Linux 2.4 Packet Filtering HOWTO

Rusty Rusell, mailing list netfilter@lists.samba.org $Revision 1.18 $ $Date: 2001/05/04 20:58:43 $
Превод на български: Александър Цанков, alexc@mbox.digsys.bg

Този документ описва как да се използва iptables за филтриране на пакети за Linux ядра 2.4

Съдържание

Забележка на преводача: Не съм завършил английска гимназия и не претендирам за 100% достоверност на превода. Всички предложения относно промяна на съдържанието или дизайна на документа са добре дошли.

1. Въведение

Добре дошъл, Драги читателю!

Предполага се, че знаеш какво е IP адрес, мрежов адрес, мрежова маска и DNS. Ако не – препоръчвам ти да прочетеш HOWTO-то за мрежовите концепции (Network Concept HOWTO).

Това HOWTO прескача между мило въведение (което ще ви остави развълнувани, но незащитени в Реалния свят) и грубо пълно разкриване (което ще остави всички, с изключение на най-коравите души, смутени, параноични и търсещи тежко оръжие).

Вашата мрежа не е сигурна. Проблемът с позволяването на бързи и удобни комуникации, но ограничавайки използването им за добри цели, е сходен с други инфраструктурни проблеми, например разрешаване на свободата на словото, но забрана на възможността да извикаш "Пожар!" в претъпкан театър. Това няма да бъде разрешено в обема на това HOWTO.

Само вие може да решите какъв ще бъде компромисът. Ще се опитам да ви инструктирам как да използвате някои от наличните инструменти и някои слабости, с които да сте наясно, с надеждата, че ще ги използвате за добро, а не за лошо. Друг еквивалентен проблем.

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

Има 3 официални сайта:

Благодарности на *Filewatcher* <http://netfilter.filewatcher.org>

Благодарности на *The Samba Team and SGI* <http://www.samba.org/netfilter>

Благодарности на *Jim Pick* <http://netfilter.kernelnotes.org>

За официален пощенски списък вижте Samba Listserver <http://lists.samba.org>

3. Е, какво е пакетен филтър?

Пакетният филтър е софтуер, преглеждащ заглавната част на пакетите по време на тяхното движение, и решаващ съдбата на целия пакет. Решението може да бъде **DROP** (т.е. отхвърляне на пакета, все едно, че никога не е пристигал), **ACCEPT** (т.е. разрешаване на пакета да премине) или някои по-сложни неща.

Пакетният филтър за Linux е вграден в ядрото или компилиран до модул за ядрото и има някои по-хитри неща, които могат да се извършат с пакетите, но общият принцип на преглеждане на заглавните части и решаване на съдбата на пакетите си е налице.

3.1 Защо бих искал да извършвам пакетно филтриране?

Контрол, Защита, Информираност.

Контрол:

Когато използвате Linux, за да свържете вашата вътрешна мрежа към друга мрежа (например Internet), вие имате възможност да позволите някои типове трафик и да забраните останалите. За пример, заглавната част на пакетите съдържа крайния адрес на пакета, така че вие може да предотвратите достигането на пакетите до дадена част от външната мрежа. Като друг пример, аз използвам Netscape за достъп до архивите Dilbert. На Web страницата има реклами от doubleclick.net и Netscape изразходва времето ми, бодро зареждайки ги. Инструктирайки пакетния филтър да не пропуска пакети от и към адресите, притежавани от doubleclick.net, решава този проблем (има и по-добър начин за извършване на това: виж Junkbuster).

Защита:

Когато Linux машината е единственото нещо между хаоса на Internet и вашата хубава, подредена мрежа, е добре да знаете, че можете да наложите ограничения върху нещата, чукащи на вратата ви. Например, вие може да разрешите всичко, излизащо от мрежата ви, но може да се притеснявате от добре познатия "Ping of Death", идващ от зложелателни външни източници. Като друг пример, вие може да не желаете telnet-ване отвън към вашата Linux машина, дори ако всичките ви акаунти са защитени с пароли. Може би вие (както повечето хора) желаете да бъдете Internet изследовател, а не сървър (доброволно или не). Просто не позволявайте никой да прониква навътре, използвайки пакетния филтър за отхвърляне на входящи пакети, целящи създаване на връзки.

Иформираност:

Понякога лошо конфигурирани машини от локалната мрежа ще решат да изплюят пакети във външния свят. Хубаво е така да се конфигурира пакетният филтър, че да ви позволява да узнаете, ако настъпи нещо ненормално; може би вие ще направите нещо или просто сте с любопитен характер.

3.2 Как да извършвам пакетно филтриране с Linux?

Linux ядрата имат възможност за пакетно филтриране от 1.1 сериите. Първата генерация, базирана на ipfw от BSD, беше портирана от Alan Cox в края на 1994. Тя бе подобрена от Jos Vos и други за Linux 2.0; потребителският инструмент 'ipfwadm' контролираше правилата за филтриране на ядрото. В средата на 1998, за Linux 2.2, аз преработих ядрото доста здраво с помощта на Michael Neuling и въведох потребителския инструмент 'ipchains'. Най-накрая, инструментът от 4 генерация 'iptables' и друга преработка на ядрото се появиха в средата на 1999 за Linux 2.4. Iptables е обектът, върху което се концентрира това HOWTO.

Вие се нуждаете от ядро, имащо инфраструктурата на мрежовия филтър в себе си: мрежовият филтър (netfilter) е общата рамка вътре в Linux ядрото, в която могат да се вмъкват други неща (като модулът iptables). Това означава, че се нуждаете от ядро 2.3.15 или по-ново и отговор 'Y' на CONFIG_NETFILTER при конфигурирането на ядрото.

3.2.1 iptables

Инструментът iptables вмъква и изтрива правила от таблицата за пакетно филтриране на ядрото. Това означава, че настройките, които сте направили, ще бъдат изгубени при рестартирането; виж 3.2.2 (Създаване на постоянни правила) за да подсигурите тяхното възстановяване при следващото зареждане на Linux.

`Iptables` заменя `ipfwadm` и `ipchains`: виж 8 (Използване на ipchains и ipfwadm) за това как безболезнено да се избегне използването на iptables, ако вие ползвате някоя от другите програми.

3.2.2 Създаване на постоянни правила

Текущата настройка на firewall-а ви се съхранява в ядрото и следователно ще бъде изгубена при рестартирането. Написването на iptables-save и iptables-restore са в моя TODO списък. Когато се появят, ще бъдат супер, обещавам.
Засега, поставете необходимите команди за настройка на правилата в инициализационен скрипт. Подсигурете изпълнението на нещо интелигентно, ако някоя от командите не може да се изпълни (обикновено 'exec /sbin/sulogin').

4. Кой, по дяволите, си ти и защо си играеш с ядрото ми?

Аз съм Rusty; поддържам Linux IP Firewall-а и съм още един програмист, който бе на точното място в точното време. Написах ipchains (виж 3.2 (Как да извършвам пакетно филтриране с Linux?) по-горе, благодарение на помощта на хората, които свършиха истинската работа) и научих достатъчно, за да създам пакетно филтриране по това време. Надявам се.

Watch Guard <`http://watchguard.com`>, прекрасна компания, продаваща хубавия plug-in Firebox, ми предложи заплащане за нищо, така че аз мога да прекарвам времето си, пишейки тези неща и поддържайки моите предни работи. Предсказах 6 месеца, а ми отне 12, но накрая почувствах, че работата е свършена добре. Много пренаписвания, отказ на твърдия диск, откраднат лаптоп, няколко скапани файлови системи и един счупен монитор по-късно – това е.

Искам да разсея някои грешни представи за мен: аз не съм kernel гуру. Аз знам тези неща, защото моята работа с ядрото създаде връзките ми с някои от тях: David S. Miller, Alexey Kuznetsov, Andy Kleen, Alan Cox. Те, обаче, са заети с извършване на големите магии, оставяйки ме да газя в плитчините, където е безопасно.

5. Бърз наръчник на Rusty за пакетно филтриране

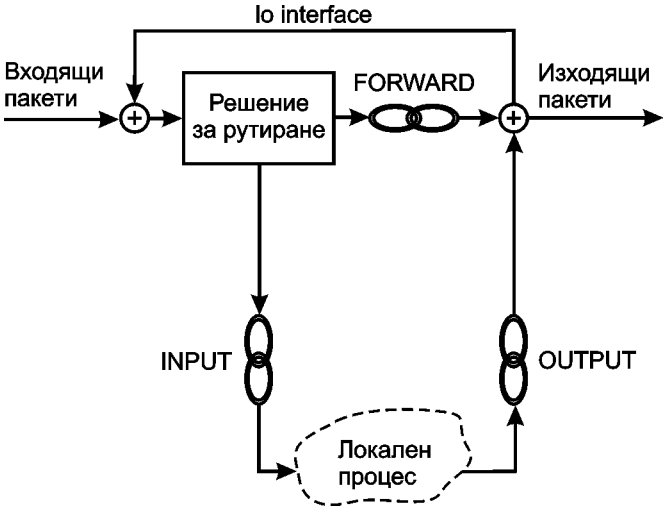
Повечето хора имат просто една PPP връзка с Internet и не искат никой да влиза в мрежата им или във firewall-а:

```
## Вмъкване на модулите за проследяване на връзката (ако не са вградени в ядрото).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Създаване на верига, блокираща всички нови връзки, освен ако не идват 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

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

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

6. Как пакетите преминават през филтрите?

При зареждане на ядрото има три набора от правила във филтриращата таблица. Тези набори се наричат вериги на firewall-а или просто вериги. Тези три вериги се наричат **INPUT**, **OUTPUT** и **FORWARD**.



Тези вериги са показани на фигурата (Забележка: това е много различно аранжиране от 2.0 и 2.2 ядрата).

Трите вериги от рисунките представляват трите вериги, споменати по-горе. Когато пакет достигне до верига от диаграмата, тя се проверява за определяне съдбата на пакета. Ако веригата казва DROP, пакетът умира там, но ако веригата казва ACCEPT, пакетът продължава пътя си по диаграмата.

Веригата е набор от **правила**. Всяко правило гласи: “Ако заглавната част на пакета изглежда по този начин, то ето какво трябва да се направи с пакета”. Ако заглавната част на пакета не отговаря на критериите на правилото, то тогава ядрото търси **политиката** на веригата, за да реши какво да прави. В правилно защитените системи тази политика инструктира ядрото да DROP-не пакета.

1. Когато пакетът влиза отвън (например през Ethernet карта), ядрото първо проверява целта на пакета: това се нарича “рутиране”.

2. Ако пакетът е предназначен за тази машина, пакетът преминава надолу по диаграмата към веригата INPUT. Ако той премине през нея, всеки процес, чакащ този пакет, ще го получи.

3. В противен случай, ако в ядрото не е активирано пренасочване (forwarding) на пакети (или ако ядрото не знае как да пренасочи пакета), пакетът се отхвърля. Ако пренасочването е активирано и пакетът е предназначен за друг мрежов интерфейс (ако имате такъв), тогава пакетът поема надясно по диаграмата към веригата FORWARD. Ако той бъде ACCEPT-нат, ще бъде изпратен навън.

4. Най-накрая, програма, стартирана на тази машина, може да изпрати пакети. Тези пакети преминават непосредствено през веригата OUTPUT: ако тя каже ACCEPT, пакетите преминават към интерфейса, за когото са предназначени.

7. Използване на iptables

Iptables има детайлна man страница (`man iptables`), ако се нуждаете от нещо по-детайлно или конкретно. Тези от вас, които са наясно с ipchains, могат просто да прегледат 10 (Разлики между iptables и ipchains); те са много подобни.

Има няколко различни неща, които могат да се вършат с `iptables`. Вие започвате с три вградени вериги: `INPUT`, `OUTPUT` и `FORWARD`, които не можете да изтриете. Нека да видим операциите за управление на цяла верига:

1. Създаване на нова верига (-N)

2. Изтриване на празна верига (-X)

3. Промяна на политиката на вградена верига (-P)

4. Преглед на правилата във верига (-L)

5. Изчистване на правилата във верига (-F)

6. Нулиране на пакетните и байтовите броячи на всички правила във верига (-Z)

Има няколко начина за манипулиране на правилата във верига:

1. Добавяне на ново правило във верига (-A)
2. Вмъкване на ново правило на определена позиция във верига (-I)
3. Подмяна на правило на определена позиция във верига (-R)
4. Изтриване на правило на определена позиция във верига (-D)
5. Изтриване на първото правило от верига, съвпадащо с критерий (-D)

7.1 Какво ще видите, когато се стартира компютърът ви?

Iptables може да бъде модул, (наречен iptable_filter.o), който трябва автоматично да се зареди, когато за пръв път стартирате `iptables`. Той може да бъде и вграден в ядрото.
Преди да са стартирани каквито и да са команди с iptables (бъдете внимателни: някои дистрибуции ще стартират iptables в своите инициализационни скриптове), няма правила в нито една от вградените вериги (INPUT, OUTPUT и FORWARD) и всички вериги ще имат политика ACCEPT. Вие може да промените политиката по подразбиране чрез добавяне на опцията 'forward=0' в модула iptables_filter.

7.2 Операции върху единично правило

Това е същината на пакетното филтриране: манипулиране на правила. Най-общо, вие вероятно ще използвате командите за добавяне (-A) и изтриване (-D). Останалите (-I за вмъкване и –R за подмяна) са просто разширения на тези концепции.

Всяко правило определя набор от условия, на които пакетът трябва да отговаря и какво да се прави, ако той отговаря на тях (целта на правилото). Например, вие може да искате да отхвърлите всички ICMP пакети, идващи от IP адреса 127.0.0.1. В този случай нашите условия са, че протоколът трябва да бъде ICMP и че адресът на източника трябва да бъде 127.0.0.1. Нашата цел е 'DROP'.

127.0.0.1 е 'loopback' интерфейса, който вие имате, дори ако нямате реална мрежова връзка. Вие може да използвате програмата 'ping', за да генерирате такива пакети (тя просто изпраща ICMP пакети тип 8 (echo request), на които всички взаимодействащи хостове са задължени да отговорят с ICMP тип 0 (echo reply) пакет). Това прави програмата полезна за тестване.

```
# 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
#
```

Може да видите, че първият ping успява ('-c 1' означава да се изпрати само един пакет).

След това добавяме (-A) към веригата INPUT правило, определящо, че пакети от 127.0.0.1 ('-s 127.0.0.1') с протокол ICMP ('-p ICMP'), трябва да бъдат прехвърлени към DROP ('-j DROP').

След това проверяваме нашето правило, използвайки втория ping. Ще има пауза, преди програмата да се предаде, чакайки отговор, който никога няма да дойде.

Можем да изтрием правило по два начина. Първо, тъй като ние знаем, че това е единственото правило във входната верига, можем да използваме номерирано изтриване,

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

```
#
```

за да изтрием правило номер едно във веригата INPUT.

Вторият начин е да се повтори -A командата, но заменяйки -A с -D. Това е полезно, когато имате комплексна верига и не искате да броите, за да установите, че правило 37 е това, от което искате да се оттървете. В този случай бихме използвали:

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

Синтаксисът на -D трябва да има точно същите опции, както и -A (или -I или -R) командата. Ако имате множество идентични правила в една и съща верига, само първото ще бъде изтрито.

7.3 Филтърни определения

Видяхме използването на '-p' за определяне на протокол и '-s' за определяне на адрес на източника, но има и други опции, които можем да използваме за определяне на характеристики на пакета. Това, което следва е изтощително резюме.

7.3.1 Определяне на адреси на източник (source) и цел (destination)

IP адресите на източник ('-s', '--source' или '--src') и цел ('-d', '--destination' или '--dst') могат да се определят по 4 начина. Най-естественият начин е да се използва пълно име, като 'localhost' или 'linuxhq.com'. Вторият начин е с определяне на IP адрес, например '127.0.0.1'.

Третият и четвъртият начини позволяват определяне на група от IP адреси, например '199.95.207.0/24' или '199.95.207.0/255.255.255.0'. И двата начина определят адресите от 199.95.207.0 до 199.95.207.255 включително; цифрите след '/' указват каква част от IP адреса е значеща. '/32' или '255.255.255.255' е подразбиращата се (съвпада с целия IP адрес). За определяне на всеки IP адрес въобще може да се използва '/0', например:

```
# [Забележка: '-s' е излишно тук]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Този запис рядко се използва, тъй като ефектът е същият, както и без уточняване на опцията '-s' въобще.

7.3.2 Определяне на инверсия

Много флагове, включително '-s' (или '--source') и '-d' (или '--destination'), могат да бъдат имат аргумент, предшестван от '!' за определяне на адреси, които НЕ са равни на зададените. Например '-s ! localhost' определя всеки пакет не идващ от локалния интерфейс.

7.3.3 Определяне на протокол

Протоколът може да бъде определен с флага '-p' (или '--protocol'). Протоколът може да бъде число (ако знаете числовите стойности за IP протоколите) или име за специалните случаи TCP, UDP или ICMP. Регистърът няма значение, така че 'tcp' работи също така, както и 'TCP'.

Името на протокола може да се предшества от '!', за да го инвертира, например '-p ! TCP' за определяне на пакети, които не са TCP.

7.3.4 Определяне на интерфейс

Опциите '-i' (или '--in-interface') и '-o' (или '--out-interface') определят името на **интерфейса** за съвпадение. Интерфейсът е физическото устройство, в което пакетът постъпва ('-i') или от което излиза ('-o'). Можете да използвате командата `ifconfig` за преглеждане на интерфейсите, които са активизирани в момента.

Пакетите, преминаващи през веригата INPUT, нямат изходен интерфейс, така че за никое правило, използващо '-o' в тази верига, няма да се получи съвпадение. Аналогично, веригата OUTPUT няма входен интерфейс, така че за никое правило, използващо '-i' в тази верига няма да се получи съвпадение.

Само пакетите, преминаващи през веригата FORWARD имат входен и изходен интерфейс.

Напълно допустимо е задаването на интерфейси, които не съществуват в момента – за съответното правило няма да има съвпадение, докато интерфейсът не се създаде.

Това е много удобно за dial-up PPP връзки (обикновено интерфейса ppp0).

В специалния случай, когато името на интерфейса завършва '+', ще има съвпадение за всички интерфейси (без значение, дали те съществуват в момента или не), започващи с този стринг. например за задаване на правило, определящо съвпадение за всички PPP интерфейси, трябва да се използва опцията '`-i ppp+`'.

Името на интерфейса може да се предхожда от '!' за да се зададе пакет, който **не** е свързан със съответния интерфейс(и).

7.3.5 Определяне на фрагменти

Съществуват пакети, твърде големи, за да се предадат наведнъж. Когато това стане, пакетът се дели на **фрагменти** и се изпраща като множество пакети. На другият край тези фрагменти се реасемблират за реконструиране на целия пакет.

Проблемът с фрагментите е , че само началният фрагмент съдържа всички полета на заглавната част (IP +TCP, UDP и ICMP), които се изследват. Следващите пакети имат само поднабор на тези полета (IP без полетата за допълнителни протоколи). Следователно, преглеждането на следващите заглавни части на пакетите за протоколи (както това става за TCP, UDP или ICMP разширенията) не е възможно.

Ако вие извършвате проследяване на връзката (connection tracking) или NAT (транслиране на мрежовите адреси), всички фрагменти ще бъдат обединени заедно преди да достигнат до кода за пакетно филтриране, така че вие няма нужда да се притеснявате за фрагментите.

В останалите случаи е важно да се разбере как фрагментите се третират от правилата за филтриране. Правилата за филтриране, според които се търси несъществуваща информация, няма никога да получат съвпадение. Следователно, само първият пакет се третира като всички нормални пакети. Вторият и следващите – не. Следователно за правилото '`-p TCP --sport www`' (определящо порт на източника 'www') никога няма да се получи съвпадение (освен за първия фрагмент). Нито пък за противоположното правило '`-p TCP --sport ! www`'

Може, обаче, да се зададе правило специално за вторите и следващите пакети, използвайки флага '-f' или ('--fragment'). Разрешено е да се зададе и правило, изключващо втория и следващите фрагменти, предхождайки '-f' с '!'.

Обикновено е безопасно да се пропускат вторият и следващите фрагменти, т.к. филтрирането ще се осъществи върху първия пакет и следователно реасемблираето ще бъде предотвратено; известни са обаче програмни грешки, позволяващи пробив в защитата просто чрез изпращане на фрагменти. Изборът е ваш.

Забележка за напредналите: нарушените пакети (TCP, UDP или ICMP пакети, твърде къси, за да може firewall-кодът да разчете портовете или ICMP кода или типа) се отхвърлят, когато се извърши такова изследване. Такива са TCP пакети, започващи от позиция 8.

Като пример, следващото правило ще отхвърли всички фрагменти, предназначени за 192.168.1.1:

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

7.3.6 Разширения към iptables: Нови цели

`Iptables` може да се **разширява**, т.е. ядрото и инструментът iptables могат да бъдат разширявани с цел въвеждане на нови възможности.

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

Разширенията към ядрото обикновено се намират в директорията с модулите към ядрото, например /lib/modules/2.3.15/net. Те се зареждат при нужда, ако вашето ядро е компилирано с установена опция CONFIG_KMOD, така че няма нужда вие ръчно да ги зареждате.

Разширенията към са поделени библиотеки, обикновено намиращи се в директорията /usr/local/lib/iptables, въпреки че при различните дистрибуции те могат да бъдат и в /lib/iptables или /usr/lib/iptables.

Разширенията са два типа: нови цели и нови съвпадения (за новите цели ще говорим малко по-късно). Някои протоколи автоматично предлагат нови проверки: за момента това са TCP, UDP и ICMP, както е показано по-долу.

За тях вие ще имате възможност да зададете нови проверки от командния ред след опцията '-p', което ще зареди тези разширения. За определено новите проверки използвайте опцията '-m' за зареждане на разширенията, след която разширените опции ще са активни.

За помощ относно разширенията използвайте опцията за зареждането им ('-p', '-j' или '-m'), следвана от '-h' или '--help'.

Например:

```
# iptables -p tcp –help
#
```

TCP разширения Зареждат се автоматично, ако е зададено ‘-p TCP’. Осигуряват следните опции (за никоя от тях няма съвпадение за фрагменти):

--tcp-flags

Следвана от опционалното ‘!’ и два стринга за флагове, позволява филтрирането на специфични TCP флагове. Първият стринг от флагове е маска – наборът от флагове, които трябва да се изследват. Вторият стринг от флагове указва кои от тях трябва да бъдат установени, например

```
#iptables –A INPUT –p TCP –tcp-flags ALL SYN,ACK
```

указва, че трябва да се проверят всички флагове (ALL е синоним на ACK, FIN, RST, URG, PSH), но само ASK и FIN трябва да са установени.

--syn

Опционално предхождано от ‘!’, това е съкращение за ‘--tcp-flags SYN,RST,ACK SYN’

--source-port (или --sport)

следвано от опционалното ‘!’ и TCP порт или диапазон от TCP портове, указва търсене на съвпадение на порт – източник на пакета. Портовете могат да се задават с техните имена или с числовите им стойности. Диапазоните са две портови имена, разделени с ‘:’; порт с добавено след него ‘:’ (за определяне на всеки порт с номер, равен или по-голям от зададения); порт, предшестван от ‘:’ (за обозначаване на всички портове с номер, равен или по-малък на зададения).

--destination-port (или --dport)

за обозначаване на порт – цел на пакета. Важат същите опции, както и за --source-port.

--tcp-option

Следвано от опционалното ‘!’ и число указва пакет с поле ‘TCP option’, равно на зададената стойност. Пакетите, нямащи цяла заглавна част, автоматично се отхвърлят, ако възникне опит за изследване на TCP опциите.

Обяснение на TCP флаговете Понякога е полезно да се разрешат TCP връзките в едната посока, но не и в другата. Например, вие може да искате да разрешите връзки към външен WWW сървър, но не и връзка от този сървър.

Естественият подход би бил да се блокират всички пакети, идващи от сървъра. За нещастие, за да работят TCP връзките въобще, се изисква пакетите да се движат и в двете посоки.

Решението е да се блокират само пакетите, използвани за създаване на връзка. Тези пакети се наричат **SYN** пакети (технически, това са пакети с установен SYN флаг и нулирани RST и ACK флагове, но ние ги наричаме за краткост SYN пакети). Чрез забрана само на тези пакети ние можем да спрем усилията за създаване на връзките, които следим.

За това се използва флагът ‘--syn’: той е валиден само за правила, определящи TCP протокол. Например, за задаване на опит за TCP връзка от 192.168.1.1:

```
-p TCP -s 192.168.1.1 –syn
```

Този флаг може да се инвертира, използвайки предшестващото ‘!’, което означава всеки пакет, различен от тези за създаване на връзка.

UDP разширения Зареждат се автоматично, ако е указано ‘-p UDP’. Осигуряват опциите ‘--source-port’ и ‘--destination-port’, аналогично на разгледаните опции за TCP.

ICMP разширения Зареждат се автоматично, ако е указано ‘-p ICMP’. Осигуряват само една нова опция:

--icmp-type

следвано от опционалното ‘!’ и след това ICMP – име (например ‘host-unreachable’) или число, указващо типа (например ‘3’), или числов тип и код, разделени от ‘/’ (например ‘3/3’). Наличните имена на ICMP типовете могат да се видят с командата ‘-p icmp --help’.

Други разширения за съвпадение Другите разширения в netfilter пакета са демонстративни разширения, които (ако са инсталирани) могат да се използват с опцията '-m'.

mac

Този модул трябва изрично да бъде указан с '-m mac' или '--match mac'. Използва се за сравняване на Ethernet (MAC) адреса на източника на входящия пакет, следователно е полезен само за пакети, преминаващи през веригите INPUT и FORWARD. Осигурява една опция:

--mac-source

следвана от опционалното '!' и Ethernet адрес в стандартната хексакод нотация, например '--mac-source 00:60:08:91:CC:B7'.

limit

Този модул трябва изрично да бъде указан с '-m limit' или '--match limit'. Използва се за ограничаване броя на съвпаденията за единица време, както и за ограничаване на log-съобщенията. По подразбиране се приема 3 попадения за час с един пик от 5 попадения. Има два опционални аргумента:

--limit

следван от число – определя максимално разрешения брой съвпадения за секунда. Единицата време може да се зададе и твърдо чрез използването на /second (/s), /minute (/m), /hour (/h), /day (/d).

--limit-burst

следван от число – указва максималния пик, преди ограничението да влезе в сила.

Това съвпадение често може да се използва с целта LOG за извършване на ограничаване на отчетните записи. За да разберем как действа това, нека да разгледаме следното правило, което извършва log-запис на пакетите с подразбиращите се ограничаващи параметри:

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

Първият път, когато се стигне до това правило, ще се извърши запис за пакета; фактически тъй като пикът по подразбиране е 5, първите 5 пакета ще бъдат записани. След това ще има 20 минути преди да се извърши запис за пакет от това правило, независимо колко пакета са стигнали до него. Също така, на всеки 20 минути, изминали без съвпадение на пакет, се разрешава запис на един от петте пикови пакета; ако няма попадение за правилото в течение на 100 минути, възможността за пиков запис ще бъде напълно възстановена; отново сме там, откъдето започнахме.

Забележка: за момента не може да създавате правило с време за възстановяване, по-голямо от 59 часа, така че ако задавате правило, ограничаващо броя на съвпаденията на едно на ден (24 часа), то максималното пиково ниво трябва да бъде по-малко от 3 (59/24=2,45..).

Може да използвате този модул и за предпазване от различни атаки от типа "отказ на услуга" (DoS), използвайки по-високи нива за повишаване на отзивчивостта:

Защита от Syn-flood:

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

От прикриит Port-scanner:

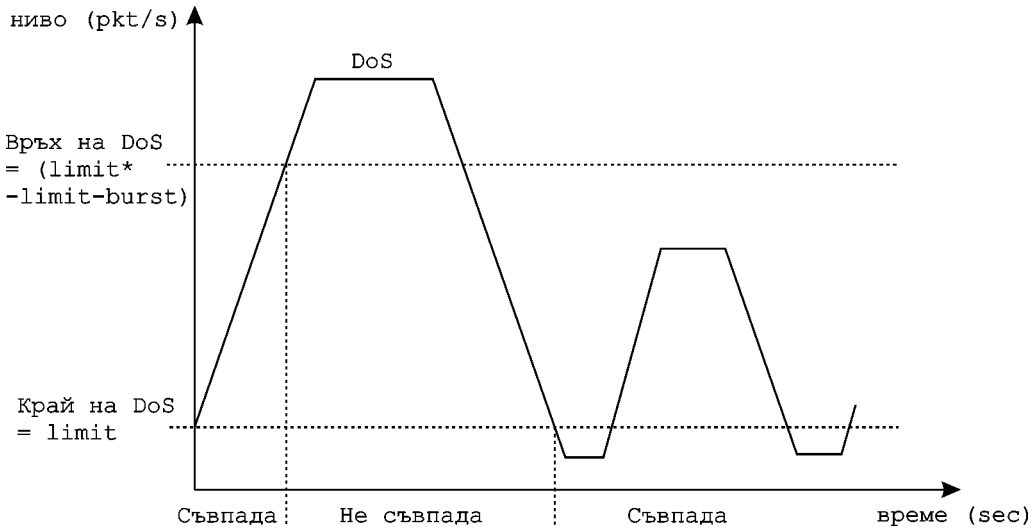
```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s \
-j ACCEPT
```

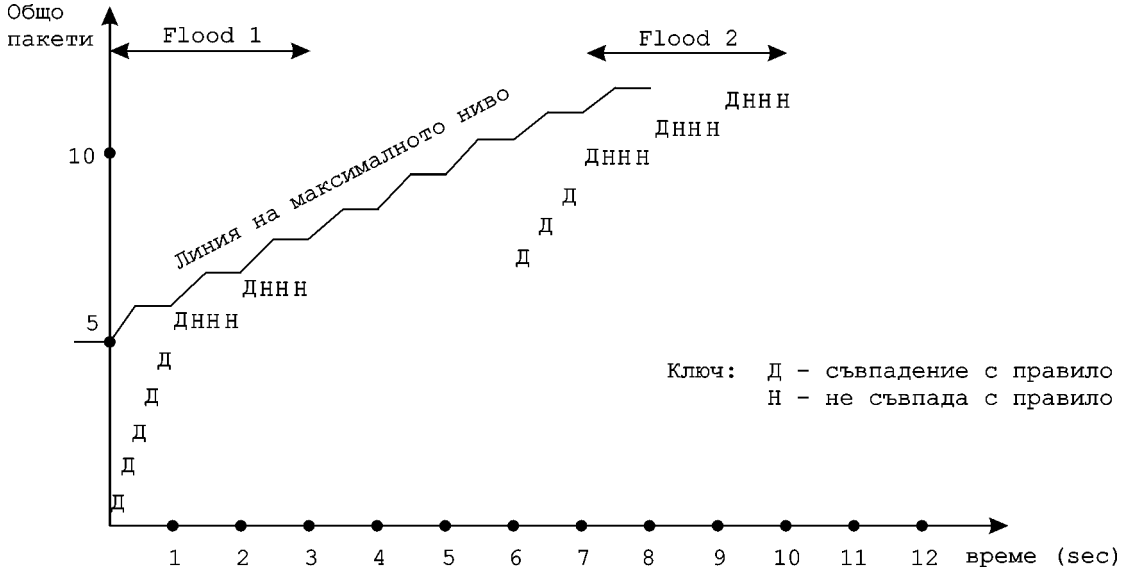
От Ping of Death:

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

Този модул работи като хистерезисен цикъл, както е показано на графиката по-долу:

Да предположим, че сме задали съвпадение 1 пакет за секунда с пик от 5 пакета, но пакетите започват да идват по 4 за секунда, за 3 секунди, след това започват отново след нови 3 секунди.





Вижда се, че за първите 5 пакета се разрешава да превишат границата от 1 пакет за секунда, след което ограничението ги отхвърля. Ако има пауза, ще се позволи нов пик, но той не може да надвиши максималната ограничаваща стойност, указана от правилото (след пика се използва 1 пакет за секунда).

owner

С този модул се цели търсене на съвпадение за различни характеристики на създателя на пакета за локално генерирани пакети. Той е валиден само за веригата OUTPUT и дори в този случай някои пакети (като ICMP ping отговори) могат да нямат собственик, следователно за тях никога няма да има съвпадение.

--uid-owner userid

Съвпадение ще настъпи, ако пакетът е създаден от процес с даден ефективен (числов) потребителски идентификатор (user id)

--gid-owner groupid

Съвпадение ще настъпи, ако пакетът е създаден от процес с даден ефективен (числов) групов идентификатор (group id)

--pid-owner processid

Съвпадение ще настъпи, ако пакетът е създаден от процес с даден идентификатор на процеса (process id)

--sid-owner sessionid

Съвпадение ще настъпи, ако пакетът е създаден от процес в дадена сесийна група

unclean

Този експериментален модул трябва изрично да се зададе с '-m unclean' или '--match unclean'. Той извършва произволни проверки на пакетите. Този модул не е изследван и не би трябвало да се използва за повишаване на сигурността (той вероятно ще влоши нещата, тъй като е много възможно да съдържа грешки в кода). Не осигурява опции.

Условие за съвпадение State Най-полезните критерии за съвпадение се получават чрез използване на разширението ‘state’, което интерпретира анализа за проследяване на връзката (connection – tracking), извършван от модула ‘ip_conntrack’. Използването му е препоръчително.

Определението ‘-m state’ позволява допълнителната опция ‘--state’, с която се задава набор от състояния, разделени със запетая, проследяващи се за съвпадение (флагът ‘!’ указва несъвпадение с тези състояния). Състоянията са:

NEW – пакет, създаващ нова връзка;

ESTABLISHED – пакет, принадлежащ на съществуваща връзка, например пакет – отговор или изходящ пакет към връзка, за която са наблюдавани отговори;

RELATED – пакет, който е свързан, но не е част от съществуваща връзка, например ICMP грешка или (ако е зареден FTP модулът) пакет, установяващ връзка за FTP данни;

INVALID – пакет, който не може да бъде идентифициран по някаква причина: това включва недостиг на памет и ICMP грешки, които не са в отговор на някоя позната връзка. Обикновено тези пакети се отхвърлят.

7.4 Определения за целта

След като вече знаем какви изследвания могат да се правят върху пакети, ние се нуждаем от начин да укажем какво да се прави тези от тях, които съвпадат с дадено правило. Това се нарича **цел** на правилото.

Съществуват две много прости вградени цели: **DROP** и **ACCEPT**. Ние вече се срещнахме с тях: ако даден пакет отговаря на условията на някое правило и целта е една от двете, други правила не се изследват – ако целта е DROP, пакетът се отхвърля, а ако целта е ACCEPT, пакетът се приема.

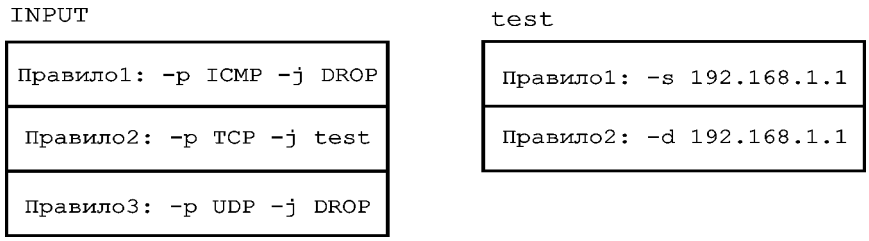
Съществуват още два типа цели, освен вградените – разширения и потребителско дефинирани вериги.

7.4.1 Потребителско дефинирани вериги

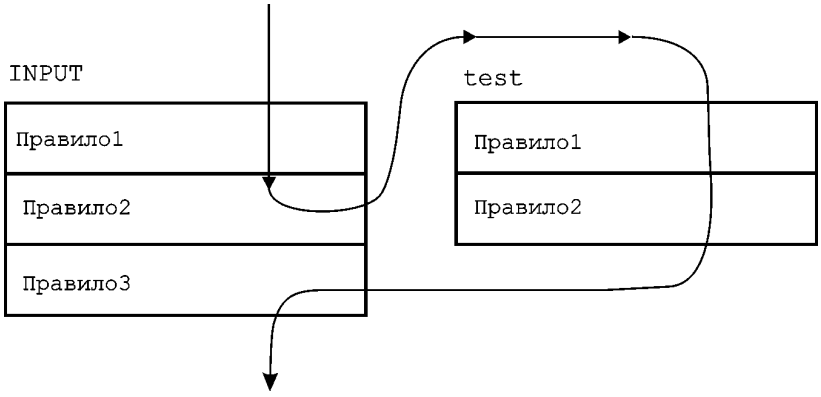
Една мощна характеристика на iptables е възможността потребителят сам да създава нови вериги, в допълнение към вградените (INPUT, OUTPUT и FORWARD). По определение потребителските вериги се задават с малки букви, за да ги различаваме (ще опишем как се създават нови потребителско дефинирани вериги в 7.5 (Операции върху цяла верига)).

Когато пакетът отговаря на някое правило, чиято цел е потребителска верига, той започва преминаване през правилата на тази верига. Ако тя не може да реши съдбата му, след като се достигне до края и, пакетът продължава движението си през следващото правило на текущата верига.

Да разгледаме две вериги: `INPUT` (вградената верига) и `test` (потребителско дефинирана верига).



Нека имаме TCP пакет, идващ от 192.168.1.1 и отиващ към 1.2.3.4. Той навлиза във веригата `INPUT` и се проверява от Правило 1 – няма съвпадение. За Правило 2 има съвпадение и целта е `test`, така че следващото проверявано правило е началото на `test`. За правило 1 от `test` има съвпадение, но то не определя цел, така че се изследва следващото правило – Правило 2. За него няма съвпадение, а ние стигаме до края на веригата. Връщаме се във веригата `INPUT`, където току що бяхме изследвали Правило 2. Сега изследваме Правило 3, за което също няма съвпадение. Пътят на пакетите е:



Потребителско дефинираните вериги могат да извършват скок към други потребителско дефинирани вериги (но не и да създават цикли: пакетите ще бъдат отхвърлени, ако попаднат в цикъл).

7.4.2 Разширения на iptables – нови цели

Другият тип цели са разширения. Разширенията на целите се състоят от модул към ядрото на ОС и опционално разширение на iptables за осигуряване на нови опции за командния ред. Има няколко разширения в подразбиращата се дистрибуция на netfilter:

LOG – осигурява възможност за водене на статистика от ядрото за пакети, отговарящи на дадено правило. Осигурява и допълнителните опции:

--log-level

следвана от номера на нивото или името му. Валидни имена са (без значение на регистъра на буквите): ‘debug’, ‘info’, ‘notice’, ‘warning’, ‘err’, ‘alert’ и ‘emerg’, отговарящи на номерата от 7 до 0. Виж man страницата на syslog.conf за обяснения на тези нива.

--log-prefix

следвана от стринг с дължина до 29 символа. Това съобщение се поставя в началото на log-записа, позволявайки той да бъде еднозначно разпознат.

REJECT – този модул има същият ефект като DROP, с изключение на това, че на източника на пакета се изпраща ICMP съобщение за грешка ‘port-unreachable’. ICMP съобщения за грешки не се изпращат ако:

- филтрираният пакет е бил ICMP съобщение за грешка или от непознат ICMP тип;
- филтрираният пакет е фрагмент без заглавна част;
- напоследък са забелязани твърде много ICMP съобщения за грешки от дадената посока.

7.4.3 Специални вградени цели

Има две специални вградени цели: **RETURN** и **QUEUE**

RETURN има същия ефект, както при достигане на края на веригата: за правило от вградена верига се изпълнява политиката на веригата. За правило от потребителска верига движението на пакета продължава през предната верига, веднага след правилото, довело до прескачане в тази верига.

QUEUE е специална верига, създаваща опашка от пакети чакащи обработка от потребителски процеси. За да бъде това полезно, са необходими два допълнителни компонента:

- “манипулатор на опашката”, който извършва истинската обработка на преминаващите пакети между ядрото и потребителското пространство; и
- потребителско приложение, което да приеме, вероятно да обработи и реши съдбата на пакетите.

Стандартният манипулатор на опашки за Ipv4 iptables е модулът ip_queue, който се разпространява заедно с ядрото и е маркиран като експериментален.

Следва бърз пример как да се използва iptables за създаване на опашка от пакети за потребителска обработка:

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

Благодарение на това правило локално генерираните изходящи ICMP пакети (създадени, например с ping) се подават към модула ip_queue, който трябва да ги достави на потребителското приложение. Ако няма такова приложение, което да очаква пакети, те се отхвърлят.

За написване на потребителски приложения използвайте libipq API-то. То се разпространява с iptables. Примерен код може да бъде намерен в набора с тестови инструменти (например redirect.c) от CVS.

Статусът на ip_queue може да се провери чрез `/proc/net/ip_queue`

Максималната дължина на опашката (т.е. броят пакети, доставяни към потребителското пространство без издадено решение за съдбата им) може да се контролира чрез:

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

Стойността по подразбиране за максималната дължина на опашката е 1024. След достигане на този лимит новите пакети ще се отхвърлят, докато дължината на опашката отново не падне под максимума. Добрите протоколи като TCP интерпретират отхвърлените пакети като насищане и вероятно ще спрат изпращането, когато

опашката се напълни. Може би, обаче, ще е нужно известно експериментиране, за да се определи идеалния максимум на дължината на опашката за дадена ситуация, ако стойността по подразбиране е твърде малка.

7.5 Операции върху цяла верига

Много полезна черта на iptables е възможността да групира свързаните едно с друго правила във вериги. Може да наречете тези вериги както искате, но ви препоръчвам използването на малки букви, за избягване на объркването с вградените вериги и цели. Имената на веригите могат да бъдат с дължина до 31 символа.

7.5.1 Създаване на нова верига

Да създадем нова верига. Тъй като имам голямо въображение, ще я нарека test. Използваме опциите '-N' или '--new-chain':

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

Толкова е просто. Сега можем да поставим правила, както бе описано по-горе.

7.5.2 Изтриване на верига

Изтриването на верига също е просто – използваме опциите '-X' или '--delete-chain'. Защо '-X'? Е, всички хубави букви вече бяха заети.

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

Има няколко ограничения при изтриване на вериги: те трябва да бъдат празни (виж 7.5.3 (Изпразване на вериги) по-долу) и не трябва да са цел на друга верига. Не може да изтриете и никоя от вградените вериги.

Ако не сте задали верига, то всички потребителско дефинирани вериги ще бъдат изтрити, ако това е възможно.

7.5.3 Изпразване на верига

Съществува прост начин за изпразване на верига, използвайки командата '-F' (или '--flush').

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

Ако не сте задали верига, всички вериги ще бъдат изпразнени

7.5.4 Преглеждане на верига

Можете да разгледате всички правила във верига, използвайки командата '-L' (или '--list').

Стойността 'refcnt', показвана за всяка потребителско дефинирана верига, е броят правила, имащи за цел тази верига. Тя трябва да бъде нула (и веригата да е празна), преди да може тази верига да бъде изтрита.

Ако се изпусне името на веригата, се преглеждат всички вериги, дори и празните.

Има три опции, които могат да акомпанират на '-L'. Опцията '-n' (numeric) е много полезна, тъй като предпазва iptables от опити за търсене на FQDN (напълно определените имена), които (ако вие използвате DNS, както повечето хора) ще доведе до големи закъснения, ако вашият DNS не е настроен правилно или вие филтрирате DNS заявките. Това води и до извеждане на TCP и UDP портовете като числа, а не като имена.

Опцията '-v' ви показва всички подробности на правилата, като пакетните и байтовите броячи, TOS сравненията и интерфейсите. В противен случай тези стойности се пропускат.

Трябва да се отбележи, че пакетните и байтовите броячи се извеждат, използвайки суфиксите 'K', 'M' или 'G' за 1 000, 1 000 000 и 1 000 000 000 съответно. Използването на флага '-x' (expand numbers) води до отпечатване на пълните цифри, без значение колко големи са те.

7.5.5 Нулиране на броячите

Полезно е да имате възможност за нулиране на броячите. Това може да стане с опцията '-Z' (или '--zero').

Разгледайте следното:

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

В този пример някои пакети могат да преминат между ‘-L’ и ‘-Z’ командите. Поради тази причина може да използвате ‘-L’ и ‘-Z’ заедно, за да нулирате броячите, докато ги прочитате.

7.5.6 Установяване на политика

Вече хвърлихме един поглед какво ще стане, когато пакет стигне края на вградена верига, когато дискутирахме как пакетът преминава през веригите. В този случай политиката на веригата определя съдбата на пакета. Само вградените вериги (INPUT, OUTPUT и FORWARD) имат политика, т.к. ако пакет попадне в края на потребителска верига, преминаването му се прехвърля към предната верига.

Политиките могат да бъдат ACCEPT или DROP, например:

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

8. Използване на ipchais и ipfwadm

В дистрибуцията на netfilter има модули ipchains.o и ipfwadm.o. Вмъкнете един от тях в ядрото (Забележка: те не са съвместими с iptables.o!). След това ще можете да използвате ipchains или ipfwadm, както в добрите стари дни.

Тази възможност ще се поддържа още известно време. Мисля, че подходящата формула е:

```
2*[съобщението за замяна – първото стабилно издание]
```

след датата, от която стабилното издание е налице.

Това означава, че за ipfwadm краят на поддръжката е:

```
2 * [Октомври 1997 (2.1.102 release) - Март 1995 (ipfwadm 1.0)] + Януари 1999 (2.2.0
release) = Март 2004.
```

За ipchains краят на поддръжката е (коригирайте ме с истинските дати):

```
2 * [Август 1999 (2.3.15 release) - Октомври 1997 (2.2.0 release)] + Юли 2000 (2.4.0
release?) = Март 2004.
```

Следователно, не трябва да се притеснявате до 2004.

9. Смесване на NAT и пакетно филтриране

Често срещано е желанието да се извършва транслация на мрежовите адреси (NAT) – виж NAT HOWTO и пакетно филтриране. Добрите новини са, че те се смесват страшно добре.

Създайте своя пакетен филтър, напълно игнорирайки какъвто и да е NAT, който извършвате. Началните и крайните адреси, “виждани” от пакетния филтър, ще бъдат “истинските” такива. Например, ако извършвате DNAT, за да прехвърляте всички връзки за 1.2.3.4 порт 80 към 10.1.1.1 порт 8080, пакетният филтър ще вижда пакети, отправени към 10.1.1.1 порт 8080 (реалната цел), а не към 1.2.3.4 порт 80. Аналогично, вие може да игнорирате маскарада: пакетите ще изглежда, че идват от техните реални вътрешни IP адреси (например 10.1.1.1), отговорите ще изглежда, че отиват обратно там.

Може да използвате разширението за съвпадение ‘state’ без да карате пакетният филтър да извършва допълнителна работа, тъй като NAT изисква проследяване на връзката. За подобряване на простия пример за маскарад в NAT HOWTO, целящ забрана на нови връзки през интерфейса ppp0, би трябвало да направите следното:

```
# Маскарад на ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

```
# Забрана на NEW И INVALID входящи или пренасочвани пакети 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

# Активиране на IP пренасочването
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10. Разлики между iptables и ipchains

Първо, имената на вградените вериги се изписват вече в горен регистър, т.к. INPUT и OUTPUT веригите приемат само пакети, насочени към локалната машина или генерирани от нея. Веригите се използват за преглед на всички входящи и изходящи пакети респективно.

- Флагът ‘-i’ сега означава входящ интерфейс и работи само с веригите INPUT и FORWARD. Правилата във веригите FORWARD или OUTPUT, използващи ‘-i’, трябва да се променят на ‘-o’.
- TCP и UDP портовете сега трябва да се задават със ‘--source-port’ или ‘--sport’ (или ‘--destination-port’/’--dport’) опциите и трябва да се поставят след ‘-p TCP’ или ‘-p UDP’ опциите, т.к. те зареждат TCP или UDP разширенията съответно.
- TCP флагът -y сега е --syn и трябва да бъде след ‘-p TCP’.
- Целта DENY сега е DROP
- Нулирането на единична верига, докато се преглеждат пакети, работи.
- Нулирането на вградените вериги също изчиства и броячите на политиката на веригите.
- Преглеждането на веригите ви дава също и броячите като моментна снимка.
- REJECT и LOG сега са разширени цели, което означава, че са отделни модули за ядрото.
- Имената на веригите могат да имат дължина до 31 символа.
- MASQ сега е MASQUERADE и се използва с различен синтаксис. REDIRECT запазва името си, но също има промяна в синтаксиса. Вижте NAT HOWTO за повече информация как да се конфигурират и двете.
- Опцията ‘-o’ вече не се използва за насочване на пакети към потребителско устройство (виж ’-i’ по-горе). Пакетите сега се препращат към потребителско устройство през целта QUEUE.
- Вероятно купища други неща, за които съм забравил.

11. Съвети за конфигуриране на пакетен филтър

Известна мъдрост в компютърната сигурност е да се блокира всичко, а след това да се отварят вратички, ако е нужно.

Това обикновено се дава с фразата: “Това, което не е изрично разрешено, е забранено”. Препоръчвам ви този подход, ако сигурността е вашата основна цел.

Не стартирайте услуги, от които нямате нужда, дори ако смятате, че сте блокирали достъпа до тях.

Ако конфигурирате машина само за firewall, в началото не стартирайте нищо и блокирайте всички пакети, а след това добавяйте услуги и позволявайте преминаването на пакети, когато това е нужно.

Препоръчвам ви задълбочаване на сигурността: използвайте tcp-wrappers (за връзка със самата firewall машина), proxy (за връзки, преминаващи през пакетния филтър), проверка на маршрута и пакетно филтриране. Проверката на маршрута (anti-spoofing) се състои в отхвърляне на пакети, идващи от неочаквани интерфейси: например, ако вътрешната ви мрежа има адреси 10.1.1.0/24 и пакет с такъв адрес на източника пристигне на вашия външен интерфейс, той ще бъде отхвърлен. За един интерфейс това може да се активира по следния начин (например за ppp0):

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

Или за всички съществуващи и бъдещи интерфейси като този:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
```

```
# echo 1 > $f
# done
#
```

При Debian това се извършва по подразбиране, когато е възможно. Ако имате асиметрично рутиране, (т.е. очаквате входящи пакети от странни посоки), то може би ще решите да забраните това филтриране за тези интерфейси.

Воденето на отчетни записи е полезно, ако настройвате firewall, за да разберете, ако нещо не работи, но при вече използващ се firewall винаги комбинирайте това с опцията 'limit', за да предотвратите наводняването на вашите log-файлове.

Горещо препоръчвам проследяването на връзките за системи, нуждаещи се от висока сигурност: това става с цената на допълнително натоварване, т.к. всички връзки се проследяват, но е много полезно за контролиране на достъпа до вашата мрежа. Ще трябва да заредите модула ip_conntrack.o, ако ядрото ви не зарежда модулите автоматично, и той не е вграден в ядрото. Ако искате акуратно да проследявате комплексни протоколи, ще трябва да заредите и подходящия помощен модул (например '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 ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP
# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Изграждането на добър firewall е извън обсега на това HOWTO, но моят съвет е "винаги бъдете минималисти". Вижте Security HOWTO за повече информация относно тестването на вашата машина.