Как да направим пакетен филтър с Линукс 2.4

Ръсти Ръсел, пощенски списък netfilter@lists.samba.org
превод Радослав Колев, rado_k@yahoo.com
$Версия: 1.26 $ $Дата: 2002/01/24 13:42:53$ превод $Версия: 0.01 $ $Дата: 2004/04/14 23:52:53$

Този документ описва как се използват iptables за филтриране на пакети в Линукс 2.4 ядрата.

Съдържание

1. Въведение

Добре дошъл, скъпи читателю!

Предполага се, че знаете какво е IP адрес, мрежов адрес, мрежова маска, рутиране и DNS. Ако не е така, препоръчвам ви да прочетете Network Concepts HOWTO.

Стилът в който е написано това ръководство се мени между кратки обяснения (които ще ви накарат да се чуствате спокойни, но на практика незащитени) и директно и пълно разглеждане на нещата (което ще накара всички, освен най-смелите да се чуствят объркани, параноични, търсещи тежко въоръжение).

Вашата мрежа не е **сигурна**. Задачата да се осигурят бързи и удобни средства за комуникация, чиято употреба да се ограничи само за добри, а не злонамерени цели е подобна на други трудно разрешими проблеми, като например да се осигури свобода на словото, като в същото време се забрани извикването на "Пожар!"в препълнена зала. Тези проблеми няма да бъдат разрешени в това ръководство.

Единствено Вие можете да решите какъв ще бъде компромисът. Аз ще се опитам да ви дам инструкции за това как да използвате някои от наличните инструменти и ще ви покажа някои слаби места за които трябва да знаете. Надявам се, че ще ги използвате за добро, а не за лошо. Това е друг подобен проблем.

2. Къде е официалният уеб сайт? Има ли пощенски списък?

Има три официални сайта:

- Благодарение на *Filewatcher* http://netfilter.filewatcher.org/ .
- Благодарение на *The Samba Team and SGI* http://netfilter.samba.org/ .
- Благодарение на *Harald Welte* http://netfilter.gnumonks.org/ .

Можете да достигнете до всеки един от тях чрез кръгови DNS заявки на адрес

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

Относно официалният пощенски списък вижте

netfilter List http://www.netfilter.org/contact.html#list .

3. И така, какво е пакетен филтър?

Пакетният филтър е програма, която преглежда *заглавните части (header)* на преминаващите пакети, и решава съдбата на целия пакет. Той може да реши да **отхвърли (DROP)** пакета (с други думи, да се държи така, все едно този пакет никога не е пристигал), **да приеме (ACCEPT)** пакета (т.е, да разреши на пакета да премине), или да извърши друго по-сложно действие.

При Линукс кодът за филтриране на пакети е част от ядрото (като модул, или компилиран директно в ядрото) и можем да правим някои доста сложни неща с пакетите, но основният принцип на предглеждане на заглавната част и решаване на съдбата на целия пакет е все още валиден.

3.1. Защо бих искал да имам пакетен филтър?

Контрол. Сигурност. Наблюдение.

Контрол:

когато използвате компютър с Линукс за да свържете вашата вътрешна мрежа с някоя друга (например Интернет) вие имате възможност да разрешите само някои конкретни типове трафик, както и да забраните други. Например в заглавната част на пакета се съдържа адреса към който е насочен този пакет (destination), така че можете да спрете всички пакети насочени към определена част от външната мрежа. Ето и друг пример - аз използвам Netscape за да разглеждам архивите Dilbert. На страниците там има реклами от doubleclick.net и Netscape губи моето време и честотна лента като радостно ги зарежда. Казвайки на пакетния филтър да не приема пакети от или за адресите притежавани от doubleclick.net може да решим този проблем (има по-добри начини да се справите с това, виж Junkbuster).

Сигурност:

когато вашият компютър с Линукс е единственото нещо между хаосът в Интернет и вашата хубава и подредена мрежа, то добре е да знаете как да ограничите какво може да почука на вашата врата. Например, може да искате да разрешите всичко което излиза навън от вашата мрежа, но се притеснявате от добре известния ‘Ping на смъртта’ идващ от злонамерени външни хора. Или пък не искате някои отвън да има телнет достъп до вашата Линукс машина, въпреки че всички потребители имат пароли. Може би (както повечето хора) вие искате да бъдете само наблюдател в Интернет, а не и сървър (със или без ваше знание). Просто не позволявайте на някой отвън да се свързва с мрежата ви, като накарате вашият пакетен филтър да отказва всички входящи пакети които се опитват да инициират връзка.

Наблюдение:

понякога лошо конфигурирана машина от локалната мрежа ще реши да изплюе пакети към външния свят. Добре е да кажете на пакетния филтър да ви уведоми в случай, че се случи нещо странно; може би ще можете да направите нещо по въпроса, или пък просто сте си любопитни по природа.

3.2. Как да направя пакетен филтър под Линукс?

Линукс ядрата имат възможности за филтриране на пакети още от версиите 1.1. Първото поколение, базирано на ipfw от BSD, беше пренесено от Алан Кокс в края на 1994 година. То беше подобрено от Йос Вос (Jos Vos) и други хора във версията на Линукс 2.0; потребителската програма ‘ipfwadm’ контролираше правилата за филтриране в ядрото. В средата на 1998 година, за Линукс 2.2 аз преработих голяма част от кода в ядрото с помощта на Майкъл Нюлинг и създадох потребителската програма ‘ipchains’. И накрая, друго пренаписване на кода в ядрото и потребителската програма от четвърто пололение ‘iptables’ станаха факт в средата на 1999 година за Линукс 2.4. По-голямата част от това ръководство е концентрирана върху iptables.

Трябва ви ядро с netfilter инфраструктура в него; netfilter е общата основа в Линукс ядрото към която се включват останалите компоненти (като iptables модула например). Това означава, че ви трябва ядро версия 2.3.15 или по-нова. Отговорете с ‘Y’ на CONFIG_NETFILTER при конфигурацията на ядрото.

Програмата `iptables` си говори с ядрото и му казва кои пакети да филтрира. Освен ако не сте програмист, или пък извънаредно любопитен, то това ще е начин по който ще контролирате филтрирането на пакети.

3.2.1. iptables

Програмата `iptables` вмъква и изтрива правила от таблицата за пакетно филтриране в ядрото. Това означава, че всички настройки които направите ще бъдат загубени след рестарт; виж 3.2.2 (Запазване на правилата след рестарт) за обяснение как да се осигури възстановяването на настройките при следващото зареждане на Линукс.

`iptables` заменя програмите `ipfwadm` и `ipchains`: виж 8 (Използване на ipchains и ipfwadm) за това как безболезнено на избегнете употребета на iptables ако вече ползвате някои от другите инструменти.

3.2.2. Запазване на правилата след рестарт

Текущата конфигурация на вашата защитна стена (firewall) се пази в ядрото, следователно тя ще се загуби при рестарт. Може да ползвате скриптовете iptables-save и iptables-restore за запазване на конфигурацията във файл и нейното възстановяване.

Друг начин е да сложите необходимите команди за задаване на желаните правила в инициализиращите скриптове. Уверете се, че ще направите нещо интелигентно, в случай че изпълнението на някоя от командите се провали (обикновено ‘exec /sbin/sulogin’).

4. Кой по дяволите си ти, и защо бърникаш из моето ядро?

Аз съм Ръсти Ръсел; човекът който поддържа кода за филтриране на IP пакети в Линукс ядрото, или просто още един програмист, който се оказа на подходящото място в подходящото време. Аз написах ipchains (виж 3.2 (Как да направя пакетен филтър под Линукс?) по-горе за съответните заслуги на хората които свършиха същинската работа), и научих достатъчно за да мога този път да направя филтрирането на пакети по правилния начин. Надявам се.

WatchGuard `http://www.watchguard.com` , една отлична компания произвеждаща защитни стени и продаваща Firebox - един много добър продукт, ми предложи да ми плаща за да не правя нищо, така че да концентрирам цялото си внимание вътху разработката на тези неща тук и върху поддръжката на предишни такива. Аз предвидих че ще са ми необходими 6 месеца, а ми трябваха 12, но накрая чуствах, че нещата са направени както трябва. Много пренаписвания, скапан хард диск, откраднат лаптоп, няколко повредени файлови системи и един счупен екран по-късно, ето го и резултата.

И докато съм още с вас, бих искал да променя грешните представи на някои хора: аз не съм гуру разработчик на ядрото. Зная това, защото по време на работата ми върху ядрото имах възможността да контактувам с някои от тях като: Дейвид С. Милър, Алексей Кузнецов, Анди Клийн, Алан Кокс. Всички те са много заети с правенето на своята магия, оставяйки ме да газя из плитките води, където е безопасно.

5. Много бърз урок по филтриране на пакети, а-ла Ръсти Ръсел

Повечето хора имат само една PPP връзка към Интернет, и искат никой да няма достъп до тяхната вътрешна мрежа или защитна стена:

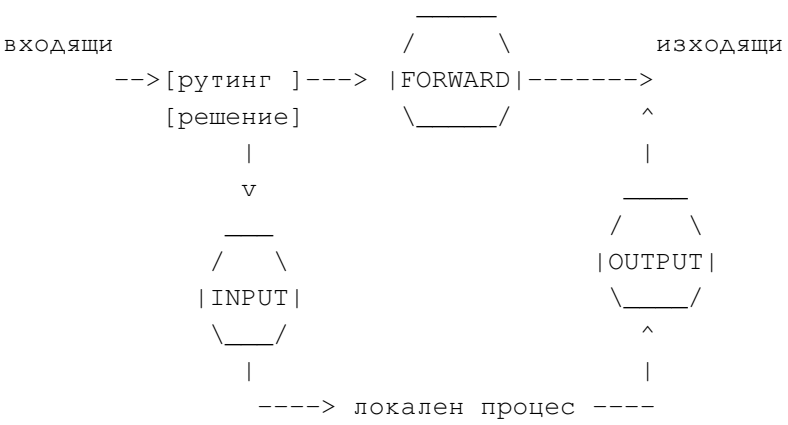
```
## Вмъква connection-tracking модулите (не е необходимо ако са вградени в ядрото).
# insmod ip_conntrack
```

```
# insmod ip_conntrack_ftp

## Създава верига, която блокира всички нови връзки, освен тези идващи отвътре
# 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

## Пренасочване към тази верига от INPUT и  FORWARD веригите.
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

6. Как пакетите преминават през филтрите

Първоначално в ядрото има три списъка от правила във 'filter' таблицата; тези списъци се наричат **вериги с правила за филтриране (firewall chains)** или просто **вериги (chains)**. Тези три вериги се наричат **INPUT (вход)**, **OUTPUT (изход)** и **FORWARD (препращане)**.

За феновете на ASCII арта, веригите са подредени ето така: **(Забележка: подреждането е доста по различно от това при ядра весрии 2.0 и 2.2 !)**



Трите кръга представляват веригите за които се спомена по-горе. Когато един пакет достигне до някой от кръговете на диаграмата се проверяват правилата в тази верига и се решава съдбата на пакета. Ако веригата каже пакета да бъде отхвърлен (DROP), то той бива заличен веднага, но ако решението е той да бъде приет (ACCEPT) пакета продължава пътя си по диаграмата.

Веригата е списък от **правила**. Всяко правило гласи 'ако заглавната част на пакета изглежда така, то направи с този пакет ето това'. Ако пакетът не отговаря на изискванията в това правило, то се преминава към следващото във веригата. И накрая, ако няма повече правила с които да се сравни пакета ядрото решава съдбата му съгласно **политиката (policy)** на веригата. Обикновено, ако в една система се държи на сигурността, то политиката казва на ядрото да откаже пакета.

1. Когато се получи пакет (примерно от мрежовата карта) ядрото първо поглежда за къде е адресиран той: това се нарича 'рутиране'.
2. Ако е адресиран за тази машина, то пакетът преминава надолу по диаграмата към INPUT веригата. Ако премине през нея процеса чакащ този пакет ще го получи.
3. В противен случай, ако в ядрото не е активирано прехвърлянето на пакети (forwarding), или пък то не знае в каква посока да го пренасочи, пакета ще бъде отказан. Ако пък прехвърлянето е включено и паке-

тът е насочен към друг мрежов интерфейс (ако имате такъв), тогава той отива надясно в диаграмата, към FORWARD веригата. Ако премине успешно през нея ще бъде изпратен по другия интерфейс.

4. И накрая, програма работеща на машината може да изпраща пакети. Те минават директно към OUTPUT веригата: ако тя каже, че пакета може да премине той продължава към който интерфейс е бил насочен.

7. Използване на iptables

iptables има доста детайлна man страница (`man iptables`), в случай, че ви трябват повече конкретни детайли. Тези от вас които са използвали ipchains може просто да погледнат 10 (Разлики между iptables и ipchains); те са много подобни.

Има няколко различни неща които можете да правите с `iptables`. Започвате с трире вградени вериги `INPUT`, `OUTPUT` и `FORWARD` които не можете да изтриете. Нека разгледаме действията чрез които можете да контролирате цели вериги:

1. Създаване на нова верига (-N).
2. Изтриване на празна верига (-X).
3. Смяна на политиката на вградена верига (-P).
4. Показване на правилата във верига (-L).
5. Изтриване на правилата във верига (-F).
6. Нулиране на байтовите и пакетни броячи на всички правила във верига (-Z).

Има няколко начина за манипулиране на правилата в една верига:

1. Добавяне на ново правило към верига (-A).
2. Вмъкване на ново правило на дадена позиция (-I).
3. Замяна на едно правило на дадена позиция с друго (-R).
4. Изтриване на правило на дадена позиция, или първото съвпадащо (-D).

7.1. Какво ще видите, когато компютърът ви стартира

iptables може да бъде модул, с име (‘iptable_filter.o’), който трябва да бъде зареден автоматично когато стартирате `iptables` за първи път. Може също така да бъде вграден в ядрото.

Преди да бъдат изпълнени съответните iptables команди (внимание: някои дистрибуции стартират iptables в своите стартиращи скриптове), няма да има никакви правила в която и да е от вградените вериги (‘INPUT’, ‘FORWARD’ и ‘OUTPUT’), всички вериги ще имат политика ACCEPT. Можете да промените политиката по подразбиране за FORWARD веригата като укажете опцията ‘forward=0’ на iptable_filter модула.

7.2. Действия върху едно правило

Това са най-често ползваните действия при филтрирането на пакети - манипулирането на правила. Вероятно най-популярните от тях са командите за добавяне (-A) и изтриване (-D). Другите две (-I за вмъкване и -R за замяна) са просто техни разширения .

Всяко правило задава определени условия на които пакета трябва да отговаря, и какво действие да се извърши с него в такъв случай (‘target’). Например, вие може да искате да забраните всички ICMP пакети идващи от 127.0.0.1. В такъв случай условията които искате да зададете са: протоколът да бъде ICMP; и адресът източник да е 127.0.0.1. Действието трябва което да се извърши е ‘DROP’.

127.0.0.1 е адреса на ‘loopback’ интерфейса, който ще имате дори и без връзка към истинска мрежа. Може да използвате програмата ‘ping’ за да генерирате такива пакети (тя просто изпраща ICMP тип 8 (echo request) пакети, на които всички мрежови устройства (или поне тези които имат доброто желание) трябва задължително да отговорят с ICMP тип 0 (echo reply) пакет). Това я прави много полезена за тестове.

```
# 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
#
```

Можете да видите, че първият ping е успешен (опцията ‘-c 1’ казва да се изпрати само един пакет).

Тогава ние добавяме (-A) към ‘INPUT’ веригата правило, което указва всички пакети идващи от 127.0.0.1 (‘-s 127.0.0.1’) с протокол ICMP (‘-p icmp’) да бъдат отказани (‘-j DROP’).

След това пробваме нашето правило, като пускаме ping отново. Този път, след кратка пауза, програмата се отказва да чака отговор който никога няма да получи.

Можем да изтрием правилото по един от двата начина. Знаем, че има само едно правило в input веригата, така че можем да го изтрием по номер:

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

Изтрива правило номер 1 от INPUT веригата.

Вторият начин е да използваме същата команда, както при добавянето на правилото, но да заменим -A с -D. Той е полезен когато имата сложна верига от правила и не искате да ги броите за да разберете кое е правило номер 37, което всъщност искате да премахнете. В такъв случай може да ползвате:

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

Синтаксисът на -D трябва да бъде същият както при -A (или -I и -R) командата. Ако има повече от едно съвпадащи правила в същата верига, то ще бъде изтрито само първото.

7.3. Дефиниране на филтри

Видяхме как се използва опцията ‘-p’ за да се укаже определен протокол, или пък ‘-s’ за адреса на изпращача. Освен тези две има много други опции, които можем да използваме за да зададем определени характеритики на пакетите. Следват кратки описания на всяка една от тях.

7.3.1. Задаване на IP адрес на изпащача и получателя

IP адресите на изпращача (‘-s’, ‘–source’ или ‘–src’) и получателя (‘-d’, ‘–destination’ или ‘–dst’) могат да бъдат зададени по четири различни начина. Най-често се използва пълното име, например ‘localhost’ или ‘www.linuxhq.com’. Вторият начин е да се укаже IP адрес като ‘127.0.0.1’.

Третият и четвъртият начини позволяват задаването на група от IP адреси като ‘199.95.207.0/24’ или ‘199.95.207.0/255.255.255.0’. И двата примера означават всички IP от 199.95.207.0 до 199.95.207.255 включително; числото след знака ‘/’ определя кои части от IP се вземат под внимание. Стойността по подразбиране е ‘/32’ или ‘/255.255.255.255’ (сравнява целия IP адрес). За да зададете всеки IP адрес може да използвате ‘/0’:

```
[ Забележка: ‘-s 0/0’ е излишно в примера по долу. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Това се ползва рядко, защото ефекта е същият както ако не е зададена опцията ‘-s’.

7.3.2. Инверсия

Пред аргументите на много от флаговете, включително ‘-s’ (или ‘–source’) и ‘-d’ (или ‘–destination’) можете да поставите ‘!’ (чете се като ‘НЕ’) за да укажете всички адреси, които НЕ са равни на дадените. Например ‘-s ! localhost’ ще съвпадне с всеки пакет, който **не** идва от localhost.

7.3.3. Задване на протокол

Протоколът се указва с опцията ‘-p’ (или ‘–protocol’). Протоколът може да бъде число (ако знаете числените стойности отговарящи на съответните протоколи в IP) или име в случаите на ‘TCP’, ‘UDP’ или ‘ICMP’. Няма значение дали буквите са малки или главни, така че ‘tcp’ е еквивалентно с ‘TCP’.

Пред името на протокола може да се постави ‘!’ за да се инвертира, например ‘-p ! TCP’ за да зададете всички пакети с протокол **различен** от TCP.

7.3.4. Задаване на интерфейс

Опциите ‘-i’ (или ‘–in-interface’) и ‘-o’ (or ‘–out-interface’) указват името на входящия или изходящия **интерфейс**. Интерфейс се нарича физическото устройство по което пакетът е пристигнал (‘-i’) или пък предстои да бъде изпратен (‘-o’). Можете да използвате командата `ifconfig` за да видите списък на всички интерфейси които са ‘вдигнати’ (т.е., работещи в момента).

Пакетите преминаващи през `INPUT` веригата нямат изходящ интерфейс, следователно правила в които е указан такъв с опцията ‘-o’ няма да съвпаднат с никой от преминаващите пакети.

Аналогично, пакетите преминаващи през `OUTPUT` веригата нямат входящ интерфейс, следователно правилата използващи опцията ‘-i’ в тази верига няма да съвпаднат с никой пакет.

Само пакетите преминаващи през `FORWARD` веригата имат едновременно и входящ и изходящ инрерфейс.

Напълно допустимо е да зададете интерфейс който в момента не съществува; правилото няма да съвпадне с никой от пакет, докато не се вдигне съответния интерфейс. Това е особенно полезно за dial-up PPP връзки (обикновено интерфейс `ppp0`) и дурги подобни случаи.

Като специален случай, интерфейси с име завършващо със знак ‘+’ ще съвпаднат с всички интерфейси (независимо дали в момента съществуват или не) чието име започва с този символен низ. Например, за да зададете правило което съвпада с всички PPP интерфейси трябва да използвате опцията `-i ppp+`.

Ако пред името на интерфейса се постави знака ‘!’ с интервали около него, то правилото ще съвпада само с пакети които **не** от посоченият интерфейс(и),например `-i ! ppp+`.

7.3.5. Задаване на фрагменти

Понякога един пакет е прекалено голям за да може да бъде изпратен целият навендъж. Когато това се случи този пакет се разделя на **фрагменти**, които се изпращат като няколко по-малки отделни пакета. Получателят

съединява тези фрагменти за да получи първоначалният пакет.

Проблемът с фрагментите е, че първият от тях има пълен набор от заглавни полета (IP + TCP, UDP и ICMP), но следващите пакети имат само част от тях (IP без полета за другите протоколи). Следователно да се преглежда съдържанието на втория и следващи фрагменти за полетата на съответните протоколи (както това правят TCP, UDP и ICMP разширенията) е невъзможно.

Ако използвате функционалността за следене на връзките (connection tracking) или NAT, тогава всички фрагменти ще бъдат съединени преди да достигнат до кода за филтриране. В такъв случай няма нужда да се притеснявате за фрагментите.

Моля да забележите, че правилата в INPUT веригата от таблицата за филтриране (или всяка друга таблица закачаща се към NF_IP_LOCAL_IN hook) се преглеждат след като пакетите са дефрагментирани от IP стека на ядрото.

Във всички други случаи е важно да разбирате как фрагментите се третират от правилата за филтриране. Всяко правило, което изисква информация която не може да бъде предоставена *няма* да съвпадне с пакета. Това означава, че първия фрагмент се третира като всеки друг пакет. За вторият и следващи фрагменти, това не е така. Например правилото `-p TCP -sport www` (задаващо изходен порт на 'www') няма никога да съвпадне с фрагмент (с изключение на първия). Същото важи и за инверсното правило `-p TCP -sport ! www`.

Можете да зададете правило което се отнася специално за втория и следващи фрагменти като използвате флага '-f' (или '–fragment'). Също така е допустимо да укажете, че определено правило *не* се отнася за втория и следващи фрагменти като поставите ' ! ' пред '-f'.

Обикновено се счита за безопасно да позволите на втория и следващите фрагменти да преминават, защото ако първия е бил филтриран, то това ще направи невъзможно сглобяването на пакета от получателя; въпреки това са известни грешки в някои софтуерни продукти които позволяват забиване на цялата машина чрез просто изпращане на фрагмент. Ваш избор.

Забележка за мрежовите специалисти: малформирани пакети (TCP, UDP или ICMP пакети прекалено малки за да бъдат прочетени портовете или ICMP кодът и тип) се отхвърлят при опит да бъдат прегледани. Същото важи и за TCP фрагменти започващи от позиция 8.

Например, следното правило ще забрани всички фрагменти насочени към 192.168.1.1:

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

7.3.6. Разширения към iptables: Нови модули за сравнение

`iptables` е **разширяем**, което означава, че и ядрото, и програмата iptables могат да бъдат разширяване за да предоставят нови възможности.

Някои от тези разширения са стандартни, а други по-екзотични. Разширенията могат да бъдат направени от други хора и да се разпространяват отделно за специфични потребители.

Разширенията към ядрото обикновено се намират в поддиректорията за модули, примерно /lib/modules/2.4.0-test10/kernel/net/ipv4/netfilter. Те се зареждат автоматично при необходимост ако ядрото ви е компилирано с опцията CONFIG_KMOD, така че не трябва да ги вмъквате ръчно.

Разширенията към програмата iptables библиотеки, които обикновено се намират в директорията /usr/local/lib/iptables/, но дистрибуциите ги слагат в /lib/iptables или /usr/lib/iptables.

Разширенията биват два типа: нови действия, и нови модули за сравнение (ще говорим за новите действия малко по-късно). Някои протоколи автоматично предоставят нови проверки: в момента това са TCP, UDP и ICMP както се вижда по-долу.

За тях ще можете да задаване новите проверки на командния ред, след опцията ‘-p’, която зарежда съответното разширение. За изрично указване на нови проверки използвайте опцията ‘-m’ за да заредите разширението, след което допълнителните опции ще бъдат налични.

За да получите помощна информация относно дадено разширение, използвайте опцията с която го зареждате (‘-p’, ‘-j’ или ‘-m’) последвана от ‘-h’ или ‘–help’, примерно:

```
# iptables -p tcp --help
#
```

TCP разширения TCP разширенията се зареждат автоматично ако е указана опцията ‘-p tcp’. Те предоставят следните опции (нито една от тях не съвпада с фрагменти).

–tcp-flags

след която можете да поставите ‘!’ за инверсия и два символни низа от флагове. Те ви позволяват да филтрирате пакета в зависимост от състоянието на някои от тях в TCP заглавната част. Първият низ е маската: списък от флаговете които искате да бъдат проверени. Вторият указва кой/кои от тях трябва да бъдат вдигнати. Например:

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

Това указва да бъдат прегледани всички флагове (‘ALL’ е синоним на ‘SYN,ACK, FIN,RST,URG,PSH’), но само SYN и ACK трябва да са вдигнати. Друг възможен аргумент е ‘NONE’, което означава да няма вдигнати флагове.

–syn

пред него може да има ‘!’, това е по-кратък вариант на ‘–tcp-flags SYN,RST,ACK SYN’.

–source-port

може да се постави ‘!’, след нея един или област от TCP портове. Това могат да бъдат имена на портове, както са указани в /etc/services, или числа. Областите могат да се зададат по няколко начина: два порта с поставени ‘:’ между тях; порт последван от ‘:’ (за да укажете портове по-големи или равни на този); или ‘:’ последвани от порт (за да укажете тези които са равни или по-малки на зададения),.

–sport

е синоним на ‘–source-port’.

–destination-port

и

–dport

са аналогични на тези по-горе, но указват порта на получателя (destination port).

–tcp-option

последван по желание от ‘!’ и число, ще съвпада с пакети чиито TCP опции са равни на това число. Ако пакета няма пълна TCP заглавна част, то той ще бъде отхвърлен автоматично при опит за проверка на TCP опциите.

Обяснение на TCP флаговете Понякога е полезно да разрешите само TCP връзки в едната посока, а да забраните тези в другата. Например, може би искате да разрешите връзките към някой външен уеб сървър, но не и тези идващи от него.

Първото нещо което идва на ум е да блокирате TCP пакетите идващи от сървъра. За съжаление, за да работят въобще, TCP връзките изискват пакети преминаващи и в двете посоки.

Решението е да блокирате само тези пакети, които се използват за иницииране на връзка. Тези пакети се наричат **SYN** пакети (добре де, това са пакети с вдигнат SYN флат, и спуснати RST и ACK флагове, за по-кратко

ги наричаме SYN пакети). Като забраните само тези пакети, можете да спрете опитите за осъществяване на връзка отвън още в зародиш.

Опцията '–syn' се използва точно за това: тя е валидна само в правила в които е указан TCP за протокол. Например за да зададете пакетите които опитват да направят TCP връзки идващи от 192.168.1.1:

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

Този флаг може да бъде инвертиран като пред него поставите '!', което означава всички пакети, освен тези за иницииране на връзка.

UDP разширения Тези разширения се зареждат автоматично,ако е указана опцията '-p udp'. Те предоставят опциите '–source-port', '–sport', '–destination-port' и '–dport' които са аналогични на тези описани за TCP по-горе.

ICMP разширения Тези разширения се зареждат автоматично ако е указан флагът '-p icmp'. Те осигуряват само една нова опция:

–icmp-type

евентуално последван от '!'за отрицание и име на icmp тип (примерно 'host-unreachable'), или числена стойност (примерно '3'), или числа за тип и код разделени с '/' (пример: '3/3'). Списък от валидните имена на icmp типове можете да видите с '-p icmp –help'.

Други разширения за сравняване Тези разширения в пакета netfilter са за демонстрация и (ако са инсталирани) могат да бъдат извикани с опцията '-m'.

mac

За да използвате този модул трябва задължително да укажете '-m mac' или '–match mac'. Той се използва за сравняване на Ethernet (или MAC) адреса на изпращача на входящите пакети, и следователно е приложим само за пакети преминаващи през PREROUTING и INPUT веригите. Той предоставя само една опция:

–mac-source

може да бъде последвана от '!', и ethernet адреса в шестнайстичен вид разделен с ':', например '–mac-source 00:60:08:91:CC:B7'.

limit

За да използвате този модул трябва задължително да зададете '-m limit' или '–match limit'. Той се използва за ограничаване на броя съвпадения за единица време, примерно за ограничаване на log съобщенията. Той ще съвпада само определен брой пъти в секунда (по подразбиране 3 съвпадения за час, с burst от 5). Модулът приема два незадължителни аргумента:

–limit

последван от число; задава средно колко най-много съвпадения за една секунда са разрешени. Можете да укажете изрично каква е мерната единица, използвайки '/second', '/minute', '/hour' или '/day', или части от тях ('5/second' е същото като '5/s').

–limit-burst

послеван от число, указва максималния burst преди лимита да влезе в сила.

Това разширение най-често се използва ограничаване на LOG съобщенията. За да разберете как работи, нека разгледаме следното правило, което логва пакети използвайки ограничението по подразбиране:

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

Първият път когато с равилото съвпадне пакет, той ще бъде записан в журналния файл; всъщност, стойността на burst по подразбиране е 5, така че първите пет пакета ще бъдат записани. След това, ще изминат 20 минути преди да бъде отбелязан пакет съвпадащ с това правило, независимо колко такива има. Също така, за всеки 20 минути през които не е преминал съвпадащ пакет burst ще възвръща по единица от стойността си; следователно ако през правилото не преминават пакети в продължение на 100 минути, то burst ще бъде напълно презареден, както в началото.

Забележка: в момента не можете да създадете правило с време за презареждане по-голямо от 59 часа, следователно ако зададере скорост от един пакет на ден, стойността указана на burst трябва да бъде по-малка от 3.

Друго приложение на този модул е за предпазване от различни атаки тип "Отказ на услуга"(DoS).

Защита от Syn-flood:

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

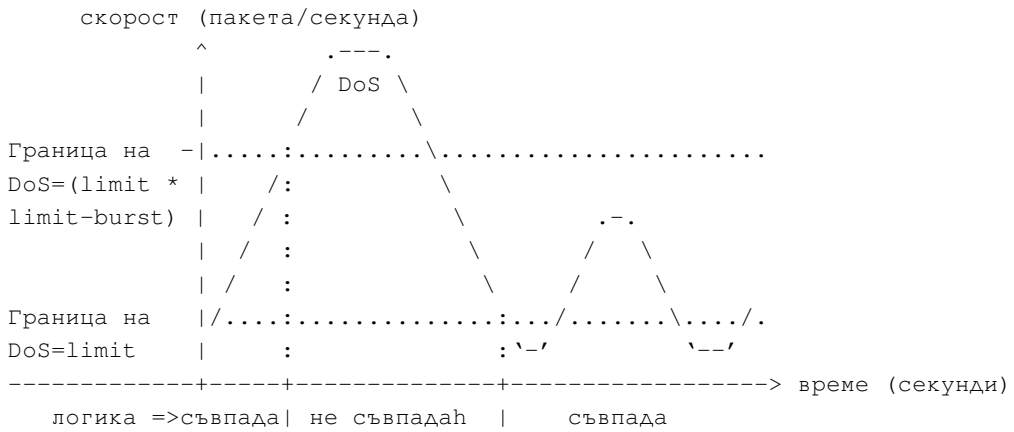
Furtive port scanner:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCE
```

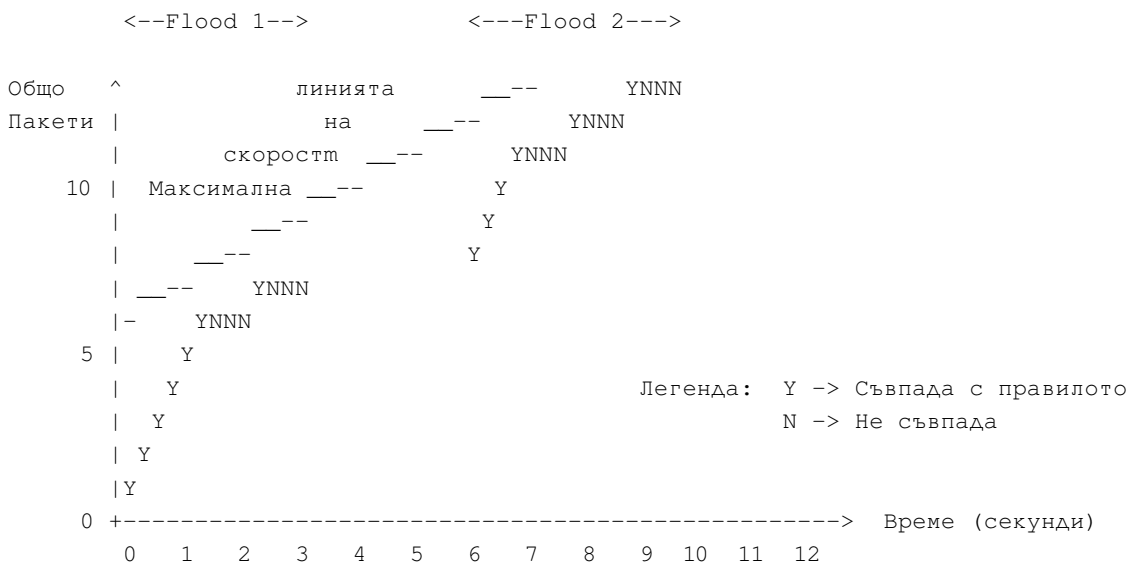
Ping на смъртта:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Този модул работи като "хистерезисна врата както е показано на графиката по-долу.



Да кажем, че сме задали лимит от 1 пакет за секунда, и burst от 5 пакета, но пакетите пристигат по 4 в секунда в продължение на 3 секунди, след което спират и след 3 сек отново идват по 4 пакета/секунда.



Виждате,че на първите пет пакета е позволено да надвишат указания лимит от един пакет за секунда, след което ограничението влиза в сила. Ако има пауза, то ще бъде позволен още един burst, но без да се надвишава максималната граница указана от правилото (1 пакет за секунда, след като се използва burst).

собственик (owner)

Този модул се опитва да свърже различните характеристики на създателя на пакета, за пакети които се генерират локално. Той е валиден само в OUTPUT веригата, и дори там някои пакети (примерно отговори на ICMP ping) може да нямат собственик, и следователно няма никога да съвпаднатh.

–uid-owner userid

Съвпада ако пакетът е създаден от процес с указаното ефективно (числено) user id.

–gid-owner groupid

Съвпада ако пакетът е създаден от процес с указаното ефективно (числено) group id.

–pid-owner processid

Съвпада ако пакетът е създаден от процес с указаното process id.

–sid-owner sessionid

Съвпада ако пакетът е създаден от процес с указаното стойност на session група.

неправилен (unclean)

Този експериментален модул трябва да бъде указан изрично чрез '-m unclean'или '–match unclean'. Той прави разнообразни случайно подбрани проверки за коректност на пакетите. Този модул не е проверен и не трябва да се използва като мярка за сигурност (най-вероятно той може да направи нещата още по-лоши, защото е възможно в самия модул да има грешки). Модулът не предоставя допълнителни опции.

Модул за състоянията (The State Match) Най-полезният критерий за сравнение се предоставя от модулът за 'състоянията', който интерпретира информацията за следене на връзките предоставена от 'ip_conntrack' модула. Използването му е силно препоръчително.

Задавайки '-m state' можем да използваме допълнителната опция '–state', която е списък от състояния, разделени със запетая, с някое от които пакета трябва да съпвада (флагът '!' указва, че пакета трябва да **не** съвпада с тези състояния). Състоянията могат да бъдат:

NEW

Пакет, който създава нова връзка.

ESTABLISHED

Пакет, който принадлежи към връзка, която вече съществува (примерно пакет-отговор, или изходящ пакет по връзка от която вече са получавани отговори).

RELATED

Пакет, който е свързан с, но не е част от вече съществуваща връзка, примерно ICMP грешка, или (ако е зареден FTP модулът) пакет който инициира връзка за предаване на данни по ftp.

INVALID

Пкает, който не може да бъде идентифициран поради някаква причина: това включва недостатъчно свободна памет и ICMP съобщения за грешки, които не са свързани с никоя от известните връзки. По принцип тези пакети трябва да бъдат спирани.

Ето един пример за употребата на това много полезно разширение:

```
# iptables -A FORWARD -i ppp0 -m state ! --state NEW -j DROP
```

7.4. Указване на действие

След като вече знаем какви проверки можем да извършваме върху един пакет, ни трябва начин по който да кажем какво да се случи с тези пакети който отговарят на тяхните изисквания. Това се нарича **действие** на правилото.

Има две много прости вградени действия: DROP и ACCEPT. Вече се запознахме с тях. Ако пакетът съвпадне с някое правило и указаното действие е едно от тези две, то не се преглеждат други правила: съдбата на пакета е вече решена.

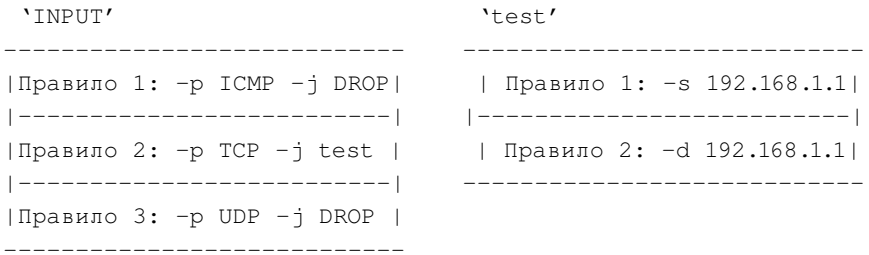
Има два други вида действия, освен вградените: разширения и вериги дефинирани от потребителя.

7.4.1. Вериги дефинирани от потребителя

Една мощна възможност, която `iptables` наследи от `ipchains` е възможността потребителят да създава нови вериги, в допълнение на трите вградени такива (INPUT, FORWARD и OUTPUT). Има конвенция имената на веригите дефинирани от потребителя да се изписват с малки букви, за да се различават по-лесно. (как се създават нови вериги е описано по-долу в 7.5 (Действия върху цяла верига)).

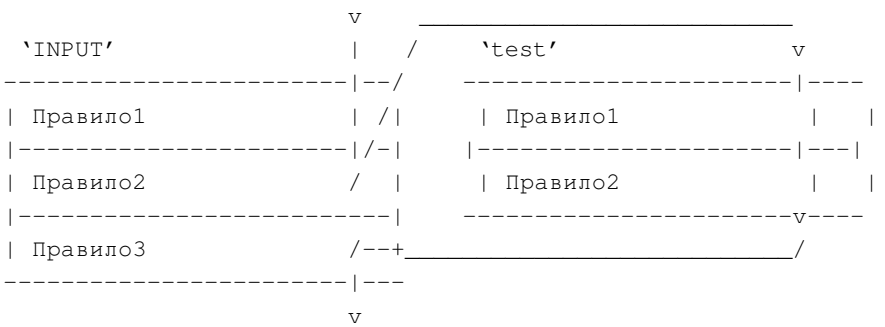
Когато пакет съвпадне с правило чието действие е верига дефинирана от потребителя, то пакетът преминава през правилата в тази верига. Ако в нея не се реши съдбата на пакета, то след като пакетът е преминал през всички правила, той продължава в следващото правило от текущата верига.

Време е за още ASCII арт. Да вземем две (безмислени) вериги: `INPUT` (вградената верига) и `test` (верига дефинирана от потребителя).



Нека един TCP пакет идва от 192.168.1.1 и отива към 1.2.3.4. Той попада в в `INPUT` веригата, и се тества спрямо правило 1 - не съвпада. Правило 2 съвпада и негово действие е `test`, така че следващото правило което се проверява е в началото на `test`. Правило 1 в `test` съвпада, но не е указано действие, и се преглежда следващото правило - правило 2. То не съвпада, следователно вече сме достигнали до края на веригата. Връщаме се обратно в `INPUT` веригата, където последно проверихме правило 2, следователно сега преглеждаме правило 3, което също не съвпада.

В крайна сметка пътя на пакета е:



Вериги дефинирани от потребителя могат да насочат пакета към други вериги дефинирани от потребителя (внимавайте да не се получи затворен кръг в който пакета да обикаля безкрайно: вашите пакети ще бъдат

отхвърлени ако се установи, че е попаднал в затворен кръг).

7.4.2. Разширения на iptables: Нови действия

Другият вид разширение е новото действието. То се състои от модул за ядрото и евентуално разширение за `iptables` което осигурява нови опции за командния ред. Има няколко такива разширения в стандартната netfilter дистрибуция:

LOG

Този модул позволява всяко съзпадение на пакет да се отбелязва в журнален файл. Той предоставя следните допълнителни опции:

–log-level

Последвана от номер или име на ниво. Валидни имена са (нечуствителнни към малки/главни букви) ‘debug’, ‘info’, ‘notice’, ‘warning’, ‘err’, ‘crit’, ‘alert’ и ‘emerg’, отговарящи на номера от 7 до 0. Виж man страницата на syslog.conf за повече информация относно тези нива. По подразбиране се ползва ‘warning’.

–log-prefix

Последвана от символен низ с дължина до 29 символа, който се изпраща преди всяко съобщение, за да позволи по-лесното му разпознаване.

Този модул е най-полезен след използване на limit, за да не препълните журналните файловете си.

REJECT

Този модул има същия ефект като ‘DROP’, но на подателя на пакета се изпраща ICMP ‘port unreachable’ съобщение за грешка. Забележете, че ICMP съобщение за грешка не се изпраща ако (виж RFC 1122):

- Пакетът който бива отфилтриран е ICMP съобщение за грешка или някакъв неизвестен ICMP тип.
- Пакетът който бива отфилтриран е фрагмент, различен от първия.
- Изпратили сме прекалено много ICMP съобщения за грешки към този адрес в последно време (виж /proc/sys/net/ipv4/icmp_ratelimit).

REJECT също така може да има аргумент ‘–reject-with’ който променя вида на пакета изпращан като отговор: виж man страницата.

7.4.3. Специални вградени действия

Има две специални вградени действия: `RETURN` и `QUEUE`.

`RETURN` има същия ефект както и достигането на края на текущата верига: ако правилото се намира в някоя от вградените вериги ще се изпълни политиката на тази верига. В случай, че е във верига дефинирана от потребителя, то преглеждането ще продължи в предишната верига, точно след правилото което е прехвърлило пакета към текущата.

`QUEUE` е специално действие, което изпраща пакета в опашка за обработка от потребителя. За да има полза от това са необходими още два компонента:

- "queue handler който се занимава със същинската работа по прехвърлянето на пакетите между пространството на ядрото и на потребителя (kernel and userspace) и
- потребителска програма,която да получи и евентуално да редактира пакетите, след което да реши тяхната съдба.

Стандартният "queue handler"за IPv4 е модулът ip_queue, който се разпространява с ядрото и е маркиран като експериментален.

Следва кратък пример за това как може да се използват iptables за изпращане на пакети за обработка в потребителското пространство:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

С това правило локално генерираните изходящи ICMP пакети (като например тези от ping) се препращат към ip_queue module, който след това се опитва да ги предаде на потребителската програма. Ако няма програма която да приеме пакетите, то те се отхвърлят .

За да напишете такава програма използвайте libipq API-то. То се разпространява с iptables. Примерен код може да бъде намерен в "testsuite tools"(примерно redirect.c) в CVS.

Статуса на ip_queue може да се провери чрез:

```
/proc/net/ip_queue
```

Максималната дължина на опашката (т.е. броят пакети препратени към потребителската програма, за които все още не е получен отговор със съответното действие) може да се контролира чрез:

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

Стойността по подразбиране е 1024. След като този лимит бъде достигнат, всички нови пакети ще бъдат отхвърляни докато дължината на опашката намалее под лимита. Добрите протоколи, като TCP например, интерпретират загубените пакети като претоварване на връзката, и би трябвало да престанат да изпращат пакети след като опашката се запълни. Въпреки това, може да е необходимо да се експериментира за да се определи оптималната дължина на опашката за конкретната ситуациа, в случай че тази по подразбиране е прекалено малка.

7.5. Действия върху цяла верига

Енда много полезна възможност на `iptables` е тази за групиране на няколко правила във верига. Можете да кръстите веригата както си поискате, но аз ви препоръчвам да използвате само малки букви, за да не ги объркате с вградените вериги или действия. Имената на веригите могат да бъдат дълги до 31 символа.

7.5.1. Създаване на нова верига

Нека да създадем една нова верига. И понеже имам страхотно въображение, ще я наречем `test`. Можем да използваме опциите ‘-N’ или ‘–new-chain’:

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

Не е ли просто? Сега можете да добавяте правила към веригата, както е показано по-долу.

7.5.2. Изтриване на верига

Изтриването на верига е също толкова просто, използвайки опциите ‘-X’ или ‘–delete-chain’. Защо ‘-X’ ли? Ами, всички хубави букви вече бяха заети.

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

Има няколко ограничения за изтриването на вериги: те трябва да бъдат празни (виж 7.5.3 (Изтриване на правилата от верига) по-долу) и също така не трябва да бъдат указани като действие в някое правило. Не е възможно да изтриете никоя трите вградени вериги.

Ако не укажете конкретна верига, то *всички* вериги дефинирани от потребителя ще бъдат изтрити, ако това е възможно.

7.5.3. Изтриване на правилата от верига

Има един много лесен начин за изтриване на всички правила от някоя верига, просто използвайте командата '-F' (или '–flush').

```
# iptables -F FORWARD
#
```

Ако не укажете верига, то правилата във *всички* вериги ще бъдат изтрити.

7.5.4. Извеждане на правилата във верига

Можете да видите списък със всички правила в дадена верига използвайки командата '-L' (или '–list').

Стойността 'x references', която се изписва след името на всяка създадена от потребителя верига, е броят на правилата които имат указана тази верига за свое действие. Този брояч трябва да бъде нула (и във веригата да няма правила) преди тя да може да бъде изтрита.

Ако не се укаже име на кокнтретна верига, то се извежда съдържанието на всички вериги, дори и празните.

Има три опции които може да се използват заедно с '-L'. Първата, '-n' (numeric) е много полезна, тъй като предотвратява опитите на `iptables` да проверява имената съответстващи на IP адресите, които (ако използвате DNS, както повечето хора) биха причинили големи закъснения, в случай че DNS услугата не е настроена правилно, или сте отфилтрирали DNS заявките. Също така TCP и UDP портовете ще бъдат изведени като цифри, вместо с имена.

Опцията '-v' ще ви покаже в детайли цялата информация за правилата, като броячите за пакети и байтове, сравнения с TOS полето и интерфейсите. Без нея тези стойности няма да бъдат изведени.

Забележете, че броячите за пакети и байтове се извеждат със суфикси K', 'M' или 'G' съответно за 1000, 1,000,000 и 1,000,000,000. Ако използвате и опцията '-x' (expand numbers) ще бъдат отпечатани целите числа, независимо колко са големи.

7.5.5. Рестартиране (Нулиране) на броячите

Полезно е да имате възможността да нулирате броячите. Това може да бъде направено с опцията '-Z' (или '–zero').

Нека разгледаме следния пример:

```
# iptables -L FORWARD
# iptables -Z FORWARD
#
```

В този случай е възможно да преминат пакети в промеждутъка от време между извикването на '-L' и извикването на '-Z', които няма да бъдат отчетени. Поради тази причина използвайте '-L' и '-Z' *заедно*, за да нулирате броячите в момента в който ги прочетете.

7.5.6. Задаване на политика

Споменахме какво се случва когато един пакет достигне до края на някоя от вградените вериги, когато разглеждахме пътя по който преминават пакетите по-горе. В този случай, съдбата на пакета се определя от **политиката** на веригата. Само вградените вериги (`INPUT`, `OUTPUT` и `FORWARD`) имат политики, защото ако пакетът достигне до края на някоя верига дефинирана от потребителя, то той ще продължи пътя си през предходната верига.

Политиката може да бъде `ACCEPT (приеми)` или `DROP (отхвърли)`, например:

```
# iptables -P FORWARD DROP
#
```

8. Използване на ipchains и ipfwadm

В дистрибуцията на netfilter има два модула с имена ipchains.o и ipfwadm.o. Включете някой от тях във вашето ядро (ЗАБЕЛЕЖКА: те са несъвместими с ip_tables.o!). Сега вече можете да използвате ipchains или ipfwadm, както в добрите стари времена.

Те ще бъдат поддържани още известно време. Мисля, че подходяща формула е 2 * [предупреждение за изключване - първа стабилна версия], след датата на която е налична стабилна версия на заместителя. Това означава, че поддръжката вероятно ще бъде прекратена в Линукс 2.6 или 2.8.

9. Съвместяване на NAT и филтриране на пакети

Много често се изисква да правите едновременно NAT (виж NAT HOWTO) и филтриране на пакети. Добрата новина е, че те работят доста добре заедно.

Изградете своя пакетен филтър, като напълно пренебрегване факта, че използвате NAT. Адресите на изпращача и получателя, които ще види пакетния филтър ще бъдат ‘истинските’ адреси. Например, ако правите DNAT за да изпращате всички връзки за 1.2.3.4 на порт 80 към 10.1.1.1 на порт 8080, то пакетния филтър ще види пакети отиващи към 10.1.1.1 на порт 8080 (истинското направление), а не към 1.2.3.4 на порт 80. По подобен начин може да игнорирате и употребата на masquerading: ще изглежда, че пакетите ’идват’ от техните истински вътрешни IP адреси (примерно 10.1.1.1), и отговорите ще се връщат пак там.

Може да използвате ‘state’ разширенията без да натоварване филтъра допълнително, защото използването на NAT изисква употребата на механизмите за преследяване състоянието на връзките. За да подобрим простият пример за маскиране от NAT HOWTO, като забраним всички нови връзки идващи от интерфейса ppp0 interface, може да използваме това:

```
# Маскирай всичко излизащо през ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Забрани NEW и INVALID идващи от ppp0.
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

# Включване на IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10. Разлики между iptables и ipchains

– Първо, имената на вградените вериги са променени от малки на ГЛАВНИ букви, защото през веригите INPUT и OUTPUT сега преминават само пакетите изпратени от и насочени към локални процеси, работещи на пакетния филтър. Преди през тях минаваше всичкият входящ и изходящ трафик.

– Опцията ‘-i’ вече означава входящ интерфейс, и работи само във веригите INPUT и FORWARD. Правилата във веригите FORWARD и OUTPUT, които използват ’-i’ трябва да бъдат променени със съответните им ползващи ‘-o’.

– TCP и UDP портовете трябва да бъдат указвани с опциите –source-port или –sport (съответно –destination-port/–dport), и да бъдат след опциите ‘-p tcp’ или ‘-p udp’, защото те зареждат съответните TCP и UDP разширения. /-

– Опцията TCP -y сега е –syn, и трябва да бъде след ‘-p tcp’.

– Действието DENY стана DROP, най-накрая.

– Нулирането на една верига, заедно с прочитане на броячите работи.

– Нулирането на броячите на вградените вериги, също булира и броячите на политиките.

– Извеждането на броячите на няколко вериги е атомарна операция.

– REJECT и LOG действията са като разширения, което означава, че са и отделни модули за ядрото.

– Имената на веригите могат да бъдат до 31 символа.

– MASQ се преименува на MASQUERADE и използва различен синтаксис. REDIRECT, въпрени че е със същото има, има нов синтаксис. Вижте NAT-HOWTO за повече информация по тези две действия.

– Опцията -o вече не се използва за премасочване на пакети към потребителското пространство (виж -i по-горе). Пакетите се пренасочват с помощта на действието QUEUE.

– Вероятно още куп други неща, за които не се сещам.

11. Съвети относно изграждането на пакетен филтър

Разпространена практика в областта на компютърната сигурност е да се забрани всичко по подразбиране, след което да се разреши само това което е необходимо. Това може да бъде изразено така ‘всичко, което не е изрично разрешено, е забранено’. Аз бих ви препоръчал точно този подход, ако сигурността е от най-голямо значение за вас.

Не стартирайте услуги от които не се нуждаете, дори и да си мислите, че сте блокирали достъпа до тях.

Ако изграждаде защитна стена, спрете всички процеси очакващи връзка от мрежата, и блокирайте всички пакети. Сега стартирайте само услугите които ви трябват и разрешете преминаването на пакетите за които това е необходимо.

Препоръчвам задълбочен подход към сигурността, комбинирайте tcp-wrappers (за връзките към самия пакетен филтър), проксита (за връзките преминаващи през него), проверка на маршрутизацията и пакетно филтриране. Проверка на маршрутизацията, означава пакетите които пристигат от неочакван интерфейс да бъдат блокирани. Например, ако вашата вътрешна мрежа има адрес 10.1.1.0/24, и пакет с такъв адрес на изпращача пристигне на външния ви интерфейс, то той ще бъде отхвърлен. Това може да бъде включена за даден интерфейс (ppp0) ето така:

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

Или за всички интерфейси:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
#
```

Дебиан прави това по подразбиране когато е възможно. В случай, че имате асиметрична маршрутизация (т.е. очаквате пакети пристигащи от странни посоки), то трябва да спрете филтрирането за тези интерфейси.

Журнализирането на събитията е много полезно когато изграждате защитната стена. Така ако нещо не работи както трябва лесно ще можете да видите къде е проблема. Когато обаче тя започнете да работи в реални условия винаги използвайте и 'limit' модула. Така ще се предпазите от препълване на журналните файлове.

Силно препоръчвам използването модулите за следене на връзките. Те повишават натоварването на системата, тъй като всички връзки се следят, но са много полезни при контролирането на достъпа до вашата мрежа. Може да се наложи да заредите 'ip_conntrack.o' модула ако вашето ядро не поддържа автоматично зареждане на модули и той не е вграден в ядрото. Ако искате следенето на работи правилно и с по сложни протоколи, то трябва да заредите съответните помощни модули (пример: 'ip_conntrack_ftp.o').

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet from ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet not from
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Темата за изграждане на добра защитна стена е извън обхвата на този документ. Прегледайте Security HOWTO за повече информация относно тестване и проучване на вашата машина.