자 와 TCP 옵션이 같은 경우의 패킷을 검사한다. TCP 옵션을 검사하려 할 때 완전한 TCP 헤더를 갖지않는 것은 자동으로 DROP 된다.

TCP 지시자에대한 설명 가끔 한쪽 방향에서의 TCP 접속만 허랑하고 다른 방향에서의 접속을 불허하는 것이 유용하다. 예로, 여러분은 외부 WWW 서버로의 접속은 허락하며 그 서버로 부터의 접속은 불허하기를 원할 것이다.

단순하게 그 서버로부터 오는 TCP 패킷을 막으면 된다고 생각할 것이다. 그러 나, 불행히도 작동하기 위해서 TCP 접속은 양방향의 패킷을 요구한다.

해법은 접속을 요구하는 패킷만 막는 것이다. 이러한 패킷을 **SYN** 패킷이라한다. (물론, 기술적으로 SYN 지시자 셋을 갖는 패킷이 있다. 그리고 FIN 과 ACK 지시 자는 지워진다. 그러나 간단히 그것을 SYN 패킷이라고 한다.) 이러한 패킷만 불가능으로 만듬으로서 외부로 부터의 접속 시도를 막을 수 있다.

이러한 것을 위해서 '–syn' 지시자가 사용된다. : 이것은 프로토콜을 TCP 로 지정했을 때만 효과가 있다. 예를 들면, 192.168.1.1 로부터의 TCP 접속을 지 정하기 위하여 다음과 같이 하면 된다.

```
-p TCP -s 192.168.1.1 --syn
```

접속을 시작한 것 외의 모든 패킷을 지정하기 위하여 '!' 옵션이 선행될 수 있다.

UDP 확장 이 확장은 '–protocol udp'가 지정되고 적용이 저정되지 않으면 자동으로 적재된다. 이것은 '–source-port', '–sport', '–destination-port', '-dport'를 지원하고 내용은 TCP 설명에서 자세히 나왔다.

ICMP 확장 이 확장은 '–protocol icmp'가 지정되고 그 적용이 지정되지 않으면 자동으로 적재된다. 이것은 단 하나의 새로운 옵션만 지원한다.:

–icmp-type

'!' 옵션이 선행될 수 있다. 이후에 ICMP 타입의 이름('host-unreachable') 이나 숫자형태 ('3'), 또는 숫자형태와 코드('/'로 분리 예. '3/3') 의 형 태가 사용된다. 사용할 수 있는 ICMP 형태의 이름의 리스트는 '-p icmp –help' 하면 나타난다.

그외의 적용 확장 Netfilter 패키지의 다른 확장은 시험적인 확장이다. 이것은 (설치 되어있다면) '-m' 옵션으로 활성화 된다.

mac

이 모듈은 '-m mac' 또는 '–match mac' 이라고 함으로 지정할 수 있다. 이것은 들어오는 패킷의 이더넷 주소를 검사한다. 그러므로 입력 체인에 서만 유용하다. 이것은 하나의 옵션만 제공한다.

–mac-source

'!' 옵션이 선행 될 수 있다. 이후에 콜론으로 분리된 16진수 숫자의 이더넷 주소가 온다. 예 '–mac-source 00:60:08:91:CC:B7'

limit

이 모듈은 '-m limit' 또는 '–match limit'이라고 함으로 지정할 수 있 다. 이것은 로그 메세지를 억제할 때 처럼 적용검사의 속도를 제한하는데 사용한다. 1초에 주어진 숫자만큼의 적용만 검사한다. (기본값은 한 시간 에 3번, 최고 5번이다.) 이것은 두개의 옵션을 제공한다.

–limit

숫자가 따라온다 : 초당 평균 최대 적용 검사 수를 지정한다. 숫자뒤 에 시간단위를 지어할 수 도 있다. ('/second', '/minute', '/hour', '/day'형태이다. 예로, '5/second' 또는 '5/s'가 가능하다)

–limit-burst

숫자가 따라온다. 위의 제한이 적용되기전의 최대 Burst(?) 를 제한 한다.

이 적용은 종종 로그의 속도를 제한하기위하여 LOG 타겟과 함께 사용된 다. 이것을 이해하기위하여 아래에 기본 제한설정을 하는 로그 패킷제한 을 보자.

```
# iptables -A FORWARD -m limit -j LOG
```

이 규칙에 도달될때까지 패킷은 로그될 것이다. 사실 Burst의 기본값은 5 이므로 처므ㅇ 5개의 패킷은 로그될것이다. 그 이후 얼마나 많은 패킷이 도달하든 간에 하나의 패킷이 로그되기전에 20분이 걸릴 것이다. 그리고 20분 동안 패킷 적용이 없으면 Burst 하나가 다시 생길 것이다. 패킷없이 100분 이 지나면 Burst는 완전이 원상 복구 될것이다. 처음 시작할때로 돌아가는 것 이다.

재복구 시간을 59시간 이상으로는 설정하지 못한다. 그러므로 평균속도를 하 루에 1개로 설정하였다면 Burst 속도는 3 이하가 되어야 한다.

owner

이 모듈은 지역에서 생성된 패킷의 생성자의 여러 특징을 적용하려고 한다. 이것은 출력 체인에만 사용되며 어떤 패킷들(ICMP ping 응답같은)은 소유자 가 없으므로 적용되지 않는다.

–uid-owner userid

유효한 사용자 id (숫자)의 프로세서가 생성한 패킷에 적용한다.

–uid-owner groupid

유효한 그룹 id (숫자)의 프로세서가 생성한 패킷에 적용한다.

–pid-owner processid

주어진 프로세서 id 의 프로세서가 생성한 패킷에 적용한다.

–sid-owner processid

세션 그룹내의 프로세서가 생성한 패킷에 적용한다.

unclean

이 시험적인 모듈은 정확히 '-m unclean' 또는 '–match unclean'으로 지정해 주어야 한다. 이것은 무작위의 여러 건전성 검사를 한다. 이것은 제대로 검사되지 않았고 안전성 도구로도 사용되지 못한다.(아마도 이것 은 무제를 더욱 힘들게 하고 버그 그 자체일 것이다.) 이것은 옵션이 없다.

상태 적용 가장 유용한 적용 기준은 'ip_conntrack' 모듈의 접속 추적 분석을 해석하는 'state' 확장이다. 이것을 강력히 추천한다.

'-m state'를 지정함으로 '–state' 옵션을 사용할 수 있는데 이후에 콤마로 분리되는 적용할 상태들의 리스트가 온다.('!' 지시자는 사용되어지지 않는 다.) 이 상태들은 ;

NEW

새로운 접속을 만드는 패킷

ESTABLISHED

존재하는 접속에 속하는 패킷 (즉, 응답 패킷을 가졌던 것)

RELATED

기존의 접속의 부분은 아니지만 연관성을 가진 패킷으로 . ICMP 에러 나 (FTP 모듈이 삽입 되어있으면) ftp 데이터 접속을 형성하는 패킷.

INVALID

어떤 이유로 확인할 수 없는 패킷: 알려진 접속과 부합하지 않는 ICMP 에러와 'out of memory' 등을 포함한다. 보통 이런 패킷은 DROP 된다.

7.4 타겟 지정

이제 패킷에서 어떤 검사를 할 수 있는지를 알았다. 이제 우리의 검사에 일치 하는 패킷을 어떻게 할 것인지를 말하는 것을 알아야 한다. 이것을 규칙 **타겟** 이라고 한다.

두개의 이미 만들어진 단순한 타겟이 있다. : DROP 과 ACCEPT. 이미 이것에 대해서는 이야기를 한 적이 있다. 적용이 되는 패킷과 그것의 타겟이 위의 두 개중 하나라면 더이상의 참고할 규칙은 없다. : 패킷의 운명은 결정 되는 것 이다.

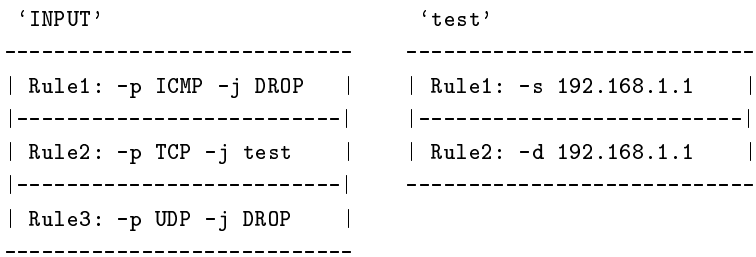
이미 만들어진 두개의 타겟외에 두가지 형태의 타겟이 있다.: 확장과 사용자 지정의 체인들 이다.

7.4.1 사용자 지정의 체인들

ipchains로 부터 상속되는 iptables의 강력한 기능중의 하나는 능력되는 사용 자가 기존의 세개의 체인(입력, 출력, 포워드)외에 새로운 체인을 생성할 수 있다는 것이다. 모임의 결과 사용자 지정의 체인은 그것을 구분하기 위하여 소문 자로 나타낸다. (아래 7.5 (전체 체인에 대한 작용) 부분에서 어떻게 사용자 지정의 새로운 체인을 만드는지 기술할 것이다.)

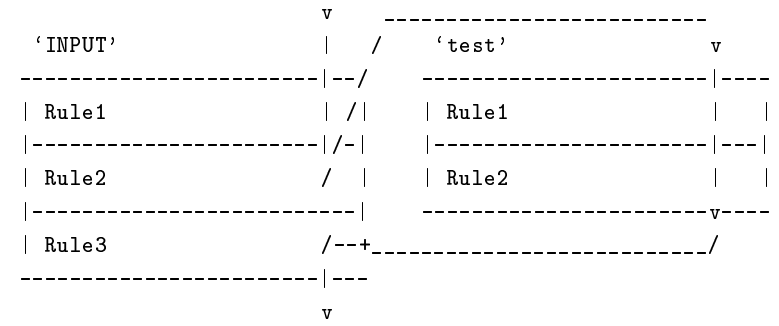
타겟이 사용자 지정의 체인인 규칙에 패킷이 맞으면 패킷은 사용자 지정의 체인을 따라 움직이게 된다. 그 체인이 패킷의 운명을 결정하지 못하면 그리고 그 체인에 따른 이송이 끝나면, 패킷은 현제 체인의 다음 규칙으로 돌아온다.

그림을 보자. 두개의 체인이 있고 그것이 입력과 테스트 라는 사용자 지정의 체인이 라고 가정하자.



192.168.1.1 로부터 와서 1.2.3.4 로 향하는 TCP 패킷이 있다고 가정한다. 이것은 입력 체인으로 들어온다. Rule1 을 검사한다. 맞지 않음. Rule2 맞음. 그것의 타겟 은 테스트, 고로 다음 검사할 규칙은 테스트의 시작이다. 테스트의 Rule1 이 맞다. 그러나 이것이 타겟을 지정하지 않는다. 그러므로 다음 규칙이 검사된다. Rule 2. 맞지 않다. 그 체인의 끝에 도달했다. 다시 입력 체인으로 돌아가서 Rule3 을 검사 한다. 그것도 맞지 않다.

여기서 패킷의 이동경로를 그림으로 나타냈다.



사용자 지정의 체인에서 대를 사용자 지정의 체인으로 갈수 있다. (그러나 루프 를 돌수는 없다. 루프를 발견하게 되면 패킷은 DROP 된다.)

7.4.2 iptables로의 확장 : 새로운 타겟

타겟의 다른 형태는 확장이다. 타겟 확장은 커널 모듈로 구성된다. 그리고 `iptables` 에 대한 선택적 확장은 새로운 명령행의 옵션을 제공한다. 기본적으로 넷필터 배포에 포함된 몇몇의 확장은 다음과 같다.

LOG

일치하는 패킷의 커널 로그를 제공한다. 이것은 부가의 옵션을 제공한다.

–log-level

레벨 숫자나 이름이 따라온다. 유효한 이름은 (상황에 따라 다르다) 'debug' 'info', 'notice', 'warning', 'err', 'crit', 'alert', 'emerg' 이고 이것 은 각각 숫자 7 에서 0 에 대응한다. 이런 레벨에 대한 설명은 syslog.conf 의 man 페이지를 보라.

–log-prefix

14자 까지의 문장이 따라온다. 이 메세지는 로그 메세지의 시작부분으로 보내져서 확인에 사용될 수 있다.

이 모듈은 'limit' 타겟 다음에 사용하면 가장 효과적이다. 그래서 로그가 넘 지나지 않도록 할 수 있다.

REJECT

이 모듈은 'DROP'과 같은 효과를 나타낸다. 다만, 'port unreachable' 이라는 에러 메세지를 ICMP 로 보낸다. 주의할 것은 ICMP 에러 메세지는 다음의 경우 보내 지지 않는다 (RFC 1122 를 보라) :

- 검사된 패킷이 ICMP 에러메세지이거나나 알수 없는 ICMP 형태인 경우
- 검사된 패킷이 헤더가 없는 분절인 경우
- 너무 많은 ICMP 에러 메세지를 그 목적지로 보낸 경우.

REJECT 는 '–reject-with'라는 옵션을 가지는데 이것은 사용할 응답 패킷을 변경한다. 자세한 것은 메뉴얼 페이지를 보라.

7.4.3 특별한 미리 만들어진 타겟

두개의 미리 만들어진 타겟이 있다 : RETURN, QUEUE

RETURN은 한 체인의 끝으로 보내지는 것과 같은 효과가 있다. : 미리 만들어진 체 인의 경우 그 체인의 정책은 실행이다. 사용자 정의 체인의 경우 이 체인으로 점프 하는 규칙의 바로 다음인 이전 체인으로 이동한다.

QUEUE은 특별한 타겟으로, 사용자공간의 작업을 위해 패킷을 대기하도록 한다. 패킷 을 위해서 대기하고있는 것이 없다면(즉, 이 패킷을 다룰 프로그램이 아직 씌어져 있지 않다면) 패킷은 DROP 될 것이다.

7.5 전체 체인에 대한 작용.

iptables의 유용한 기능주 하나는 여러 관계가 있는 규칙을 하나의 체인속으로 그룹화 하는 것이다. 체인의 이름은 어떤 것을 사용할 수도 있으나 미리 만들어 진 체인과의 혼동을 막기 위하여 소문자를 사용하기를 권한다. 체인의 이름은 16 자 까지 가능하다.

7.5.1 새오룬 체인 생성

새로운 체인을 만들어 보자. 나는 매우 상상력이 좋은 사람이므로 이것을 테스트 라고 부르기로 하겠다. '-N' 또는 '–new-chain' 옵션을 사용한다.

```
# iptables -N test
#
```

단순하다. 이제 이 체인에 상세한 규칙을 적용할 수 있다.

7.5.2 체인 제거

체인을 제거하는 것도 단순하ㄷ. '-X' 나 '–delete-chain' 을 사용한다.

```
# iptables -X test
#
```

체인을 제거 하는데는 몇가지 제한이 있다. 이것은 비어있어야 한다. (아래의 7.5.3 (체인 비우기)를 보라) 그리고 그것은 다른 어떤 규칙의 타겟도 아니어야 한다. 미리 만들어진 세개의 체인은 제거할 수 없다.

체인의 이름을 지정하지 않으면 모든 사용자 정의의 체인은 제거된다.

7.5.3 체인 비우기

하나의 체인의 모든 규칙을 비우는 간단한 방법이 있으니, '-F' ('–flush') 명령이다.

```
# iptables -F forward
#
```

체인을 지정하지 않으면 모든 체인의 규칙이 지워진다.

7.5.4 체인 규칙 나열하기

한 체인의 모든 규칙은 '-L' 명령으로 나열할 수 있다.

각 사용자 정의의 체인을 나열했던 'refcnt' 는 그 체인을 그들의 타겟으로하 는 규칙들의 번호이다. 체인이 제거되기 위해서는 이것이 '0' 으로 되어야 한다. (그리고 그 체인은 비어야 한다)

체인의 이름이 생략되면 비어있는 것을 포함한 모든 체인이 나열된다.

'-L' 명령에 따르는 옵션은 세개가 있다. '-n' (numeric) 옵션은 `iptables`가 여러분이 DNS 요구를 필터링 아웃한 경우나 DNS가 적절이 설정되어 있지 않다면 오랜 시간이 걸리는, IP 주소를 찾는 것을 예방하는 아주 유용한 옵션이다. 이것 은 TCP와 UDP 포트가 이름이 아닌 숫자로 출력되도록 하기도 한다.

'-v' 옵션은 규칙의 자세한 정보(패킷과 바이트 카운터, TOS 비교, 인터페이서와 같은)를 나타낸다.

패킷과 바이트 카운트는 'K'(1000), 'M'(1,000,000), 'G'(1,000,000,000) 와 같은 접미어와 함께 나타난다. '-x' (확장 수) 지시자를 사용하면 얼마나 큰 숫자든 전 체 숫자가 나타난다.

7.5.5 카운트 리셋트 ('0'으로 만들기)

이것은 카운트를 리셋하는데 유용하다. 이것은 '-Z' ('–zero') 옵션으로 가능하다.

이것의 문제점은 리셋하기 직전의 카운트 값을 알필요가 있을 때가 가끔 있다는 것 이다. 이러한 경우의 예로, 어떤 패킷이 '-L' 과 '-Z' 명령 사이에 지나갈 수 있다. 이런이유로 카운트를 읽는 것과 동시에 리셋하기 위해서 '-L' 과 '-Z' 명령을 같이 사용할 수 있다.

7.5.6 정책 설정

우리가 이전 체인을 패킷이 어떻게 지나가는가를 의논할 때, 미리 만들어진 체인의 끝에 패킷이 다다렀을 때 무슨 일이 일어날 것인가를 설명하였다. 이 경우 체인의 **정책**이 그 패킷의 운명을 결정한다. 미리 만들어진 체인(**입력**, **출력**, **포워드**)만이 정책을 가지는데, 이것은 사용자 정의의 체인의 끝에 다다른 패킷의 이동은 이전 체인에서 요약되어지기 때문이다.

정책은 `ACCEPT` 또는 `DROP` 이 될수 있다.

7.5.7 ipchains와 ipfwadm을 사용하기

배포되는 넷필터에는 ipchains.o 와 ipfwadm.o 라는 모듈이 있다. 이것을 여러 분의 커널에 포함시키면 이전과 똑 같이 ipchains 나 ipfwadm을 사용할 수 있 다. (주의 : 이들은 iptables.o, ip_conntrack.o, ip_nat.o와 호환성이 없다)

이것은 아직 한동안은 지원될 것이다. 이들을 완전히 대치하는 안정판이 나오 는데 까지는 2 * [대치할 것이라는 발표 - 첫번째 안정판] 이라는 공식이 적 용된다고 생각한다.

즉, ipfwadm의 경위 이것의 지원이 종료될 때는 :

```
2 * [October 1997 (2.1.102 release) - March 1995 (ipfwadm 1.0)]
        + January 1999 (2.2.0 release)
    = November 2003.
```

그리고 ipchains의 경우 이것의 지원이 종료될 때는 :

```
2 * [August 1999 (2.3.15 release) - October 1997 (2.2.0 release)]
        + January 2000 (2.3.0 release?)
    = September 2003.
```

그러므로, 2004년 까지는 걱정할 필요가 없을 것이다.

제 8 절 iptables와 ipchains의 차이점

- 첫째로, 미리 만들어진 체인의 이름들이 소문자에서 대문자로 바뀌었는데 이것은 입력과 출력 체인은 이게 지역 넷으로 향하는 그리고 지역에서 생성된 패킷만을 적용하기 때문이다. 이것은 모든 들어오는것과 나가는 패킷을 다룬다.
- '-i' 지시자는 들어오는 인터페이스만 의미하고 입력 과 포워드 체인 에서만 작동한다. 포워드 나 출력 체인에 사용되었던 '-i' 는 '-o'로 바꿔야 한다.
- 이제 TCP 와 UDP 포트는 –source-port 또는 –spotr (또는 –destination-port / –dport) 옵션과 함께 사용되어져야 할 필요가 있고 '-p tcp' 또는 '-p udp' 옵션과 함께 사용되어져야 한다. 그러면 이것은 TCP 또는 UDP 확장을 각각 적재 할 것이다. (ipt_tcp 와 ipt_udp 모듈을 수동으로 적제 하기위해서 포함 시킬 수도 있다.)
- TCP -y 지시자는 –syn으로 바뀌었고 '-p tcp'다음에 와야 한다.
- DENY target 는 DROP 으로 바뀌었다.
- Zeroing single chains while listing them works.
- 만들어진 체인을 '0'으로하면 정책 카운터도 지워진다.
- 체인을 나열하는 것은 카운트의 스넵샷을 제공한다.
- REJECT 와 LOG 는 확장된 target이다. 즉, 이것들은 독립된 커널 모듈이다는 의미이다.
- 체인 이름은 16자 까지 가능하다.
- MASQ 와 REDIRECT 는 더이상 target 이 아니다. iptables은 패킷을 변화 시키지 않는다. 이것을 위해서 NAT라는 하부구조가 있다. 이것은 ipnatctl 하우두를 읽어보아라.
- 그 외의 것들은 잊어먹었다.