

Iptables 指南 1.1.19

Oskar Andreasson

blueflux@koffein.net

Copyright © 2001–2003 by Oskar Andreasson

本文在符合 GNU Free Documentation 许可版本1.1的条件下，可以拷贝、分发、更改，但必须保留绪言和所有的章节，如印刷成书，封面要包括“原著：Oskar Andreasson”，且书背不准有文字。本文附录有“GNU Free Documentation License”的详细内容。

文中的所有脚本均置于GNU General Public License版本2下，可以自由地分发、更改。

给出这些脚本是希望它们有所作用，但没有任何保证，也没有商业可用性或某些特殊用途的内在保证。参见GNU General Public License

本文附带一份GNU General Public License，在章节“GNU Free Documentation License”中，如没有，请联系the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111– 1307 USA

献辞

首先，我要把本文献给我那wonderful的女友Nine1（她给我的帮助远远胜过我给她的）：希望我能让你幸福，就象你给我的。（译者注：我没有想到合适的词能表达作者女友的wonderful，你就自己想去吧。还有，不知他们现在是否结婚了:)）

其次，我要把这篇文章献给所有Linux的开发者和维护者，就是他们完成了令人无法相信的艰难工作，使这么优秀的操作系统成为可能。

目录

[译者序](#)

[关于作者](#)

[如何阅读](#)

[必备知识](#)

[本文约定](#)

1. [序言](#)

1.1. [为什么要写这个指南](#)

1.2. [指南是如何写的](#)

1.3. [文中出现的术语](#)

2. [准备阶段](#)

2.1. [哪里能取得iptables](#)

2.2. [内核配置](#)

2.3. [编译与安装](#)

2.3.1. [编译](#)

2.3.2. [在Red Hat 7.1上安装](#)

3. [表和链](#)

- 3.1. [概述](#)
 - 3.2. [mangle 表](#)
 - 3.3. [nat 表](#)
 - 3.4. [Filter 表](#)
- 4. [状态机制](#)
 - 4.1. [概述](#)
 - 4.2. [conntrack记录](#)
 - 4.3. [数据包在用户空间的状态](#)
 - 4.4. [TCP 连接](#)
 - 4.5. [UDP 连接](#)
 - 4.6. [ICMP 连接](#)
 - 4.7. [缺省的连接操作](#)
 - 4.8. [复杂协议和连接跟踪](#)
- 5. [保存和恢复数据管理规则](#)
 - 5.1. [速度](#)
 - 5.2. [restore的不足之处](#)
 - 5.3. [iptables-save](#)
 - 5.4. [iptables-restore](#)
- 6. [规则是如何练成的](#)
 - 6.1. [基础](#)
 - 6.2. [Tables](#)
 - 6.3. [Commands](#)
 - 6.4. [Matches](#)
 - 6.4.1. [通用匹配](#)
 - 6.4.2. [隐含匹配](#)
 - 6.4.3. [显式匹配](#)
 - 6.4.4. [针对非正常包的匹配](#)
 - 6.5. [Targets/Jumps](#)
 - 6.5.1. [ACCEPT target](#)
 - 6.5.2. [DNAT target](#)
 - 6.5.3. [DROP target](#)
 - 6.5.4. [LOG target](#)
 - 6.5.5. [MARK target](#)
 - 6.5.6. [MASQUERADE target](#)
 - 6.5.7. [MIRROR target](#)
 - 6.5.8. [QUEUE target](#)
 - 6.5.9. [REDIRECT target](#)
 - 6.5.10. [REJECT target](#)
 - 6.5.11. [RETURN target](#)
 - 6.5.12. [SNAT target](#)
 - 6.5.13. [TOS target](#)
 - 6.5.14. [TTL target](#)
 - 6.5.15. [ULOG target](#)
- 7. [防火墙配置实例 rc.firewall](#)
 - 7.1. [关于rc.firewall](#)
 - 7.2. [rc.firewall详解](#)
 - 7.2.1. [参数配置](#)
 - 7.2.2. [外部模块的装载](#)
 - 7.2.3. [proc的设置](#)
 - 7.2.4. [规则位置的优化](#)
 - 7.2.5. [缺省策略的设置](#)
 - 7.2.6. [自定义链的设置](#)
 - 7.2.7. [INPUT链](#)

- [7.2.8. FORWARD链](#)
 - [7.2.9. OUTPUT链](#)
 - [7.2.10. PREROUTING链](#)
 - [7.2.11. POSTROUTING链](#)
- 8. [例子简介](#)
 - 8.1. [rc.firewall.txt脚本的结构](#)
 - 8.1.1. [脚本结构](#)
 - 8.2. [rc.firewall.txt](#)
 - 8.3. [rc.DMZ.firewall.txt](#)
 - 8.4. [rc.DHCP.firewall.txt](#)
 - 8.5. [rc.UTIN.firewall.txt](#)
 - 8.6. [rc.test-iptables.txt](#)
 - 8.7. [rc.flush-iptables.txt](#)
 - 8.8. [Limit-match.txt](#)
 - 8.9. [Pid-owner.txt](#)
 - 8.10. [Sid-owner.txt](#)
 - 8.11. [Ttl-inc.txt](#)
 - 8.12. [Iptables-save ruleset](#)
- A. [常用命令详解](#)
 - A.1. [查看当前规则集的命令](#)
 - A.2. [修正和清空iptables的命令](#)
- B. [常见问题于与解答](#)
 - B.1. [模块装载问题](#)
 - B.2. [未设置SYN的NEW状态包](#)
 - B.3. [NEW状态的SYN/ACK包](#)
 - B.4. [使用私有IP地址的ISP](#)
 - B.5. [放行DHCP数据](#)
 - B.6. [关于mIRC DCC的问题](#)
- C. [ICMP类型](#)
- D. [其他资源和链接](#)
- E. [鸣谢](#)
- F. [History](#)
- G. [GNU Free Documentation License](#)
 - 0. [PREAMBLE](#)
 - 1. [APPLICABILITY AND DEFINITIONS](#)
 - 2. [VERBATIM COPYING](#)
 - 3. [COPYING IN QUANTITY](#)
 - 4. [MODIFICATIONS](#)
 - 5. [COMBINING DOCUMENTS](#)
 - 6. [COLLECTIONS OF DOCUMENTS](#)
 - 7. [AGGREGATION WITH INDEPENDENT WORKS](#)
 - 8. [TRANSLATION](#)
 - 9. [TERMINATION](#)
 - 10. [FUTURE REVISIONS OF THIS LICENSE](#)
 - [How to use this License for your documents](#)
- H. [GNU General Public License](#)
 - 0. [Preamble](#)
 - 1. [TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION](#)
 - 2. [How to Apply These Terms to Your New Programs](#)
- I. [示例脚本的代码](#)
 - I.1. [rc.firewall脚本代码](#)
 - I.2. [rc.DMZ.firewall脚本代码](#)
 - I.3. [rc.UTIN.firewall脚本代码](#)

- I. 4. [rc.DHCP.firewall脚本代码](#)
- I. 5. [rc.flush-iptables脚本代码](#)
- I. 6. [rc.test-iptables脚本代码](#)

List of Tables

- 3-1. [以本地为目标（就是我们自己的机子了）的包](#)
- 3-2. [以本地为源的包](#)
- 3-3. [被转发的包](#)
- 4-1. [数据包在用户空间的状态](#)
- 4-2. [内部状态](#)
- 6-1. [Tables](#)
- 6-2. [Commands](#)
- 6-3. [Options](#)
- 6-4. [Generic matches](#)
- 6-5. [TCP matches](#)
- 6-6. [UDP matches](#)
- 6-7. [ICMP matches](#)
- 6-8. [Limit match options](#)
- 6-9. [MAC match options](#)
- 6-10. [Mark match options](#)
- 6-11. [Multiport match options](#)
- 6-12. [Owner match options](#)
- 6-13. [State matches](#)
- 6-14. [TOS matches](#)
- 6-15. [TTL matches](#)
- 6-16. [DNAT target](#)
- 6-17. [LOG target options](#)
- 6-18. [MARK target options](#)
- 6-19. [MASQUERADE target](#)
- 6-20. [REDIRECT target](#)
- 6-21. [REJECT target](#)
- 6-22. [SNAT target](#)
- 6-23. [TOS target](#)
- 6-24. [TTL target](#)
- 6-25. [ULOG target](#)
- C-1. [ICMP类型](#)

译者序

译者sllscn是[中国Linux公社](#)里的“Linux 新鲜社员”，一个Linux爱好者，在实际工作中使用iptables构造防火墙时，发现有关iptables的中文资料太少，故而不得已参考英文版的材料。为了今后参考的方便，也为了广大使用者，不怕自己的英文水平太差，翻着字典翻译了本文。翻译只为了能看懂，达不到“好看”，勿怪！

第一章序言部分除了第三小节介绍的术语要看看，其他都没什么。第二章对想要亲自编译iptables的兄弟们是有些帮助的。第三、第四两章可以使我们理解、掌握iptables工作方式和流程。第五章和第六章是iptables命令使用方法的详细介绍。第七章与第八章是实例讲解，对我们编写自己的规则很有指导意义的，强烈建议你看一看。附录里有一些资源链接是很好的，相信你一定会喜欢。

因为术语的缘故，目录部分有一些未翻译，但正文的内容都翻译了。附录F是本文的更新历史，附录G是GNU Free Documentation License，附录H是GNU General Public License，它们对理解 iptables没有什么作用，故未翻译。

在阅读本文时，你可能会发现有重复的地方，这不是原作者的水平不高，反而恰恰是他为我们考虑的结果。你可以把这篇文章的任何一章抽出来阅读，而不需要反复地参照其他章节。在此，再次向作者表示敬意！

因译者水平有限，对原文的理解不敢保证完全正确，如有意见或建议，可以联系译者 slcl@sohu.com

郑重声明：翻译得到了原文作者Oskar Andreasson的许可。对于本文（不是原文），可自由使用、修改、传播、转载，但对以盈利为目的使用，保留所有权利。

关于作者

我的局域网里有很多“年老的”计算机，他们也想连接到Internet上，还要保证安全。做到这一点，iptables是的ipchains的一个很好的升级。使用ipchains你可以通过丢弃所有“目的端口不是特定端口”的包来建立一个安全的网络。但这将导致一些服务出现问题，比如被动FTP，还有在IRC中流出的DCC。它们在服务器上分配端口，并告知客户端，然后再让客户连接。但是，iptables的代码中也有一些小毛病，在某些方面我发现这些代码并没有为作为完整的产品发布做好准备，但我仍然建议使用ipchains或更老的 ipfwadm 的人们进行升级，除非他们对正在使用的代码满意，或它们足以满足他们的需要。

如何阅读

本文介绍了iptables，以便你可以领会iptables的精彩，文中不包含iptables或Netfilter在安全方面的 bug。如果你发现iptables（或其组成部分）任何bug或特殊的行为，请联系 Netfilter mailing lists，他们会告诉你那是否是bug或如何解决。iptables或Netfilter中几乎没有安全方面的bug，当然偶尔也会出些问题，它们能在[Netfilter主页](#)中找到。

文中用到的脚本不能解决Netfilter内部的bug，给出它们，只是为了演示如何构造规则，以便我们能解决遇到的数据流管理问题。但本文没有包括象“如何关闭HTTP端口，因为Apache 1.2.12偶尔会被攻击”这样的问题。本指南会告诉你如何通过iptables关闭HTTP端口，但不是因为Apache偶尔会被攻击。

本文适合于初学者，但也尽可能完善。因为有太多的targets或matches，所以没有完全收录。如果你需要这方面的信息，可以访问[Netfilter主页](#)。

必备知识

阅读本文，要具备一些基础知识，如Linux/Unix，shell脚本编写，内核编译，最好还有一些简单的内核知识。

我尝试着尽可能使读者不需要这些知识也能完全看懂这篇文章, 但要理解扩展部分是不行的。所以还是要有点基础的:)

本文约定

以下的约定会在文中用到:

- 代码和命令输出使用定宽字体, 命令用粗体。

```
[blueflux@work1 neigh]$ ls
default  eth0  lo
[blueflux@work1 neigh]$
```

- 所有的命令和程序名都用粗体。
 - 所有的系统部件, 如硬件、内核部件、loopback使用*斜体*。
 - 计算机文本输出用 *这种字体*。
 - 文件名和路径名象这样 /usr/local/bin/iptables 。
-

1. 序言

1.1. 为什么要写这个指南

我发现目前所有的HOWTO都缺乏Linux 2.4.x 内核中的Iptables和Netfilter 函数的信息, 于是我试图回答一些问题, 比如状态匹配。我会用插图和例子 [*rc.firewall.txt*](#) 加以说明, 此处的例子可以在你的/etc/rc.d/使用。最初这篇文章是以HOWTO文档的形式书写的, 因为许多人只接受HOWTO文档。

还有一个小脚本[*rc.flush-iptables.txt*](#), 我写它只是为使你在配置它的时候能象我一样有成功的感觉。

1.2. 指南是如何写的

我请教了Marc Boucher 及netfilter团队的其他核心成员。对他们的工作以及对我在为boingworld.com 书写这个指南时的帮助表示极大的谢意, 现在这个指南在我自己的站点frozentux.net上进行维护。这个文档将一步一步教你setup过程, 让你对iptables包有更多的了解。这大部分的东西都基于例子rc.firewall 文件, 因为我发现这是学习iptables的一个好方法。我决定自顶向下地跟随rc.firewall 文件来学习 iptables。虽然这样会困难一些, 但更有逻辑。当你碰到不懂的东西时再来查看这个文件。

1.3. 文中出现的术语

文中包含了一些术语，你应该有所了解。这里有一些解释，并说明了本文中如何使用它们。

DNAT - Destination Network Address Translation 目的网络地址转换。DNAT是一种改变数据包目的ip地址的技术，经常和SNAT联用，以使多台服务器能共享一个ip地址连入Internet，并且继续服务。通过对同一个ip地址分配不同的端口，来决定数据的流向。

Stream - 流 是指发送和接收的数据包和通信的双方都有关系的一种连接（译者注：本文中，作者把连接看作是单向的，流表示双向的连接）。一般的，这个词用于描述在两个方向上发送两个或三个数据包的连接。对于TCP，流意味着连接，它发送了一个SYN，然后又回复SYN/ACK。但也可能是指这样的连接，发送一个SYN，回复ICMP主机不可达信息。换句话说，我使用这个词很随意。

SNAT - Source Network Address Translation源网络地址转换。这是一种改变数据包源ip地址的技术，经常用来使多台计算机分享一个Internet地址。这只在IPv4中使用，因为IPv4的地址已快用完了，IPv6将解决这个问题。

State - 状态 指明数据包处于什么状态。状态在[*RFC 793 - Transmission Control Protocol*](#)中定义，或由用户在Netfilter/iptables中自定义。需要注意的是Netfilter设定了一些关于连接和数据包的状态，但没有完全使用使用RFC 793的定义。

User space - 用户空间，指在内核外部或发生在内核外部的任何东西。例如，调用 `iptables -h` 发生在内核外部，但 `iptables -A FORWARD -p tcp -j ACCEPT`（部分地）发生在内核内部，因为一条新的规则加入了规则集。

Kernel space - 内核空间，与用户空间相对，指那些发生在内核内部。

Userland - 参见用户空间

target - 这个词在后文中有大量的应用，它表示对匹配的数据包所做的操作。

2. 准备阶段

这一章是学习iptables的开始，它将帮助你理解Netfilter和iptables在Linux中扮演的角色。它会告诉你如何配置、安装防火墙，你的经验也会随之增长。当然，要想达到你的目标，是要花费时间，还要有毅力。（译者注：听起来很吓人的:)）

2.1. 哪里能取得iptables

iptables 可以从www.netfilter.org 下载，网站中的FAQs也是很好的教程。iptables 也使用一些内核空间，可以在用make configure配置内核的过程中配置，下面会介绍必要的步骤。

2.2. 内核配置

为了运行iptables, 需要在内核配置期间, 选择以下一些选项, 不管你用**make config**或其他命令。

CONFIG_PACKET - 允许程序直接访问网络设备 (译者注: 最常用的就是网卡了), 象tcpdump和snort就要使用这个功能。



严格地说, iptables并不需要CONFIG_PACKET, 但是它有很多用处 (译者注: 其他程序需要), 所以就选上了。当然, 你不想要, 不选就是了。 (译者注: 建议还是选的为好)

CONFIG_NETFILTER - 允许计算机作为网关或防火墙。这个是必需的, 因为整篇文章都要用到这个功能。我想你也需要这个, 谁叫你学iptables呢:)

当然, 你要给网络设备安装正确的驱动程序, 比如, Ethernet 网卡, PPP 还有 SLIP。上面的选项, 只是在内核中建立了一个框架, iptables确实已经可以运行, 但不能做任何实质性的工作。我们需要更多的选项。以下给出内核2.4.9的选项和简单的说明:

CONFIG_IP_NF_CONNTRACK - 连接跟踪模块, 用于 NAT (网络地址转换) 和 Masquerading (ip地址伪装), 当然, 还有其他应用。如果你想把LAN中的一台机器作为防火墙, 这个模块你算选对了。脚本[rc.firewall.txt](#) 要想正常工作, 就必需有它的存在。

CONFIG_IP_NF_FTP - 这个选项提供针对FTP连接进行连接跟踪的功能。一般情况下, 对FTP连接进行连接跟踪是很困难的, 要做到这一点, 需要一个名为helper的动态链接库。此选项就是用来编译helper的。如果没有这个功能, 就无法穿越防火墙或网关使用FTP。

CONFIG_IP_NF_IPTABLES - 有了它, 你才能使用过滤、伪装、NAT。它为内核加入了iptables标识框架。没有它, iptables毫无作用。

CONFIG_IP_NF_MATCH_LIMIT - 此模块并不是十分必要, 但我在例子[rc.firewall.txt](#)中用到了。它提供匹配LIMIT的功能, 以便于使用一个适当的规则来控制每分钟要匹配的数据包的数量。比如, `-m limit --limit 3/minute` 的作用是每分钟最多匹配三个数据包。这个功能也可用来消除某种DoS攻击。

CONFIG_IP_NF_MATCH_MAC - 选择这个模块, 可以根据MAC地址匹配数据包。例如, 我们想要阻塞使用了某些MAC地址的数据包, 或阻塞某些计算机的通信, 用这个很容易。因为每个Ethernet网卡都有它自己的MAC地址, 且几乎从不会改变。但我在[rc.firewall.txt](#)中没有用到这个功能, 其他例子也未用到。 (译者注: 这又一次说明了学习是为将来打基础:))

CONFIG_IP_NF_MATCH_MARK - 这个选项用来标记数据包。对数据包做 MARK (标记) 操作, 我们就可以在后面的表中用这个标记来匹配数据包。后文有详细的说明。

CONFIG_IP_NF_MATCH_MULTIPORT - 选择这个模块我们可以使用端口范围来匹配数据包, 没有它, 是无法做到这一点的。

CONFIG_IP_NF_MATCH_TOS - 使我们可以设置数据包的TOS (Type Of Service 服务类型)。这个工作也可以用命令ip/tc完成, 还可在mangle表中用某种规则设定。

CONFIG_IP_NF_MATCH_TCPMSS - 可以基于MSS匹配TCP数据包。

CONFIG_IP_NF_MATCH_STATE - 相比较**ipchains** 这是最大的更新, 有了它, 我们可以对数据包做状态匹配。比如, 在某个TCP连接的两个方向上已有通信, 则这个连接上的数据包就被看作**ESTABLISHED** (已建立连接) 状态。在[rc.firewall.txt](#) 里大量使用了此模块的功能。

CONFIG_IP_NF_MATCH_UNCLEAN - 匹配那些不符合类型标准或无效的 P、TCP、UDP、ICMP数据包 (译者注: 之所以此模块名为UNCLEAN, 可以这样理解, 凡不是正确模式的包都是脏的。这有些象操作系统内存管理中的“脏页”, 那这里就可以称作“脏包”了, 自然也就UNCLEAN了)。我们一般丢弃这样的包, 但不知这样做是否正确。另外要注意, 这种匹配功能还在实验阶段, 可能会有些问题。

CONFIG_IP_NF_MATCH_OWNER - 根据套接字的拥有者匹配数据包。比如, 我们只允许root访问Internet。在**iptables**中, 这个模块最初只是用一个例子来说明它的功能。同样, 这个模块也处于实验阶段, 还无法使用。

CONFIG_IP_NF_FILTER - 这个模块为**iptables**添加基本的过滤表, 其中包含INPUT、FORWARD、OUTPUT链。通过过滤表可以做完全的IP过滤。只要想过滤数据包, 不管是接收的还是发送的, 也不管做何种过滤, 都必需此模块。

CONFIG_IP_NF_TARGET_REJECT - 这个操作使我们用ICMP错误信息来回应接收到的数据包, 而不是简单地丢弃它。有些情况必须要有回应的, 比如, 相对于ICMP和UDP来说, 要重置或拒绝TCP连接总是需要一个TCP RST包。

CONFIG_IP_NF_TARGET_MIRROR - 这个操作使数据包返回到发送它的计算机。例如, 我们在INPUT链里对目的端口为HTTP的包设置了MIRROR操作, 当有人访问HTTP时, 包就被发送回原计算机, 最后, 他访问的可能是他自己的主页。(译者注: 应该不难理解为什么叫做MIRROR了)

CONFIG_IP_NF_NAT - 顾名思义, 本模块提供NAT功能。这个选项使我们有权访问nat表。端口转发和伪装是必需此模块的。当然, 如果你的LAN里的所有计算机都有唯一的有效的 IP地址, 那在做防火墙或伪装时就无须这个选项了。[rc.firewall.txt](#) 是需要的:)

CONFIG_IP_NF_TARGET_MASQUERADE - 提供MASQUERADE (伪装) 操作。如果我们不知道连接Internet的IP, 首选的方法就是使用MASQUERADE, 而不是DNAT或SNAT。换句话说, 就是如果我们使用PPP或SLIP等连入Internet, 由DHCP或其他服务分配IP, 使用这个比SNAT好。因为MASQUERADE 不需要预先知道连接Internet的IP, 虽然对于计算机来说MASQUERADE要比NAT的负载稍微高一点。

CONFIG_IP_NF_TARGET_REDIRECT - 这个操作和代理程序一起使用是很有用的。它不会让数据包直接通过, 而是把包重新映射到本地主机, 也就是完成透明代理。

CONFIG_IP_NF_TARGET_LOG - 为**iptables**增加 **LOG** (日志) 操作。通过它, 可以使用系统日志服务记录某些数据包, 这样我们就能了解在包上发生了什么。这对于我们做安全审查、调试脚本的帮助是无价的。

CONFIG_IP_NF_TARGET_TCPMSS - 这个选项可以对付一些阻塞ICMP分段信息的ISP (服务提供商) 或服务。没有ICMP分段信息, 一些网页、大邮件无法通过, 虽然小邮件可以, 还有, 在握手完成之后, ssh可以但scp不能工作。我们可以用TCPMSS解决这个问题, 就是使MSS (Maximum Segment Size) 被钳制于PMTU (Path Maximum Transmit Unit)。这个方法可以处理被Netfilter开发者们在内核配置帮助中称作“criminally brain-dead ISPs or servers”的问题。

CONFIG_IP_NF_COMPAT_IPCHAINS - **ipchains** 的, 这只是为内核从2.2转换到2.4而使用的, 它会在2.6中删除。

CONFIG_IP_NF_COMPAT_IPFWADM – 同上, 这只是 **ipfwadm**的暂时使用的兼容模式。

上面, 我简要介绍了很多选项, 但这只是内核2.4.9中的。要想看看更多的选项, 建议你去 Netfilter 看看patch-o-matic。在那里, 有其他的一些选项。POM可能会被加到内核里, 当然现在还没有。这有很多原因, 比如, 还不稳定, Linus Torvalds没打算或没坚持要把这些补丁放入主流的内核, 因为它们还在实验。

把以下选项编译进内核或编译成模块, [*rc.firewall.txt*](#)才能使用。

- CONFIG_PACKET
- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_FTP
- CONFIG_IP_NF_IRC
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_TARGET_LOG
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_TARGET_MASQUERADE

以上是为保证 [*rc.firewall.txt*](#)正常工作而需要的最少的选项。其他脚本需要的选项, 在相应的章节里都有说明。目前, 我们只需注意要学习的这个脚本。

2.3. 编译与安装

下面, 我们来看看如何编译**iptables**。iptables很多组件的配置、编译是与内核的配置、编译相关联的, 了解这一点是很重要的。某些Linux产品预装了**iptables**, 比如Red Hat, 但是它的缺省设置是不启用iptables的。后文我们会介绍如何启用它, 也会介绍一下其他 Linux 产品里的iptables情况。

2.3.1. 编译

首先要解压iptables包。这里, 我用iptables 1.2.6a做例子 (译者注: 在我翻译时, 最新版本已经是 1.2.9, 其中又有了不少改进, 修补了一些bug, 增添了几个match和target。)。

命令 `bzip2 -cd iptables-1.2.6a.tar.bz2 | tar -xvf -` (当然也可以用 `tar -xjvf iptables-1.2.6a.tar.bz2`, 但这个命令可能对一些老版的 `tar` 不适用) 将压缩包解压至目录 `iptables-1.2.6a`, 其中的 `INSTALL` 文件有很多对编译、运行有用的信息。

这一步, 你将配置、安装一些额外的模块, 也可以为内核增加一些选项。我们这里只是检查、安装一些未被纳入内核的标准的补丁。当然, 更多的在实验阶段的补丁, 仅在进行其他某些操作时才会用到。



有一些补丁仅仅处在实验阶段, 把它们也安装上不是一个好主意。这一步, 你会遇到很多十分有趣的匹配和对数据包的操作, 但它们还正在实验。

为了完成这一步, 我们要在 `iptables` 的目录内用到如下一些命令:

```
make pending-patches KERNEL_DIR=/usr/src/linux/
```

变量 `KERNEL_DIR` 指向内核原码的真实路径。一般情况下, 都是 `/usr/src/linux/`, 但也会不一样, 这要看你所用的 Linux 产品了。



总之, 只有某些补丁会被询问是否加入内核, 而 Netfilter 的开发者们有大量的补丁或附件想要加入内核, 但还要再实验一阵子才能做到。如果你想安装这些东西, 就用下面的命令:

```
make most-of-pom KERNEL_DIR=/usr/src/linux/
```

这个命令会安装部分 `patch-o-matic` (netfilter 世界对补丁的称呼), 忽略掉的是非常极端的那一部分, 它们可能会对内核造成严重的破坏。你要知道这个命令的作用, 要了解它们对内核原码的影响, 好在在你选用之前, 会有所提示。下面的命令可以安装所有的 `patch-o-matic` (译者注: 一定要小心哦)。

```
make patch-o-matic KERNEL_DIR=/usr/src/linux/
```

要仔细的读读每一个补丁的帮助文件, 因为有些 `patch-o-matic` 会损坏内核, 而有些对其他补丁有破坏作用。



你要是不打算用 `patch-o-matic` 修补内核, 以上的命令都用不着, 它们不是必需的。不过, 你可以用这些命令来看看有什么有趣的玩意儿, 这不会影响任何东西。

安装好 `patch-o-matic`, 现在应该重新编译内核了, 因为其中增加了一些补丁。但别忘了重新配置内核, 现有的配置文件里可没有你增加的补丁的信息。当然, 你也可以先编译 `iptables`, 再来编译内核。

接下来就该编译 `iptables` 了, 用下面这个简单的命令:

```
make KERNEL_DIR=/usr/src/linux/
```

`iptables` 应该编译好了, 如果不行, 好好考虑考虑问题在哪儿, 要么订阅 [Netfilter mailing list](#), 那里可能有人能帮助你。

一切顺利的话, 我们该安装 `iptables` 了, 这几乎不会有问题的。我们用下面的命令来完成这一步:

```
make install KERNEL_DIR=/usr/src/linux/
```

现在大功告成了。如果你在前面没有重新编译、安装内核，现在就要做了，不然，你还是不能使用更新后的iptables。好好看看INSTALL吧，那里面有详细的安装信息。

2.3.2. 在Red Hat 7.1上安装

Red Hat 7.1使用2.4.x的内核，支持Netfilter和iptables。Red Hat包含了所有基本的程序和需要的配置文件，但缺省使用的是B class=COMMAND>ipchains。“iptables为什么不能用”是最常见的问题，下面就让我们来说一说如何关闭ipchains而起用iptables。



Red Hat 7.1预装的iptables版本有些老了，在使用之前，你可能想装个新的，再自己编译一下内核。

我们先要关闭ipchains，并且不想再让它运行起来，做到这一点，要更改目录/etc/rc.d/下的一些文件名。用以下命令完成：

```
chkconfig --level 0123456 ipchains off
```

这个命令把所有指向/etc/rc.d/init.d/ipchains的软连接改名为 K92ipchains。以S开头表示，在启动时会由初始化脚本运行此脚本。改为K开头后，就表示终止服务，或以后在启动时不再运行。这样，ipchains以后不会再次开机就运行了。

要想终止正在运行的服务，要用service命令。终止ipchains 服务的命令是：

```
service ipchains stop
```

现在，我们可以启动iptables服务了。首先，要确定在哪个运行层运行，一般是 2, 3和5，这些层有不同的用处：

- 2. 不带NFS的多用户环境，和层3的区别仅在于不带网络支持。
- 3. 多用户环境，就是我们一般事用的层。
- 5. X11，图形界面。

用下面的命令以使iptables能在这些层运行：

```
chkconfig --level 235 iptables on
```

你也可以使用这个命令使iptables能在其他层运行。但没这个必要，因为层1是单用户模式，一般用在维修上；层4保留不用；层6用来关闭计算机。

启动iptables用：

```
service iptables start
```

在脚本iptables里还没有定义规则。在Red Hat 7.1中添加规则的方法有二：第一个方法是编辑/etc/rc.d/init.d/iptables，要注意在用RPM升级iptables时，已有的规则可能会被删除。另

一个方法是先装载规则，然后用命令**iptables-save**把规则保存到文件中，再由目录rc.d下的脚本（/etc/rc.d/init.d/iptables）自动装载。

我们先来说明如何利用“剪切粘贴大法”设置/etc/rc.d/init.d/iptables。为了能在计算机启动iptables时装载规则，可以把规则放在“start)”节或函数start()中。注意：如果把规则放在“start)”节里，则不要在“start)”节里运行start()，还要编辑“stop)”节，以便在关机时或进入一个不需要iptables的层时，脚本知道如何处理。还应检查“restart”节和“condrestart”节的设置。一定要注意，我们所做的改动在升级iptables时可能会被删除，而不管是通过Red Hat网络自动升级还是用RPM升级。

下面介绍第二种方法：先写一个规则的脚本，或直接用iptables命令生成规则。规则要适合自己的需要，别忘了实验一下是否有问题，确认正常之后，使用命令**iptables-save**来保存规则。一般用**iptables-save > /etc/sysconfig/iptables**生成保存规则的文件 /etc/sysconfig/iptables，也可以用**service iptables save**，它能把规则自动保存在/etc/sysconfig/iptables中。当计算机启动时，rc.d下的脚本将用命令**iptables-restore**调用这个文件，从而就自动恢复了规则。

以上两种方法最好不要混用，以免用不同方法定义的规则互相影响，甚至使防火墙的设置无效。

至此，可以删除预装的**ipchains**和**iptables**了，这样可以避免新旧版本的**iptables**之间的冲突。其实，只有当你从原码安装时，才需要这样做。但一般来说，也不会出现互相影响的问题，因为基于rpm的包不使用原码的缺省目录。删除用以下命令：

```
rpm -e iptables
```

既然不用**ipchains**为什么要保留呢？删吧！命令如下：

```
rpm -e ipchains
```

历经磨难，胜利终于到来了。你已经能够从源码安装iptables了。那些老版的东西就删掉吧。

Chapter 3. 表和链

这一章我们来讨论数据包是以什么顺序、如何穿越不同的链和表的。稍后，在你自己写规则时，就会知道这个顺序是多么的重要。一些组件是iptables与内核共用的，比如，数据包路由的判断。了解到这一点是很重要的，尤其在你用iptables改变数据包的路由时。这会帮助你弄明白数据包是如何以及为什么被那样路由，一个好的例子是**DNAT**和**SNAT**，不要忘了TOS的作用。

3.1. 概述

当数据包到达防火墙时，如果MAC地址符合，就会由内核里相应的驱动程序接收，然后会经过一系列操作，从而决定是发送给本地的程序，还是转发给其他机器，还是其他的什么。

我们先来看一个以本地为目的的数据包，它要经过以下步骤才能到达要接收它的程序：

下文中有个词mangle，我实在没想到什么合适的词来表达这个意思，只因为我的英语太差！我只能把我理解的写出来。这个词表达的意思是，会对数据包的一些传输特性进行修改，在mangle表中允许的操作是 TOS、TTL、MARK。也就是说，今后只要我们见到这个词能理解它的作用就行了。

Table 3-1. 以本地为目标（就是我们自己的机子了）的包

Step (步骤)	Table (表)	Chain (链)	Comment (注释)
1			在线路上传输(比如, Internet)
2			进入接口 (比如, eth0)
3	mangle	PREROUTING	这个链用来mangle数据包, 比如改变TOS等
4	nat	PREROUTING	这个链主要用来做DNAT。不要在这个链做过虑操作, 因为某些情况下包会溜过去。
5			路由判断, 比如, 包是发往本地的, 还是要转发的。
6	mangle	INPUT	在路由之后, 被送往本地程序之前, mangle数据包。
7	filter	INPUT	所有以本地为目的的包都要经过这个链, 不管它们从哪儿来, 对这些包的过滤条件就设在这里。
8			到达本地程序了(比如, 服务程序或客户程序)

注意, 相比以前(译者注: 就是指ipchain) 现在数据包是由INPUT链过, 而不是FORWARD链。这样更符合逻辑。刚看上去可能不太好理解, 但仔细想想就会恍然大悟的。

现在我们来看看源地址是本地器的包要经过哪些步骤:

Table 3-2. 以本地为源的包

Step	Table	Chain	Comment
1			本地程序 (比如, 服务程序或客户程序)
2			路由判断, 要使用源地址, 外出接口, 还有其他一些信息。
3	mangle	OUTPUT	在这儿可以mangle包。建议不要在这儿做过滤, 可能有副作用哦。
4	nat	OUTPUT	这个链对从防火墙本身发出的包进行DNAT操作。
5	filter	OUTPUT	对本地发出的包过滤。
6	mangle	POSTROUTING	这条链主要在包DNAT之后(译者注: 作者把这一次DNAT称作实际的路由, 虽然在前面有一次路由。对于本地的包, 一旦它被生成, 就必须经过路由代码的处理, 但这个包具体到哪儿去, 要由NAT代码处理之后才能确定。所以把这称作实际的路由。), 离开本地之前, 对包mangle。有两种包会经过这里, 防火墙所在本子本身产生的包, 还有被转发的包。

7	nat	POSTROUTING	在这里做SNAT。但不要在这里做过滤，因为有副作用，而且有些包是会溜过去的，即使你用了DROP策略。
8			离开接口(比如: eth0)
9			在线路上传输(比如, Internet)

在这个例子中，我们假设一个包的目的是另一个网络中的一台机器。让我们来看看这个包的旅程：

Table 3-3. 被转发的包

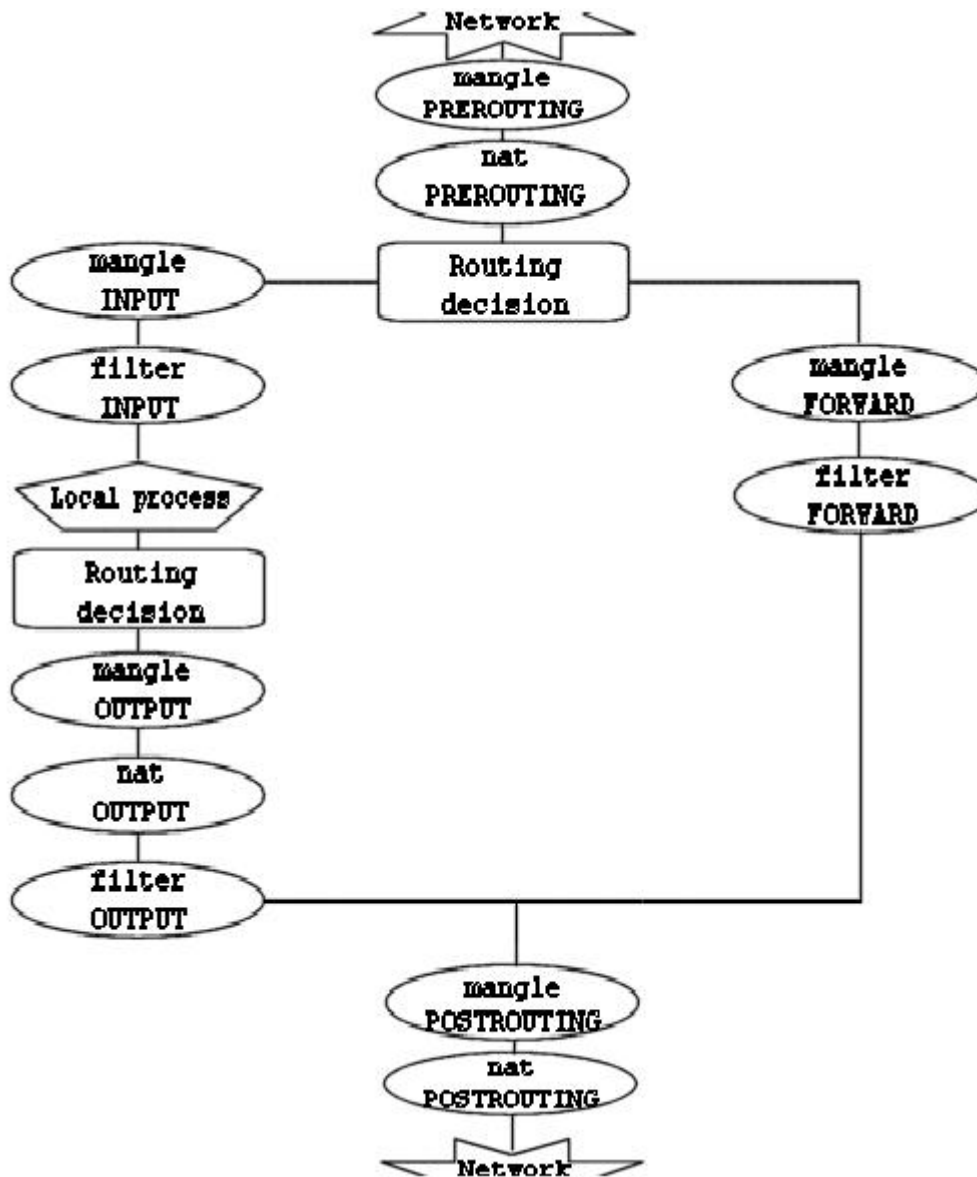
Step	Table	Chain	Comment
1			在线路上传输(比如, Internet)
2			进入接口(比如, eth0)
3	mangle	PREROUTING	mangle数据包, , 比如改变TOS等。
4	nat	PREROUTING	这个链主要用来做DNAT。不要在这个链做过滤操作，因为某些情况下包会溜过去。稍后会做SNAT。
5			路由判断，比如，包是发往本地的，还是要转发的。
6	mangle	FORWARD	包继续被发送至mangle表的FORWARD链，这是非常特殊的情况才会用到的。在这里，包被mangle（还记得mangle的意思吗）。这次mangle发生在最初的路由判断之后，在最后一次更改包的目的之前（译者注：就是下面的FORWARD链所做的，因其过滤功能，可能会改变一些包的目的地址，如丢弃包）。
7	filter	FORWARD	包继续被发送至这条FORWARD链。只有需要转发的包才会走到这里，并且针对这些包的所有过滤也在这里进行。注意，所有要转发的包都要经过这里，不管是外网到内网的还是内网到外网的。在你自己书写规则时，要考虑到这一点。
8	mangle	POSTROUTING	这个链也是针对一些特殊类型的包（译者注：参考第6步，我们可以发现，在转发包时，mangle表的两个链都用在特殊的应用上）。这一步mangle是在所有更改包的目的地址的操作完成之后做的，但这时包还在本地上。
9	nat	POSTROUTING	这个链就是用来做SNAT的，当然也包括Masquerade（伪装）。但不要在这儿做过滤，因为某些包即使不满足条件也会通过。
10			离开接口(比如: eth0)
11			又在线路上传输了(比如, LAN)

就如你所见的，包要经历很多步骤，而且它们可以被阻拦在任何一条链上，或者是任何有问题的地方。我们的主要兴趣是iptables的概貌。注意，对不同的接口，是没有什么特殊的链和表的。所有要经防火墙/ 路由器转发的包都要经过FORWARD链。



在上面的情况里，不要在INPUT链上做过滤。INPUT是专门用来操作那些以我们的机器为目的地址的包的，它们不会被路由到其它地方的。

现在，我们来看看在以上三种情况下，用到了哪些不同的链。图示如下：



要弄清楚上面的图，可以这样考虑。在第一个路由判断处，不是发往本地的包，我们会发送它穿过 FORWARD链。若包的目的地是本地监听的IP地址，我们就会发送这个包穿过INPUT链，最后到达本地。

值得注意的是，在做NAT的过程中，发往本机的包的目的地址可能会在PREROUTING链里被改变。这个操作发生在第一次路由之前，所以在地址被改变之后，才能对包进行路由。注意，所有的包都会经过上图中的某一条路径。如果你把一个包DNAT回它原来的网络，这个包会继续走完相应路径上剩下的链，直到它被发送回原来的网络。



想要更多的信息，可以看看[rc.test-iptables.txt](#)，这个脚本包括了一些规则，它们会向你展示包是怎样通过各个表和链的。

3.2. mangle 表

这个表主要用来mangle包，你可以使用mangle匹配来改变包的TOS等特性。



强烈建议你不要在这个表里做任何过滤，不管是DANT，SNAT或者Masquerade。

以下是mangle表中仅有的几种操作：

- TOS
- TTL
- MARK

TOS操作用来设置或改变数据包的服务类型域。这常用来设置网络上的数据包如何被路由等策略。注意这个操作并不完善，有时得不所愿。它在Internet上还不能使用，而且很多路由器不会注意到这个域值。换句话说，不要设置发往Internet的包，除非你打算依靠TOS来路由，比如用iproute2。

TTL操作用来改变数据包的生存时间域，我们可以让所有数据包只有一个特殊的TTL。它的存在有一个很好的理由，那就是我们可以欺骗一些ISP。为什么要欺骗他们呢？因为他们不愿意让我们共享一个连接。那些ISP会查找一台单独的计算机是否使用不同的TTL，并且以此作为判断连接是否被共享的标志。

MARK用来给包设置特殊的标记。iproute2能识别这些标记，并根据不同的标记（或没有标记）决定不同的路由。用这些标记我们可以做带宽限制和基于请求的分类。

3.3. nat 表

此表仅用于NAT，也就是转换包的源或目标地址。注意，就象我们前面说过的，只有流的第一个包会被这个链匹配，其后的包会自动被做相同的处理。实际的操作分为以下几类：

- DNAT
- SNAT
- MASQUERADE

DNAT操作主要用在这样一种情况，你有一个合法的IP地址，要把对防火墙的访问重定向到其他的机子上（比如DMZ）。也就是说，我们改变的是目的地址，以使包能重路由到某台主机。

SNAT改变包的源地址，这在极大程度上可以隐藏你的本地网络或者DMZ等。一个很好的例子是我们知道防火墙的外部地址，但必须用这个地址替换本地网络地址。有了这个操作，防火墙就能自动地对包做SNAT和De-SNAT(就是反向的SNAT)，以使LAN能连接到Internet。如果使用类似 192.168.0.0/24这样的地址，是不会从Internet得到任何回应的。因为IANA定义这些网络（还有其他的）为私有的，只能用于LAN内部。

MASQUERADE的作用和MASQUERADE完全一样, 只是计算机的负荷稍微多一点。因为对每个匹配的包, MASQUERADE都要查找可用的IP地址, 而不象SNAT用的IP地址是配置好的。当然, 这也有好处, 就是我们可以使用通过PPP、PPPOE、SLIP等拨号得到的地址, 这些地址可是由ISP的DHCP随机分配的。

3.4. Filter 表

filter 表用来过滤数据包, 我们可以在任何时候匹配包并过滤它们。我们就是在这里根据包的内容对包做DROP或ACCEPT的。当然, 我们也可以预先在其他地方做些过滤, 但是这个表才是设计用来过滤的。几乎所有的target都可以在这儿使用。大量具体的介绍在后面, 现在你只要知道过滤工作主要是在这儿完成的就行了。

Chapter 4. 状态机制

本章将详细介绍状态机制。通读本章, 你会对状态机制是如何工作的有一个全面的了解。我们用一些例子来进行说明状态机制。实践出真知嘛。

4.1. 概述

状态机制是iptables中特殊的一部分, 其实它不应该叫状态机制, 因为它只是一种连接跟踪机制。但是, 很多人都认可状态机制这个名字。文中我也或多或少地用这个名字来表示和连接跟踪相同的意思。这不应该引起什么混乱的。连接跟踪可以让Netfilter知道某个特定连接的状态。运行连接跟踪的防火墙称作带有状态机制的防火墙, 以下简称为状态防火墙。状态防火墙比非状态防火墙要安全, 因为它允许我们编写更严密的规则。

在iptables里, 包是和被跟踪连接的四种不同状态有关的。它们是NEW, ESTABLISHED, RELATED和INVALID。后面我们会深入地讨论每一个状态。使用--state匹配操作, 我们能很容易地控制 “谁或什么能发起新的会话”。

所有在内核中由Netfilter的特定框架做的连接跟踪称作conntrack (译者注: 就是connection tracking 的首字母缩写)。conntrack可以作为模块安装, 也可以作为内核的一部分。大部分情况下, 我们想要, 也需要更详细的连接跟踪, 这是相比于缺省的conntrack而言。也因为此, conntrack中有许多用来处理TCP, UDP或ICMP协议的部件。这些模块从数据包中提取详细的、唯一的信息, 因此能保持对每一个数据流的跟踪。这些信息也告知conntrack流当前的状态。例如, UDP流一般由他们的目的地址、源地址、目的端口和源端口唯一确定。

在以前的内核里, 我们可以打开或关闭重组功能。然而, 自从iptables和Netfilter, 尤其是连接跟踪被引入内核, 这个选项就被取消了。因为没有包的重组, 连接跟踪就不能正常工作。现在重组已经整合入 conntrack, 并且在conntrack启动时自动启动。不要关闭重组功能, 除非你要关闭连接跟踪。

除了本地产生的包由OUTPUT链处理外, 所有连接跟踪都是在PREROUTING链里进行处理的, 意思就是, iptables会在PREROUTING链里重新计算所有的状态。如果我们发送一个流的初始化

包, 状态就会在OUTPUT链里被设置为**NEW**, 当我们收到回应的包时, 状态就会在PREROUTING链里被设置为**ESTABLISHED**。如果第一个包不是本地产生的, 那就会在PREROUTING链里被设置为NEW状态。综上, 所有状态的改变和计算都是在nat表中的PREROUTING链和OUTPUT链里完成的。

4.2. conntrack记录

我们先来看看怎样阅读/proc/net/ip_conntrack里的conntrack记录。这些记录表示的是当前被跟踪的连接。如果安装了ip_conntrack模块, `cat /proc/net/ip_conntrack` 的显示类似:

```
tcp      6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775 \
        dport=22 [UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22 \
        dport=32775 use=2
```

conntrack模块维护的所有信息都包含在这个例子中了, 通过它们就可以知道某个特定的连接处于什么状态。首先显示的是协议, 这里是tcp, 接着是十进制的6 (译者注: tcp的协议类型代码是6)。之后的117是这条conntrack记录的生存时间, 它会有规律地被消耗, 直到收到这个连接的更多的包。那时, 这个值就会被设为当时那个状态的缺省值。接下来的是这个连接在当前时间点的状态。上面的例子说明这个包处在状态 SYN_SENT, 这个值是iptables显示的, 以便我们好理解, 而内部用的值稍有不同。SYN_SENT说明我们正在观察的这个连接只在一个方向发送了一TCP SYN包。再下面是源地址、目的地址、源端口和目的端口。其中有个特殊的词UNREPLIED, 说明这个连接还没有收到任何回应。最后, 是希望接收的应答包的信息, 他们的地址和端口和前面是相反的。

连接跟踪记录的信息依据IP所包含的协议不同而不同, 所有相应的值都是在头文件 `linux/include/netfilter-ipv4/ip_conntrack*.h`中定义的。IP、TCP、UDP、ICMP协议的缺省值是在 `linux/include/netfilter-ipv4/ip_conntrack.h`里定义的。具体的值可以查看相应的协议, 但我们这里用不到它们, 因为它们大都只在conntrack内部使用。随着状态的改变, 生存时间也会改变。



最近patch-o-matic里有一个新的补丁, 可以把上面提到的超时时间也作为系统变量, 这样我们就能够在系统空闲时改变它们的值。以后, 我们就不必为了改变这些值而重编译内核了。

这些可通过/proc/sys/net/ipv4/netfilter下的一些特殊的系统调用来改变。仔细看看/proc/sys/net/ipv4/netfilter/ip_ct_*里的变量吧。

当一个连接在两个方向上都有传输时, conntrack记录就删除[UNREPLIED]标志, 然后重置。在末尾有 [ASSURED]的记录说明两个方向已没有流量。这样的记录是确定的, 在连接跟踪表满时, 是不会被删除的, 没有[ASSURED]的记录就要被删除。连接跟踪表能容纳多少记录是被一个变量控制的, 它可由内核中的ip_sysctl函数设置。默认值取决于你的内存大小, 128MB可以包含8192条目录, 256MB是16376条。你也可以在 /proc/sys/net/ipv4/ip_conntrack_max里查看、设置。

4.3. 数据包在用户空间的状态

就象前面说的, 包的状态依据IP所包含的协议不同而不同, 但在内核外部, 也就是用户空间里, 只有4种状态: **NEW**, **ESTABLISHED**, **RELATED** 和 **INVALID**。它们主要是和状态匹配一起使用。下面就简要地介绍以下这几种状态:

Table 4-1. 数据包在用户空间的状态

State (状态)	Explanation (注释)
NEW	NEW 说明这个包是我们看到的第一个包。意思就是, 这是conntrack模块看到的某个连接第一个包, 它即将被匹配了。比如, 我们看到一个SYN 包, 是我们所留意的连接的第一个包, 就要匹配它。第一个包也可能不是SYN包, 但它仍会被认为是 NEW 状态。这样做有时会导致一些问题, 但对某些情况是有非常大的帮助的。例如, 在我们想恢复某条从其他的防火墙丢失的连接时, 或者某个连接已经超时, 但实际上并未关闭时。
ESTABLISHED	ESTABLISHED 已经注意到两个方向上的数据传输, 而且会继续匹配这个连接的包。处于 ESTABLISHED 状态的连接是非常容易理解的。只要发送并接到应答, 连接就是 ESTABLISHED 的了。一个连接要从 NEW 变为 ESTABLISHED , 只需要接到应答包即可, 不管这个包是发往防火墙的, 还是要由防火墙转发的。ICMP的错误和重定向等信息包也被看作是 ESTABLISHED , 只要它们是我们所发出的信息的应答。
RELATED	RELATED 是个比较麻烦的状态。当一个连接和某个已处于 ESTABLISHED 状态的连接有关系时, 就被认为是 RELATED 的了。换句话说, 一个连接要想是 RELATED 的, 首先要有一个 ESTABLISHED 的连接。这个 ESTABLISHED 连接再产生一个主连接之外的连接, 这个新的连接就是 RELATED 的了, 当然前提是conntrack模块要能理解 RELATED 。ftp是个很好的例子, FTP-data 连接就是和FTP-control有 RELATED 的。还有其他的例子, 比如, 通过IRC的DCC连接。有了这个状态, ICMP应答、FTP传输、DCC等才能穿过防火墙正常工作。注意, 大部分还有一些UDP协议都依赖这个机制。这些协议是很复杂的, 它们把连接信息放在数据包里, 并且要求这些信息能被正确理解。
INVALID	INVALID 说明数据包不能被识别属于哪个连接或没有任何状态。有几个原因可以产生这种情况, 比如, 内存溢出, 收到不知属于哪个连接的ICMP 错误信息。一般地, 我们DROP这个状态的任何东西。

这些状态可以一起使用, 以便匹配数据包。这可以使我们的防火墙非常强壮和有效。以前, 我们经常打开1024以上的所有端口来放行应答的数据。现在, 有了状态机制, 就不需再这样了。因为我们可以只开放那些有应答数据的端口, 其他的都可以关闭。这样就安全多了。

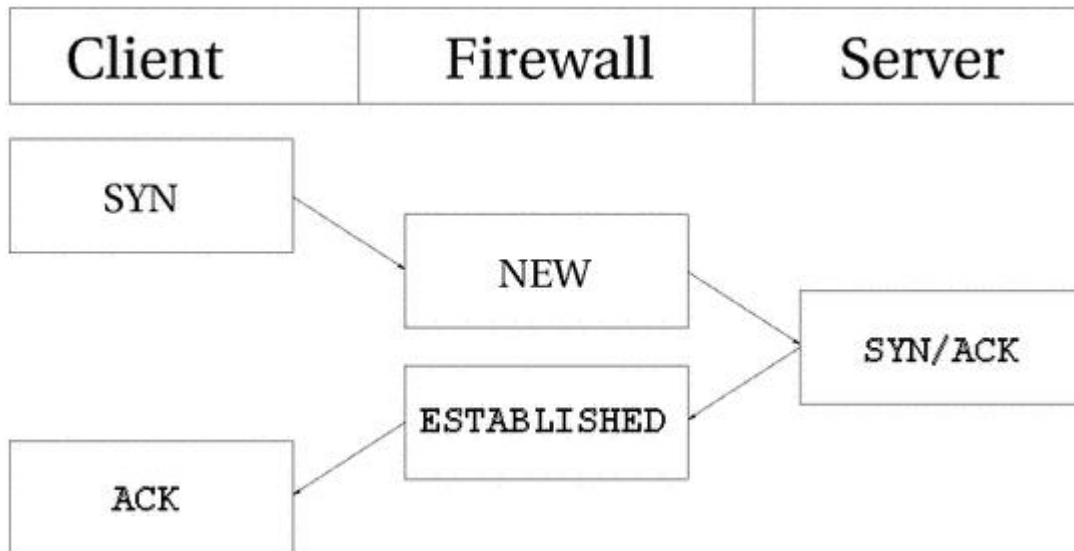
4.4. TCP 连接

本节和下面的几节, 我们来详细讨论这些状态, 以及在TCP、UDP和ICMP这三种基本的协议里怎样操作它们。当然, 也会讨论其他协议的情况。我们还是从TCP入手, 因为它本身就是一个带状态的协议, 并且具有很多关于iptables状态机制的详细信息。

一个TCP连接是经过三次握手协商连接信息才建立起来的。整个会话由一个SYN包开始, 然后是一个 SYN/ACK包, 最后是一个ACK包, 此时, 会话才建立成功, 能够发送数据。最大的问题在于连接跟踪怎样控制这个过程。其实非常简单。

默认情况下, 连接跟踪基本上对所有的连接类型做同样的操作。看看下面的图片, 我们就能明白在连接的不同阶段, 流是处于什么状态的。就如你看到的, 连接跟踪的代码不是从用户

的观点来看待TCP连接建立的流程的。连接跟踪一看到SYN包，就认为这个连接是NEW状态，一看到返回的SYN/ACK包，就认为连接是 ESTABLISHED状态。如果你仔细想想第二步，应该能理解为什么。有了这个特殊处理，NEW和ESTABLISHED包就可以发送出本地网络，且只有ESTABLISHED的连接才能有回应信息。如果把整个建立连接的过程中传输的数据包都看作NEW，那么三次握手所用的包都是NEW状态的，这样我们就不能阻塞从外部到本地网络的连接了。因为即使连接是从外向内的，但它使用的包也是NEW状态的，而且为了其他连接能正常传输，我们不得不允许NEW状态的包返回并进入防火墙。更复杂的是，针对TCP连接内核使用了很多内部状态，它们的定义在 [RFC 793 - Transmission Control Protocol](#)的21-23页。但好在我们在用户空间用不到。后面我们会详细地介绍这些内容。



正如你看到的，以用户的观点来看，这是很简单的。但是，从内核的角度看这一块还有点困难的。我们来看一个例子。认真考虑一下在`/proc/net/ip_conntrack`里，连接的状态是如何改变的。

```

tcp      6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 \
        dport=23 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 \
        dport=1031 use=1
  
```

从上面的记录可以看出，SYN_SENT状态被设置了，这说明连接已经发出一个SYN包，但应答还没发送过来，这可从[UNREPLIED]标志看出。

```

tcp      6 57 SYN_RECV src=192.168.1.5 dst=192.168.1.35 sport=1031 \
        dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 \
        use=1
  
```

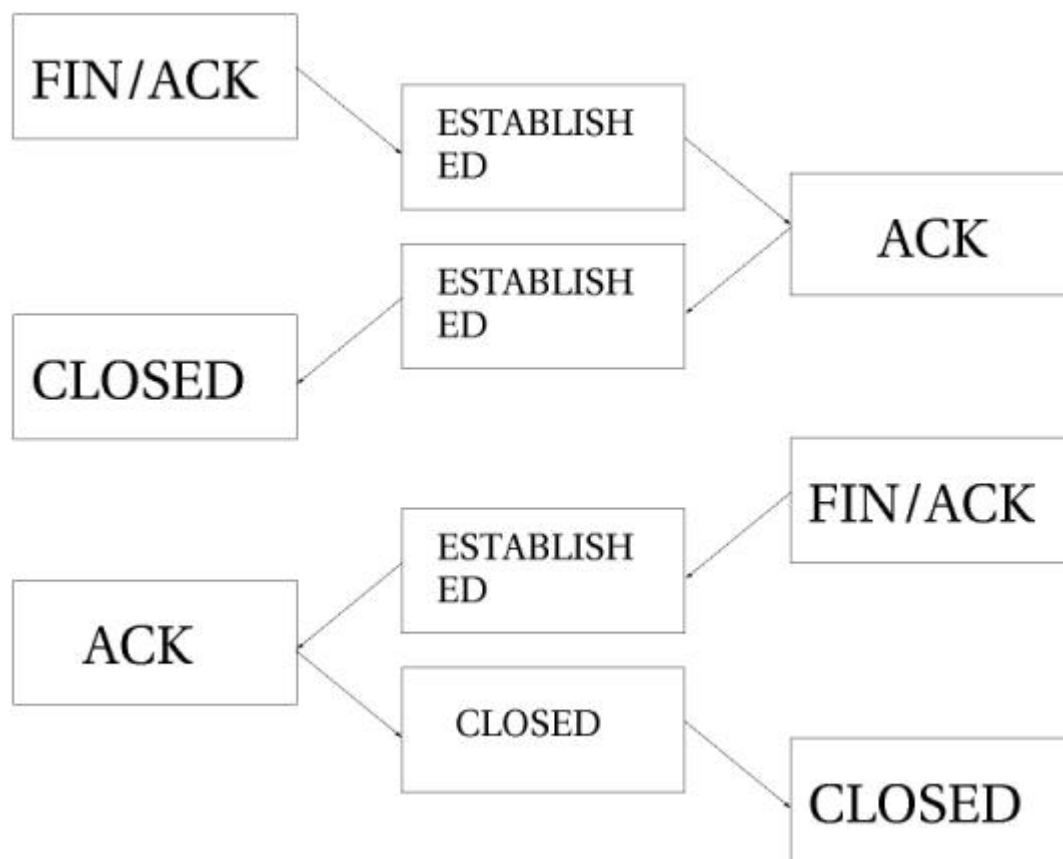
现在我们已经收到了相应的SYN/ACK包，状态也变为SYN_RECV，这说明最初发出的SYN包已正确传输，并且SYN/ACK包也到达了防火墙。这就意味着在连接的两方都有数据传输，因此可以认为两个方向都有相应的回应。当然，这是假设的。

```

tcp      6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 \
        sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 \
        sport=23 dport=1031 use=1
  
```

现在我们发出了三步握手的最后一个包，即ACK包，连接也就进入ESTABLISHED状态了。再传输几个数据包，连接就是[ASSURED]的了。

下面介绍TCP连接在关闭过程中的状态。



如上图，在发出最后一个ACK包之前，连接（指两个方向）是不会关闭的。注意，这只是针对一般的情况。连接也可以通过发送关闭，这用在拒绝一个连接的时候。在RST包发送之后，要经过预先设定的一段时间，连接才能断掉。

连接关闭后，进入TIME_WAIT状态，缺省时间是2分钟。之所以留这个时间，是为了让数据包能完全通过各种规则的检查，也是为了数据包能通过拥挤的路由器，从而到达目的地。

如果连接是被RST包重置的，就直接变为CLOSE了。这意味着在关闭之前只有10秒的默认时间。RST包是不需要确认的，它会直接关闭连接。针对TCP连接，还有其他一些状态我们没有谈到。下面给出一个完整的状态列表和超时值。

Table 4-2. 内部状态

State	Timeout value
NONE	30 minutes
ESTABLISHED	5 days
SYN_SENT	2 minutes
SYN_RECV	60 seconds
FIN_WAIT	2 minutes
TIME_WAIT	2 minutes
CLOSE	10 seconds
CLOSE_WAIT	12 hours

LAST_ACK	30 seconds
LISTEN>	2 minutes

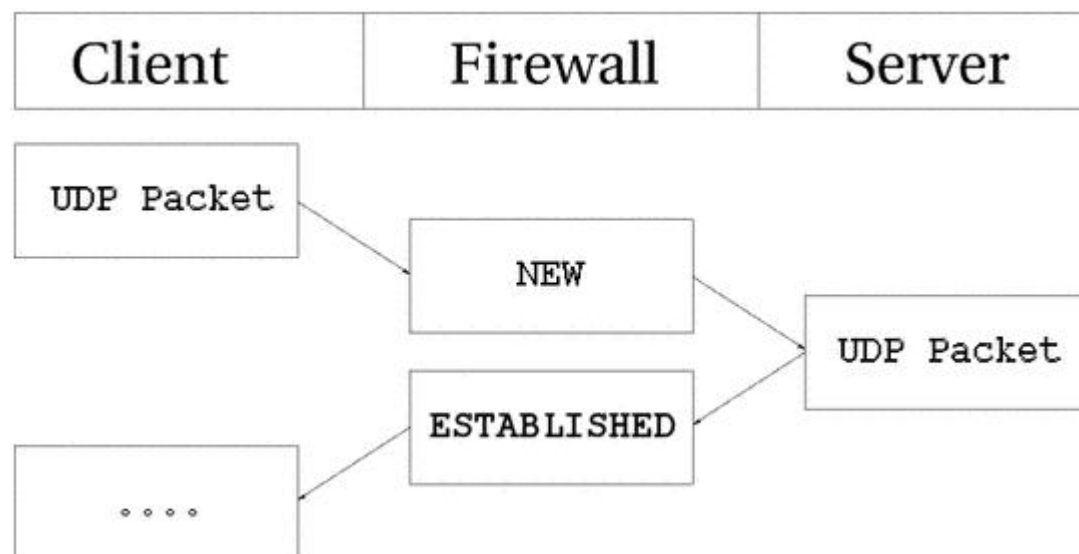
这些值不是绝对的，可以随着内核的修订而变化，也可以通过`/proc/sys/net/ipv4/netfilter/ip_ct_tcp_*`的变量更改。这些默认值都是经过实践检验的。它们的单位是jiffies（百分之一秒），所以3000就代表30秒。



注意状态机制在用户空间里的部分不会查看TCP包的标志位（也就是说TCP标志对它而言是透明的）。如果我们想让NEW状态的包通过防火墙，就要指定NEW状态，我们理解的NEW状态的意思就是指SYN包，可是iptables又不查看这些标志位。这就是问题所在。有些没有设置SYN或ACK的包，也会被看作NEW状态的。这样的包可能会被冗余防火墙用到，但对只有一个防火墙的网络是很不利的（可能会被攻击哦）。那我们怎样才能不受这样的包的影响呢？你可以使用[未设置SYN的NEW状态包](#)里的命令。还有一个办法，就是安装patch-o-matic里的tcp-window-tracking扩展功能，它可以使防火墙能根据TCP的一些标志位来进行状态跟踪。

4.5. UDP连接

UDP连接是无状态的，因为它没有任何的连接建立和关闭过程，而且大部分是无序列号的。以某个顺序收到的两个数据包是无法确定它们的发出顺序的。但内核仍然可以对UDP连接设置状态。我们来看看是如何跟踪UDP连接的，以及conntrack的相关记录。



从上图可以看出，以用户的角度考虑，UDP连接的建立几乎与TCP的一样。虽然conntrack信息看起来有点儿不同，但本质上是一样的。下面我们先来看看第一个UDP包发出后的conntrack记录。

```
udp      17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 \
[UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025 \
dport=137 use=1
```

从前两个值可知，这是一个UDP包。第一个是协议名称，第二个是协议号，第三个是此状态的生存时间，默认是30秒。接下来是包的源、目地址和端口，还有期待之中回应包的源、目地址和端口。[UNREPLIED]标记说明还未收到回应。

```
udp      17 170 src=192.168.1.2 dst=192.168.1.5 sport=137 \
dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 \
dport=137 use=1
```

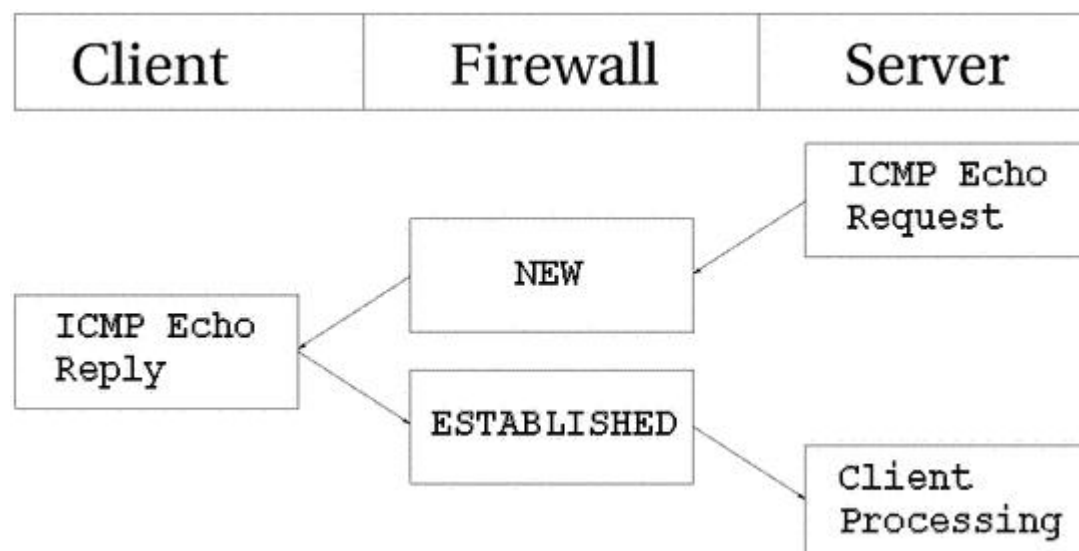
一旦收到第一个包的回应, [UNREPLIED] 标记就会被删除, 连接就被认为是 ESTABLISHED 的, 但在记录里并不显示 ESTABLISHED 标记。相应地, 状态的超时时间也变为 180 秒了。在本例中, 只剩 170 秒了, 10 秒后, 就会减少为 160 秒。有个东西是不可少的, 虽然它可能会有些变化, 就是前面提过的 [ASSURED]。要想变为 [ASSURED] 状态, 连接上必须要再有些流量。

```
udp      17 175 src=192.168.1.5 dst=195.22.79.2 sport=1025 \
dport=53 src=195.22.79.2 dst=192.168.1.5 sport=53 \
dport=1025 [ASSURED] use=1
```

可以看出来, [ASSURED] 状态的记录和前面的没有多大差别, 除了标记由 [UNREPLIED] 变成 [ASSURED]。如果这个连接持续不了 180 秒, 那就要被中断。180 秒是短了点儿, 但对大部分应用足够了。只要遇到这个连接的包穿过防火墙, 超时值就会被重置为默认值, 所有的状态都是这样的。

4.6. ICMP 连接

ICMP 也是一种无状态协议, 它只是用来控制而不是建立连接。ICMP 包有很多类型, 但只有四种类型有应答包, 它们是回显请求和应答 (Echo request and reply), 时间戳请求和应答 (Timestamp request and reply), 信息请求和应答 (Information request and reply), 还有地址掩码请求和应答 (Address mask request and reply), 这些包有两种状态, **NEW** 和 **ESTABLISHED**。时间戳请求和信息请求已经废除不用了, 回显请求还是常用的, 比如 ping 命令就用的到, 地址掩码请求不太常用, 但是可能有时很有用并且值得使用。看看下面的图, 就可以大致了解 ICMP 连接的 **NEW** 和 **ESTABLISHED** 状态了。



如图所示, 主机向目标发送一个回显请求, 防火墙就认为这个包处于 **NEW** 状态。目标回应一个回显应答, 防火墙就认为包处于 **ESTABLISHED** 了。当回显请求被发送时, ip_conntrack 里就有这样的记录了:

```
icmp     1 25 src=192.168.1.6 dst=192.168.1.10 type=8 code=0 \
```

```
id=33029 [UNREPLIED] src=192.168.1.10 dst=192.168.1.6 \
type=0 code=0 id=33029 use=1
```

可以看到, ICMP的记录和TCP、UDP的有点区别, 协议名称、超时时间和源、目地址都一样, 不同之处在于没有了端口, 而新增了三个新的字段: type, code和id。字段type说明ICMP的类型。code说明ICMP的代码, 这些代码在附录 [ICMP类型](#) 里有说明。id是ICMP包的ID。每个ICMP包被发送时都被分配一个ID, 接受方把同样的ID 分配给应答包, 这样发送方能认出是哪个请求的应答。

[UNREPLIED]的含义和前面一样, 说明数的传输只发生在一个方向上, 也就是说未收到应答。再往后, 是应答包的源、目地址, 还有相应的三个新字段, 要注意的是type和code是随着应答包的不同而变化的, id和请求包的一样。

和前面一样, 应答包被认为是**ESTABLISHED**的。然而, 在应答包之后, 这个ICMP 连接就不再有数据传输了。所以, 一旦应答包穿过防火墙, ICMP的连接跟踪记录就被销毁了。

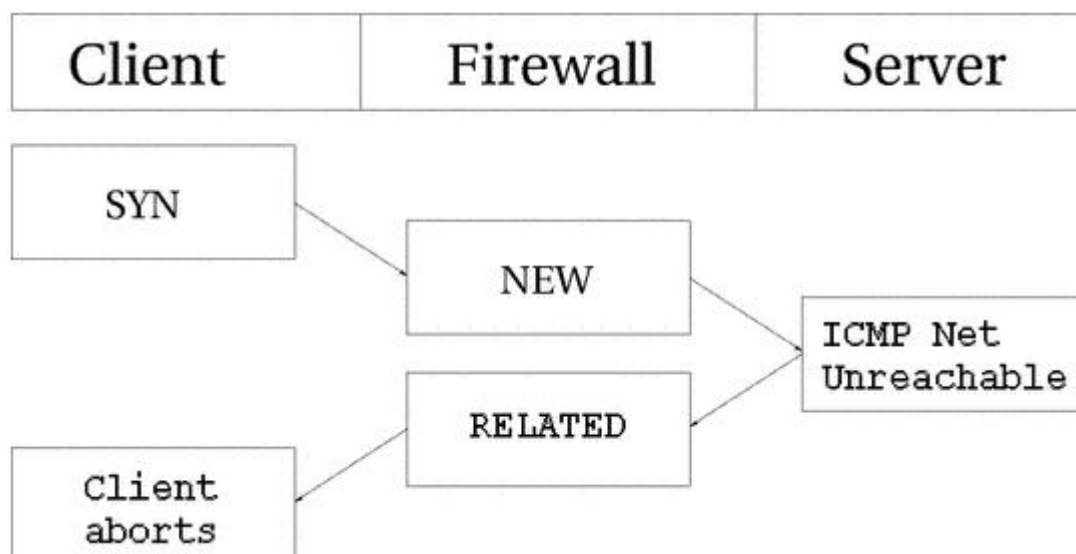
以上各种情况, 请求被认为**NEW**, 应答是**ESTABLISHED**。换句话说, 就是当防火墙看到一个请求包时, 就认为连接处于**NEW**状态, 当有应答时, 就是**ESTABLISHED**状态。



注意, 应答包必须符合一定的标准, 连接才能被认作established的, 每个传输类型都是这样。

ICMP的缺省超时是30秒, 可以在`/proc/sys/net/ipv4/netfilter/ip_ct_icmp_timeout`中修改。这个值是比较合适的, 适合于大多数情况。

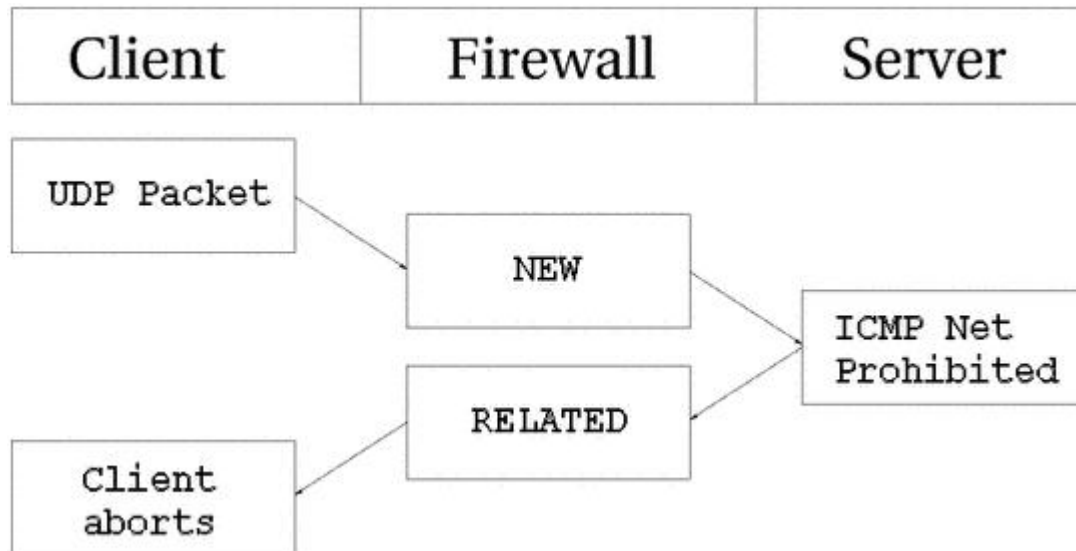
ICMP的另一个非常重要的作用是, 告诉UDP、TCP连接或正在努力建立的连接发生了什么, 这时ICMP应答被认为是**RELATED**的。主机不可达和网络不可达就是这样的例子。当试图连接某台机器不成功时(可能那台机器被关上了), 数据包所到达的最后一台路由器就会返回以上的ICMP信息, 它们就是**RELATED**的, 如下图:



我们发送了一个SYN包到某一地址, 防火墙认为它的状态是**NEW**。但是, 目标网络有问题不可达, 路由器就会返回网络不可达的信息, 这是**RELATED**的。连接跟踪会认出这个错误信息是哪个连接的, 连接会中断, 同时相应的记录删除会被删除。

当UDP连接遇到问题时, 同样会有相应的ICMP信息返回, 当然它们的状态也是**RELATED** , 如下

图:



我们发送一个UDP包，当然它是**NEW**的。但是，目标网络被一些防火墙或路由器所禁止。我们的防火墙就会收到网络被禁止的信息。防火墙知道它是和哪个已打开的UDP连接相关的，并且把这个信息（状态是**RELATED**）发给它，同时，把相应的记录删除。客户机收到网络被禁止的信息，连接将被中断。

4.7. 缺省的连接操作

有时，conntrack机制并不知道如何处理某个特殊的协议，尤其是在它不了解这个协议或不知道协议如何工作时，比如，NETBLT，MUX还有EGP。这种情况下，conntrack使用缺省的操作。这种操作很象对UDP连接的操作，就是第一个包被认作**NEW**，其后的应答包等等数据都是**ESTABLISHED**。

使用缺省操作的包的超时值都是一样的，600秒，也就是10分钟。当然，这个值可以通过`/proc/sys/net/ipv4/netfilter/ip_ct_generic_timeout`更改，以便适应你的通信量，尤其是在耗时较多、流量巨大的情况下，比如使用卫星等。

4.8. 复杂协议和连接跟踪

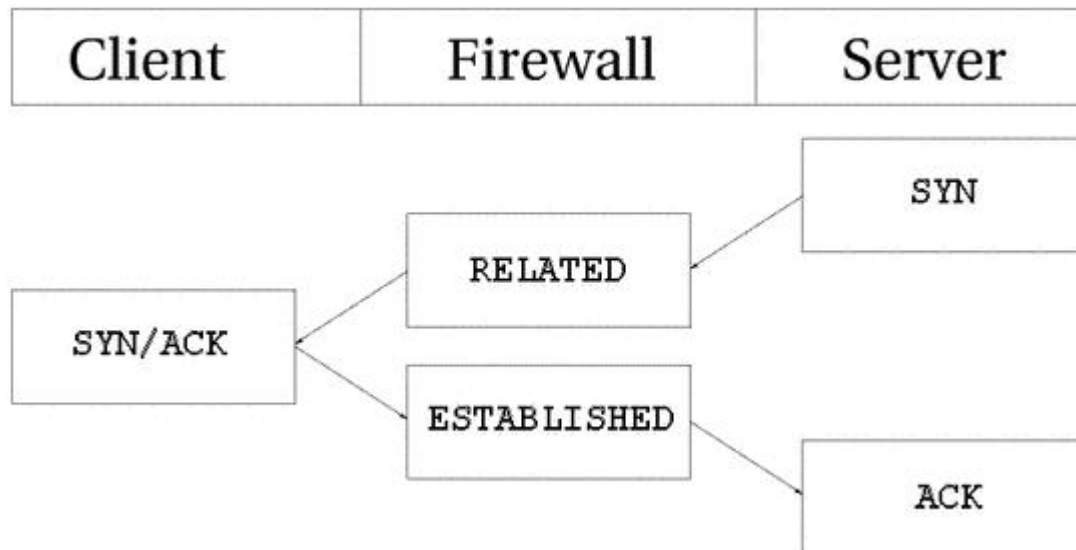
有些协议比其他协议更复杂，这里复杂的意思是指连接跟踪机制很难正确地跟踪它们，比如，ICQ、IRC 和FTP，它们都在数据包的数据域里携带某些信息，这些信息用于建立其他的连接。因此，需要一些特殊的 helper来完成工作。

下面以FTP作为例子。FTP协议先建立一个单独的连接——FTP控制会话。我们通过这个连接发布命令，其他的端口就会打开以便传输和这个命令相关的数据。这些连接的建立方法有两种：主动模式和被动模式。先看看主动模式，FTP客户端发送端口和IP地址信息给服务器端，然后，客户端打开这个端口，服务器端从它自己的20端口（FTP-Data端口号）建立与这个端口的连接，接着就可以使用这个连接发送数据了。

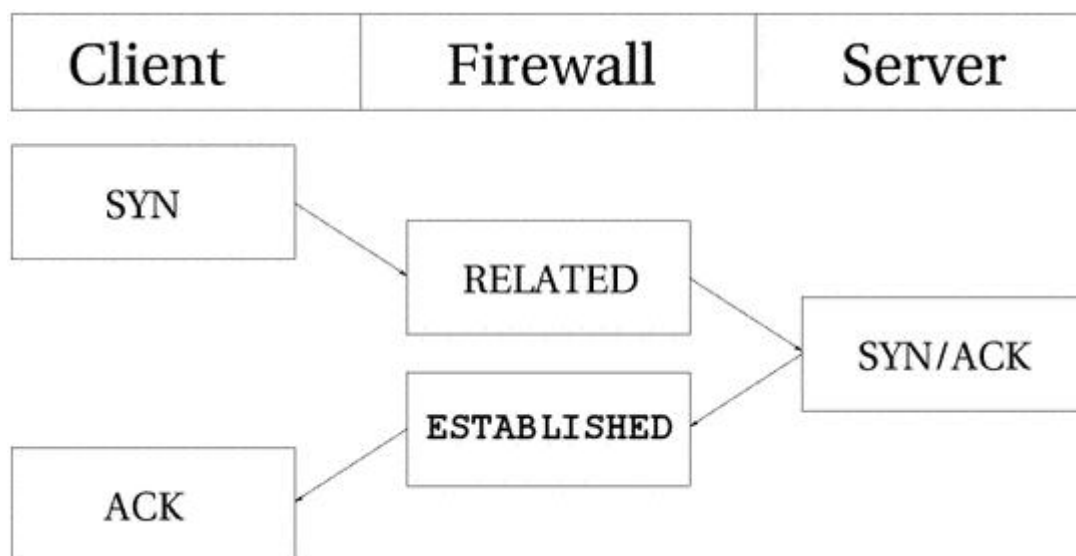
问题在于防火墙不知道这些额外的连接（相对于控制会话而言），因为这些连接在建立时的

磋商信息都在协议数据包的数据域内，而不是在可分析的协议头里。因此，防火墙就不知道是不是该放这些从服务器到客户机的连接过关。

解决的办法是为连接跟踪模块增加一个特殊的helper，以便能检测到那些信息。这样，那些从FTP服务器到客户机的连接就可以被跟踪了，状态是**RELATED**，过程如下图所示：



被动FTP工作方式下，data连接的建立过程和主动FTP的相反。客户机告诉服务器需要某些数据，服务器就把地址和端口发回给客户机，客户机据此建立连接接受数据。如果FTP服务器在防火墙后面，或你对用户限制的比较严格，只允许他们访问HTTP和FTP，而封闭了其他所有端口，为了让在Internet是的客户机能访问到FTP，也需要增加上面提到的helper。下面是被动模式下data连接的建立过程：



有些conntrack helper已经包含在内核中，在写这篇文章时，FTP和IRC已有了相应的conntrack helper。如果在内核里没有你想要的helper，可以到iptables用户空间的patch-o-matic目录中看看，那里有很多的helper，比如针对ntalk或H.323协议的等等。如果没找到，还有几个选择：可以查查iptables的 CVS，或者联系[Netfilter-devel](#)问问有没有你要的。还不行的话，只有你自己写了，我可以给你介绍一篇好文章，[Rusty Russell's Unreliable Netfilter Hacking HOW-TO](#)，连接放在附录里[其他资源和链接](#)。

Conntrack helper即可以被静态地编译进内核，也可以作为模块，但要用下面的命令装载：

```
modprobe ip_conntrack_*
```

注意连接跟踪并不处理NAT，因此要对连接做NAT就需要增加相应的模块。比如，你想NAT并跟踪FTP连接，除了FTP的相应模块，还要有NAT的模块。所有的NAT helper名字都是以ip_nat_开头的，这是一个命名习惯：FTP NAT helper叫做ip_nat_ftp，IRC的相应模块就是ip_nat_irc。conntrack helper 的命名也遵循一样的习惯：针对IRC的conntrack helper叫ip_conntrack_irc，FTP的叫作ip_conntrack_ftp。

Chapter 5. 规则的保存与恢复

iptables提供了两个很有用的工具用来处理大规则集：**iptables-save**和**iptables-restore**，它们把规则存入一个与标准脚本代码只有细微查别的特殊格式的文件中，或从中恢复规则。

5.1. 速度

使用**iptables-save**和**iptables-restore**的一个最重要的原因是，它们能在相当程度上提高装载、保存规则的速度。使用脚本更改规则的问题是，改动每个规则都要调运命令iptables，而每一次调用iptables，它首先要将Netfilter内核空间中的整个规则集都提取出来，然后再插入或附加，或做其他的改动，最后，再把新的规则集从它的内存空间插入到内核空间中。这会花费很多时间。

为了解决这个问题，可以使用命令**iptables-save**和**restore**。**iptables-save**用来把规则集保存到一个特殊格式的文本文件里，而**iptables-restore**是用来把这个文件重新装入内核空间的。这两个命令最好的地方在于一次调用就可以装载和保存规则集，而不象脚本中每个规则都要调用一次iptables。**iptables-save**运行一次就可以把整个规则集从内核里提取出来，并保存到文件里，而**iptables-restore**每次装入一个规则表。换句话说，对于一个很大的规则集，如果用脚本来设置，那这些规则就会反反复复地被卸载、安装很多次，而我们现在可以把整个规则集一次就保存下来，安装时则是一次一个表，这可是节省了大量的时间。

如果你的工作对象是一组巨大的规则，这两个工具是明显的选择。当然，它们也有不足之处，下面的章节会详细说明。

5.2. restore的不足之处

iptables-restore能替代所有的脚本来设置规则吗？不，到现在为止不行，很可能永远都不行。iptables-restore的主要不足是不能用来做复杂的规则集。例如，我们想在计算机启动时获取连接的动态分配的IP地址，然后用在脚本里。这一点，用iptables-restore来实现，或多或少是不可能的。

一个可能的解决办法是写一个小脚本来获取那个IP地址，并在iptables-restore调用的配置文件中设置相应的关键字，然后用获取的IP值替换关键字。你可以把更改后的配置文件存到一个临时文件中，再由 iptables-restore使用它。然而这会带来很多问题，并且你不能

iptables-save来保存带关键字的配置文件。此法较笨。

另一个办法是先装入iptables-restore文件，再运行一个特定的脚本把动态的规则装入。其实，这也是较笨的方法。iptables-restore并不适合于使用动态IP的场合，如果你想在配置文件里使用选项来实现不同的要求，iptables-restore也不适用。

iptables-restore和iptables-save还有一个不足，就是功能不够齐全。因为使用的人不是太多，所以发现这个问题的人也不多，还有就是一些match和target被引用时考虑不细致，这可能会出现我们预期之外的行为。尽管存在这些问题，我还是强烈建议你使用它们，因为它们对于大部分规则集工作的还是很好的，只要在规则中别包含那些新的都不知如何使用的match和target。

5.3. iptables-save

iptables-save用来把当前的规则存入一个文件里以备iptables-restore使用。它的使用很简单，只有两个参数：

```
iptables-save [-c] [-t table]
```

参数-c的作用是保存包和字节计数器的值。这可以使我们在重启防火墙后不丢失对包和字节的统计。带-c参数的iptables-save命令使重启防火墙而不中断统计记数程序成为可能。这个参数默认是不使用的。

参数-t指定要保存的表，默认是保存所有的表。下面给出未装载任何规则的情况下iptables-save的输出。

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*filter
:INPUT ACCEPT [404:19766]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [530:43376]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*mangle
:PREROUTING ACCEPT [451:22060]
:INPUT ACCEPT [451:22060]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [594:47151]
:POSTROUTING ACCEPT [594:47151]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
```

我们来解释一下这个输出格式。#后面的是注释。表都以*<table-name>开始，例如*mangle。每个表都包含链和规则，链的详细说明是:<chain-name> <chain-policy> [<packet-counter>:<byte-counter>]。例如，链的名字是 PREROUTING，策略是ACCEPT，然后是包计数器和字节计数器，这两个计数器和iptables -L -v输出中用到的计数器一样。每个表的描述

都以关键字COMMIT结束，它说明在这一点，就要把规则装入内核了。

上面的例子是最基本的，我想用一个简短的例子说明会更好，其中包含一个非常小的规则集 [*iptables-save ruleset*](#)。iptables-save的输出如下：

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*filter
:INPUT DROP [1:229]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth1 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*mangle
:PREROUTING ACCEPT [658:32445]
:INPUT ACCEPT [658:32445]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [891:68234]
:POSTROUTING ACCEPT [891:68234]
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*nat
:PREROUTING ACCEPT [1:229]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
-A POSTROUTING -o eth0 -j SNAT --to-source 195.233.192.1
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
```

每个命令前都有包和字节计数器，这说明使用了-c参数。除了有计数器，其他的都和普通的脚本一样。现在的问题是怎么把输出保存到文件中。非常简单，既然使用linux，你应该早就知道了，用重定向啊：

```
iptables-save -c > /etc/iptables-save
```

这就会把规则集保存到/etc/iptables-save中，而且还有计数器。

5.4. iptables-restore

iptables-restore用来装载由iptables-save保存的规则集。不幸的是，它只能从标准输入接受输入，而不能从文件接受。下面是它的事方法：

```
iptables-restore [-c] [-n]
```

参数-c要求装入包和字节计数器。如果你用iptables-save保存了计数器，现在想重新装入，就必须用这个参数。它的另一种较长的形式是--counters。

参数-n告诉iptables-restore不要覆盖已有的表或表内的规则。默认情况是清除所有已存的规则。这个参数的长形式是--noflush。

用**iptables-restore**装载规则有好几种方法，我们来看看最简单、最一般的：

这样规则集应该正确地装入内核并正常工作了。如果有问题，你就要除错了。

Chapter 6. 规则是如何练成的

本章将详细地讨论如何构件你自己的规则。规则就是指向标，在一条链上，对不同的连接和数据包阻塞或允许它们去向何处。插入链的每一行都是一条规则。我们也会讨论基本的 **match** 及其用法，还有各种各样的 **target**，以及如何建立我们自己的 **target**（比如，一个新的子链）。

6.1. 基础

我们已经解释了什么是规则，在内核看来，规则就是决定如何处理一个包的语句。如果一个包符合所有的条件（就是符合 **match** 语句），我们就运行 **target** 或 **jump** 指令。书写规则的语法格式是：

```
iptables [-t table] command [match] [target/jump]
```

对于这个句法没什么可说的，但注意 **target** 指令必须在最后。为了易读，我们一般用这种语法。总之，你将见到的大部分规则都是按这种语法写的。因此，如果你看到别人写的规则，你很可能会发现用的也是这种语法，当然就很容易理解那些规则了。

如果你不想用标准的表，就要在 [*table*] 处指定表名。一般情况下没有必要指定使用的表，因为 **iptables** 默认使用 **filter** 表来执行所有的命令。也没有必要非得在这里指定表名，实际上几乎可在规则的任何地方。当然，把表名在开始处已经是约定俗成的标准。

尽管命令总是放在开头，或者是直接放在表名后面，我们也要考虑考虑到底放在哪儿易读。**command** 告诉程序该做什么，比如：插入一个规则，还是在链的末尾增加一个规则，还是删除一个规则，下面会仔细地介绍。

match 细致地描述了包的某个特点，以使这个包区别于其它所有的包。在这里，我们可以指定包的来源 IP 地址，网络接口，端口，协议类型，或者其他什么。下面我们将会看到许多不同的 **match**。

最后是数据包的目标所在。若数据包符合所有的 **match**，内核就用 **target** 来处理它，或者说把包发往 **target**。比如，我们可以让内核把包发送到当前表中的其他链（可能是我们自己建立的），或者只是丢弃这个包而没有什么处理，或者向发送者返回某个特殊的应答。下面有详细的讨论。

6.2. Tables

选项 **-t** 用来指定使用哪个表，它可以是下面介绍的表中的任何一个，默认的是 **filter** 表。注意，下面的介绍只是章节 [表和链](#) 的摘要。

Table 6-1. Tables

Table (表名)	Explanation (注释)
nat	nat表的主要用处是网络地址转换, 即Network Address Translation, 缩写为NAT。做过NAT操作的数据包的地址就被改变了, 当然这种改变是根据我们的规则进行的。属于一个流的包只会经过这个表一次。如果第一个包被允许做NAT或Masqueraded, 那么余下的包都会自动地被做相同的操作。也就是说, 余下的包不会再通过这个表, 一个一个的被NAT, 而是自动地完成。这就是我们为什么不应该在这个表中做任何过滤的主要原因, 对这一点, 后面会有更加详细的讨论。PREROUTING 链的作用是在包刚刚到达防火墙时改变它的目的地址, 如果需要的话。OUTPUT链改变本地产生的包的目的地址。POSTROUTING链在包就要离开防火墙之前改变其源地址。
mangle	这个表主要用来mangle数据包。我们可以改变不同的包及包头的内容, 比如 TTL , TOS 或 MARK 。注意 MARK 并没有真正地改动数据包, 它只是在内核空间为包设了一个标记。防火墙内的其他的规则或程序(如tc)可以使用这种标记对包进行过滤或高级路由。这个表有五个内建的链: <i>PREROUTING</i> , <i>POSTROUTING</i> , <i>OUTPUT</i> , <i>INPUT</i> 和 <i>FORWARD</i> 。 <i>PREROUTING</i> 在包进入防火墙之后、路由判断之前改变包, <i>POSTROUTING</i> 是在所有路由判断之后。 <i>OUTPUT</i> 在确定包的目的地之前更改数据包。 <i>INPUT</i> 在包被路由到本地之后, 但在用户空间的程序看到它之前改变包。 <i>FORWARD</i> 在最初的路由判断之后、最后一次更改包的目的地之前mangle包。注意, mangle表不能做任何NAT, 它只是改变数据包的 TTL , TOS 或 MARK , 而不是其源目地址。NAT是在nat表中操作的。
filter	<i>filter</i> 表是专门过滤包的, 内建三个链, 可以毫无问题地对包进行 DROP 、 LOG 、 ACCEPT 和 REJECT 等操作。 <i>FORWARD</i> 链过滤所有不是本地产生的并且目的地不是本地(所谓本地就是防火墙了)的包, 而 <i>INPUT</i> 恰恰针对那些目的地是本地的包。 <i>OUTPUT</i> 是用来过滤所有本地生成的包的。

上面介绍了三个不同的表的最基本的内容。你应该知道它们的使用目的完全不同, 还要清楚每一条链的使用。如果你不了解, 就可能会在防火墙上留下漏洞, 给人以可乘之机。在章节[表和链](#)中, 我们已详细地讨论了这些必备的表和链。如果你没有完全理解包是怎样通过这些表、链的话, 我建议你回过头去再仔细看看。

6.3. Commands

在这一节里, 我们将要介绍所有的command以及它们的用途。command指定**iptables** 对我们提交的规则要做什么样的操作。这些操作可能是在某个表里增加或删除一些东西, 或做点儿其他什么。以下是iptables可用的command(要注意, 如不做说明, 默认表的是 *filter* 表。):

Table 6-2. Commands

Command	-A , --append
Example	iptables -A INPUT ...
Explanation	在所选择的链末添加规则。当源地址或目的地址是以名字而不是ip地址的形式出现时, 若这些名字可以被解析为多个地址, 则这条规则会和所有可用的地址结合。

Command	-D, --delete
Example	iptables -D INPUT --dport 80 -j DROP 或 iptables -D INPUT 1
Explanation	从所选链中删除规则。有两种方法指定要删除的规则：一是把规则完完整整地写出来，再就是指定规则在所选链中的序号（每条链的规则都各自从1被编号）。
Command	-R, --replace
Example	iptables -R INPUT 1 -s 192.168.0.1 -j DROP
Explanation	在所选中的链里指定的行上（每条链的规则都各自从1被编号）替换规则。它主要的用处是试验不同的规则。当源地址或目的地址是以名字而不是ip地址的形式出现时，若这些名字可以被解析为多个地址，则这条command会失败。
Command	-I, --insert
Example	iptables -I INPUT 1 --dport 80 -j ACCEPT
Explanation	根据给出的规则序号向所选链中插入规则。如果序号为1，规则会被插入链的头部，其实默认序号就是1。
Command	-L, --list
Example	iptables -L INPUT
Explanation	显示所选链的所有规则。如果没有指定链，则显示指定表中的所有链。如果什么都没有指定，就显示默认表所有的链。精确输出受其它参数影响，如 -n 和 -v 等参数，下面会介绍。
Command	-F, --flush
Example	iptables -F INPUT
Explanation	清空所选的链。如果没有指定链，则清空指定表中的所有链。如果什么都没有指定，就清空默认表所有的链。当然，也可以一条一条地删，但用这个command会快些。
Command	-Z, --zero
Example	iptables -Z INPUT
Explanation	把指定链（如未指定，则认为是所有链）的所有计数器归零。
Command	-N, --new-chain
Example	iptables -N allowed
Explanation	根据用户指定的名字建立新的链。上面的例子建立了一个名为 allowed 的链。注意，所用的名字不能和已有的链、target同名。
Command	-X, --delete-chain
Example	iptables -X allowed
Explanation	删除指定的用户自定义链。这个链必须没有被引用，如果被引用，在删除之前你必须删除或者替换与之有关的规则。如果没有给出参数，这条命令将会删除默认表所有非内建的链。
Command	-P, --policy
Example	iptables -P INPUT DROP
Explanation	为链设置默认的target（可用的是 DROP 和 ACCEPT ，如果还有其它的可用，请告诉我），这个target称作策略。所有不符合规则的包都被强制使用这个策略。只有内建的链才可以使用规则。但内建的链和用户自定义链都不能被作为策略使用，也就是说不能象这样使用： iptables -P INPUT allowed （或者是内建的链）。
Command	-E, --rename-chain

Example	<code>iptables -E allowed disallowed</code>
Explanation	对自定义的链进行重命名，原来的名字在前，新名字在后。如上，就是把 <code>allowed</code> 改为 <code>disallowed</code> 。这仅仅是改变链的名字，对整个表的结构、工作没有任何影响。

在使用 `iptables` 时，如果必须的参数没有输入就按了回车，那么它就会给出一些提示信息：告诉你需要哪些参数等等。`iptables` 的选项 `-v` 用来显示 `iptables` 的版本，`-h` 给出语法的简短说明。。下面将要介绍的就是部分选项，还有它们的作用。

Table 6-3. Options

Option (选项)	<code>-v, --verbose</code> (详细的)
可用此选项的命令	<code>--list, --append, --insert, --delete, --replace</code>
Explanation (说明)	这个选项使输出详细化，常与 <code>--list</code> 连用。与 <code>--list</code> 连用时，输出中包括网络接口的地址、规则的选项、TOS掩码、字节和包计数器，其中计数器是以K、M、G（这里用的是10的幂而不是2的幂哦）为单位的。如果想知道到底有多少个包、多少字节，还要用到选项 <code>-x</code> ，下面会介绍。如果 <code>-v</code> 和 <code>--append</code> 、 <code>--insert</code> 、 <code>--delete</code> 或 <code>--replace</code> 连用， <code>iptables</code> 会输出详细的信息告诉你规则是如何被解释的、是否正确地插入等等。
Option	<code>-x, --exact</code> (精确的)
Commands used with	<code>--list</code>
Explanation	使 <code>--list</code> 输出中的计数器显示准确的数值，而不用K、M、G等估值。注意此选项只能和 <code>--list</code> 连用。
Option	<code>-n, --numeric</code> (数值)
Commands used with	<code>--list</code>
Explanation	使输出中的IP地址和端口以数值的形式显示，而不是默认的名字，比如主机名、网络名、程序名等。注意此选项也只能和 <code>--list</code> 连用。
Option	<code>--line-numbers</code>
Commands used with	<code>--list</code>
Explanation	又是一个只能和 <code>--list</code> 连用的选项，作用是显示出每条规则在相应链中的序号。这样你可以知道序号了，这对插入新规则很有用哦。
Option	<code>-c, --set-counters</code>
Commands used with	<code>--insert, --append, --replace</code>
Explanation	在创建或更改规则时设置计数器，语法如下： <code>--set-counters 20 4000</code> ，意思是让内核把包计数器设为20，把字节计数器设为4000。
Option	<code>--modprobe</code>
Commands used with	All
Explanation	此选项告诉 <code>iptables</code> 探测并装载要使用的模块。这是非常有用的一个选项，万一 <code>--modprobe</code> 命令不在搜索路径中，就要用到了。有了这个选项，在装载模块

时，即使有一个需要用到的模块没装载上，iptables也知道要去搜索。

6.4. Matches

这一节，我们会详细讨论一些matche，我把它们归为五类。第一类是*generic matches*（通用的匹配），适用于所有的规则；第二类是*TCP matches*，顾名思义，这只能用于TCP包；第三类是*UDP matches*，当然它只能用在UDP包上了；第四类是*ICMP matches*，针对ICMP包的；第五类比较特殊，针对的是状态（state），所有者（owner）和访问的频率限制（limit）等，它们已经被分到更多的小类当中，尽管它们并不是完全不同的。我希望这是一种大家都容易理解的分类。

6.4.1. 通用匹配

无论我们使用的是何种协议，也不管我们又装入了匹配的何种扩展，通用匹配都使可用的。也就是说，它们可以直接使用，而不需要什么前提条件，在后面你会看到，有很多匹配操作是需要其他的匹配作为前提的。

Table 6-4. Generic matches

Match	<code>-p, --protocol</code>
Example	<code>iptables -A INPUT -p tcp</code>
Explanation	<p>匹配指定的协议。指定协议的形式有以下几种：</p> <ol style="list-style-type: none">1、名字，不分大小写，但必须是在 /etc/protocols 中定义的。2、可以使用它们相应的整数值。例如，ICMP的值是1，TCP是6，UDP是17。3、缺省设置，ALL，相应数值是0，但要注意这只代表匹配TCP、UDP、ICMP，而不是 /etc/protocols 中定义的所有协议。4、可以是协议列表，以英文逗号为分隔符，如：<code>udp, tcp</code>5、可以在协议前加英文的感叹号表示取反，注意有空格，如：<code>--protocol ! tcp</code> 表示非tcp协议，也就是UDP和ICMP。可以看出这个取反的范围只是TCP、UDP和ICMP。
Match	<code>-s, --src, --source</code>
Example	<code>iptables -A INPUT -s 192.168.1.1</code>
Explanation	<p>以IP源地址匹配包。地址的形式如下：</p> <ol style="list-style-type: none">1、单个地址，如 <code>192.168.1.1</code>，也可写成 <code>192.168.1.1/255.255.255.255</code> 或 <code>192.168.1.1/32</code>2、网络，如 <code>192.168.0.0/24</code>，或 <code>192.168.0.0/255.255.255.0</code>3、在地址前加英文感叹号表示取反，注意空格，如 <code>--source !</code>

	192.168.0.0/24 表示除此地址外的所有地址 4、缺省是所有地址
Match	<code>-d, --dst, --destination</code>
Example	<code>iptables -A INPUT -d 192.168.1.1</code>
Explanation	以IP目的地址匹配包。地址的形式和 <code>-- source</code> 完全一样。
Match	<code>-i, --in-interface</code>
Example	<code>iptables -A INPUT -i eth0</code>
Explanation	以包进入本地所使用的网络接口来匹配包。要注意这个匹配操作只能用于 <i>INPUT</i> , <i>FORWARD</i> 和 <i>PREROUTING</i> 这三个链, 用在其他任何地方都会提示错误信息。指定接口有一下方法: 1、指定接口名称, 如: eth0、ppp0等 2、使用通配符, 即英文加号, 它代表字符数字串。若直接用一个加号, 即 <code>iptables -A INPUT -i +</code> 表示匹配所有的包, 而不考虑使用哪个接口。这也是不指定接口的默认行为。通配符还可以放在某一类接口的后面, 如: eth+表示所有Ethernet接口, 也就是说, 匹配所有从Ethernet接口进入的包。 3、在接口前加英文感叹号表示取反, 注意空格, 如: <code>-i ! eth0</code> 意思是匹配来自除eth0外的所有包。
Match	<code>-o, --out-interface</code>
Example	<code>iptables -A FORWARD -o eth0</code>
Explanation	以包离开本地所使用的网络接口来匹配包。使用的范围和指定接口的方法与 <code>--in-interface</code> 完全一样。
Match	<code>-f, --fragment</code>
Example	<code>iptables -A INPUT -f</code>
Explanation	用来匹配一个被分片的包的第二片或及以后的部分。因为它们不包含源或目的地址, 或ICMP类型等信息, 其他规则无法匹配到它, 所以才有这个匹配操作。要注意碎片攻击哦。这个操作也可以加英文感叹号表示取反, 但要注意位置, 如: <code>! -f</code> 。取反时, 表示只能匹配到没有分片的包或者是被分片的包的第一个碎片, 其后的片都不行。现在内核有完善的碎片重组功能, 可以防止碎片攻击, 所以不必使用取反的功能来防止碎片通过。如果你使用连接跟踪, 是不会看到任何碎片的, 因为在它们到达任何链之前就被处理过了。

6.4.2. 隐含匹配

这种匹配操作是自动地或隐含地装载入内核的。例如我们使用 `--protocol tcp` 时, 不需再装入任何东西就可以匹配只有IP包才有的一些特点。现在有三种隐含的匹配针对三种不同的协议, 即 *TCP matches*, *UDP matches*和 *ICMP matches*。它们分别包括一套只适用于相应协议的判别标准。相对于隐含匹配的是显式匹配, 它们必须使用 `-m`或 `--match`被明确地装载, 而不能是自动地或隐含地, 下一节会介绍到。

6.4.2.1. TCP matches

TCP matches只能匹配TCP包或流的细节，它们必须有**--protocol tcp**作为前提条件。

Table 6-5. TCP matches

Match	--sport, --source-port
Example	iptables -A INPUT -p tcp --sport 22
Explanation	<p>基于TCP包的源端口来匹配包，端口的指定形式如下：</p> <ol style="list-style-type: none"> 1、不指定此项，则暗示所有端口。 2、使用服务名或端口号，但名字必须是在 <i>/etc/services</i> 中定义的，因为 iptables从这个文件里查找相应的端口号。从这可以看出，使用端口号会使规则装入快一点儿，当然，可读性就差些了。但是如果你想写一个包含200条或更多规则的规则集，那你还是老老实实在地用端口号吧，时间是主要因素（在一台稍微慢点儿地机子上，这最多会有10秒地不同，但要是1000条、10000 条呢）。 3、可以使用连续的端口，如：--source-port 22:80这表示从22到80的所有端口，包括22和80。如果两个号的顺序反了也没关系，如：--source-port 80:22这和 --source-port 22:80的效果一样。 4、可以省略第一个号，默认第一个是0，如：--source-port :80表示从0到80的所有端口。 5、也可以省略第二个号，默认是65535，如：--source-port 22:表示从22到65535的所有端口 6、在端口号前加英文感叹号表示取反，注意空格，如：--source-port ! 22表示除22号之外的所有端口；--source-port ! 22:80表示从22到80（包括22和80）之外的所有端口。 <p>注意：这个匹配操作不能识别不连续的端口列表，如：--source-port ! 22, 36, 80 这样的操作是由后面将要介绍的多端口匹配扩展来完成的。</p>
Match	--dport, --destination-port
Example	iptables -A INPUT -p tcp --dport 22
Explanation	基于TCP包的目的地端口来匹配包，端口的指定形式和 --sport 完全一样。
Match	--tcp-flags
Example	
Explanation	<p>匹配指定的TCP标记。有两个参数，它们都是列表，列表内部用英文的逗号作分隔符，这两个列表之间用空格分开。第一个参数指定我们要检查的标记（作用就象掩码），第二个参数指定“在第一个列表中出现过的且必须被设为1（即状态是打开的）的”标记（第一个列表中其他的标记必须置0）。也就是说，第一个参数提供检查范围，第二个参数提供被设置的条件（就是哪些位置1）。这个匹配操作可以识别以下标记：<i>SYN, ACK, FIN, RST, URG, PSH</i>。另外还有两个词也可使用，就是ALL和NONE。顾名思义，ALL是指选定所有的标记，NONE是指未选定任何标记。这个匹配也可在参数前加英文的感叹号表示取反。例如：</p> <ol style="list-style-type: none"> 1、iptables -p tcp --tcp-flags SYN,FIN,ACK SYN表示匹配那些SYN标记被

	<p>设置而FIN和ACK标记没有设置的包，注意各标记之间只有一个逗号而没有空格。</p> <p>2、<code>--tcp-flags ALL NONE</code>匹配所有标记都未置1的包。</p> <p>3、<code>iptables -p tcp --tcp-flags ! SYN, FIN, ACK SYN</code>表示匹配那些FIN和ACK标记被设置而SYN标记没有设置的包，注意和例1比较一下。</p>
Match	<code>--syn</code>
Example	<code>iptables -p tcp --syn</code>
Explanation	<p>这个匹配或多或少算是ipchains时代的遗留物，之所以还保留它，是为了向后兼容，也是为了方便规则在iptables和ipchains间的转换。它匹配那些SYN标记被设置而ACK和RST标记没有设置的包，这和<code>iptables -p tcp --tcp-flags SYN, RST, ACK SYN</code>的作用毫无二样。这样的包主要用在TCP连接初始化时发出请求。如果你阻止了这样的包，也就阻止了所有由外向内的连接企图，这在一定程度上防止了一些攻击。但外出的连接不受影响，恰恰现在有很多攻击就利用这一点。比如有些攻击黑掉服务器之后安装会一些软件，它们能够利用已存的连接到达你的机子，而不要再新开一个端口。这个匹配也可用英文感叹号取反，如：<code>! --syn</code>用来匹配那些RST或ACK被置位的包，换句话说，就是状态为已建立的连接的包。</p>
Match	<code>--tcp-option</code>
Example	<code>iptables -p tcp --tcp-option 16</code>
Explanation	<p>根据匹配包。TCP选项是TCP头中的特殊部分，有三个不同的部分。第一个8位组表示选项的类型，第二个8位组表示选项的长度（这个长度是整个选项的长度，但不包含填充部分所占的字节，而且要注意不是每个TCP选项都有这一部分的），第三部分当然就是选项的内容了。为了适应标准，我们不必执行所有的选项，但我们可以查看选项的类型，如果不是我们所支持的，那就只是看看长度然后跳过数据部分。这个操作是根据选项的十进制值来匹配的，它也可以用英文感叹号取反。所有的选项都可在Internet Engineering Task Force里找到。</p>

6.4.2.2. UDP matches

UDP matches是在指定`--protocol UDP`时自动装入的。UDP是一种无连接协议，所以在它打开、关闭连接以及在发送数据时没有多少标记要设置，它也不需要任何类型的确认。数据丢失了，就丢失了（不会发送ICMP错误信息的）。这就说明UDP matches要比TCP matches少多了。即使UDP和ICMP是无连接协议，状态机制也可以很好的工作，就象在TCP上一样，这在前面对讨论过。

Table 6-6. UDP matches

Match	<code>--sport, --source-port</code>
Example	<code>iptables -A INPUT -p udp --sport 53</code>
Explanation	基于UDP包的源端口来匹配包，端口的指定形式和TCP matches中的 <code>--sport</code> 完全一样。
Match	<code>--dport, --destination-port</code>
Example	<code>iptables -A INPUT -p udp --dport 53</code>
Explanation	基于UDP包的目的地端口来匹配包，端口的指定形式和TCP matches中的 <code>--sport</code>

完全一样。

6.4.2.3. ICMP matches

ICMP协议也是无连接协议，ICMP包更是短命鬼，比UDP的还短。ICMP协议不是IP协议的下属协议，而是它的辅助者，其主要作用是报告错误和连接控制。ICMP包的头和IP的很相似，但又有很多不同。这个协议最主要的特点是它有很多类型，以应对不同的情况。比如，我们想访问一个无法访问的地址，就会收到一个ICMP host unreachable信息，它的意思是主机无法到达。在附录[ICMP类型](#)里有完整的ICMP类型列表。虽然有这么多类型，但只有一个 ICMP matche，这就足够对付它们了。这个matche是在指定**--protocol ICMP**时自动装入的。注意所有的通用匹配都可以使用，这样我们就可以匹配ICMP包的源、目地址。

Table 6-7. ICMP matches

Match	--icmp-type
Example	iptables -A INPUT -p icmp --icmp-type 8
Explanation	根据ICMP类型匹配包，类型的指定可以使用十进制数值或相应的名字，数值在RFC792中有定义，名字可以用 iptables --protocol icmp --help 查看，或者在附录 ICMP类型 中查找。这个匹配也可用英文感叹号取反，如： --icmp-type ! 8 就表示匹配除类型8之外的所有ICMP包。要注意有些ICMP 类型已经废弃不用了，还有一些可能会对无防护的主机带来“危险”，因为它们可能把包重定向到错误的地方。

6.4.3. 显式匹配

显式匹配必须用**-m**或**--match**装载，比如要使用状态匹配就必须使用**-m state**。有些匹配还需要指定协议，有些就不需要，比如连接状态就不要。这些状态是**NEW**（还未建立好的连接的第一个包），**ESTABLISHED**（已建立的连接，也就是已经在内核里注册过的），**RELATED**（由已经存在的、处于已建立状态的连接生成的新连接），等等。有些匹配还处在开发阶段，或者还只是为了说明iptables的强大能力。这说明不是所有的匹配一开始就是实用的，但以后你可能会用到它。随着iptables 新版本的发布，会有一些新的匹配可用。隐含匹配和显式匹配最大的区别就是一个是跟随协议匹配自动装载的，一个是显式装载的。

6.4.3.1. Limit match

这个匹配操作必须由**-m limit**明确指定才能使用。有了它的帮助，就可以对指定的规则的日志数量加以限制，以免你被信息的洪流淹没哦。比如，你可以事先设定一个限定值，当符合条件的包的数量不超过它时，就记录；超过了，就不记录了。我们可以控制某条规则在一段时间内的匹配次数（也就是可以匹配的包的数量），这样就能够减少DoS syn flood攻击的影响。这是它的主要作用，当然，还有很多其他作用（译者注：比如，对于某些不常用的服务可以限制连接数量，以免影响其他服务）。**limit match**也可以用英文感叹号取反，如：**-m limit ! --limit 5/s**表示在数量超过限定值后，所有的包都会被匹配。

（译者注：为了更好地理解这个匹配操作，我们通过一个比喻来解释一下。原文也做了类似地比喻，但我觉得对于初学者不易理解，故未采用。）**limit match**的工作方式就像一个单位大门口的保安，当有人要进入时，需要找他办理通行证。早上上班时，保安手里有一定数量的通行证，来一个人，就签发一个，当通行证用完后，再来人就进不去了，但他们不会等，

而是到别的地方去（在iptables里，这相当于一个包不符合某条规则，就会由后面的规则来处理，如果都不符合，就由缺省的策略处理）。但有个规定，每隔一段时间保安就要签发一个新的通行证。这样，后面来的人如果恰巧赶上，也就可以进去了。如果没有人来，那通行证就保留下来，以备来的人用。如果一直没人来，可用的通行证的数量就增加了，但不是无限增大的，最多也就是刚开始时保安手里有的那个数量。也就是说，刚开始时，通行证的数量是有限的，但每隔一段时间就有新的通行证可用。limit match有两个参数就对应这种情况，`--limit-burst`指定刚开始时有多少通行证可用，`--limit`指定要隔多长时间才能签发一个新的通行证。要注意的是，我这里强调的是“签发一个新的通行证”，这是以iptables的角度考虑的。在你自己写规则时，就要从这个角度考虑。比如，你指定了`--limit 3/minute --limit-burst 5`，意思是开始时有5个通行证，用完之后每20秒增加一个（这就是从iptables的角度看的，要是以用户的角度看，说法就是每一分钟增加三个或者每分钟只能过三个）。你要是想每20分钟过一个，只能写成`--limit 3/hour --limit-burst 5`，也就是说你要把时间单位凑成整的。

Table 6-8. Limit match options

Match	<code>--limit</code>
Example	<code>iptables -A INPUT -m limit --limit 3/hour</code>
Explanation	为limit match设置最大平均匹配速率，也就是单位时间内limit match可以匹配几个包。它的形式是一个数值加一个时间单位，可以是/second /minute /hour /day。默认值是每小时3次（用户角度），即3/hour，也就是每20分钟一次（iptables角度）。
Match	<code>--limit-burst</code>
Example	<code>iptables -A INPUT -m limit --limit-burst 5</code>
Explanation	这里定义的是limit match的峰值，就是在单位时间（这个时间由上面的一limit指定）内最多可匹配几个包（由此可见，--limit-burst的值要比一limit的大）。默认值是5。为了观察它是如何工作的，你可以启动“只有一条规则的脚本” <i>Limit-match.txt</i> ，然后用不同的时间间隔、发送不同数量的ping数据包。这样，通过返回的 <i>echo replies</i> 就可以看出其工作方式了。

6.4.3.2. MAC match

基于包的MAC源地址匹配包。到写这篇文章时，这个match还有一点限制（就是只能匹配MAC源地址匹），但今后定会有所发展，会更有用的。



注意，这个match是由`-m mac`装入的，而不是一些人想当然的`-m mac-source`，后者只是前者的选项而已。

Table 6-9. MAC match options

Match	<code>--mac-source</code>
Example	<code>iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01</code>
Explanation	基于包的MAC源地址匹配包，地址格式只能是XX:XX:XX:XX:XX:XX，当然它也可以用英文感叹号取反，如 <code>--mac-source ! 00:00:00:00:00:01</code> ，意思很简单了，就是除此之外的地址都可接受嘛。注意，因为 <i>MAC addresses</i> 只用于Ethernet类型的网络，所以这个match只能用于Ethernet接口。而且，它还只能在PREROUTING, FORWARD 和 INPUT链里使用。

6.4.3.3. Mark match

以包被设置的mark来匹配包，这个值只能由内核更改。前面曾经提到过，mark比较特殊，它不是包本身的一部分，而是在包穿越计算机的过程中由内核分配的和它相关联的一个字段。它可能被用来改变包的传输路径或过滤。时至今日，在linux里只有一种方法能设置mark，即iptables的**MARK** target，以前在ipchains里是**FWMARK** target。这就是为什么在高级路由里我们仍要参照**FWMARK**的原因。mark字段的值是一个无符号的整数，在32位系统上最大可以是4294967296（就是2的32次方），这足够用的了:)

Table 6-10. Mark match options

Match	<code>--mark</code>
Example	<code>iptables -t mangle -A INPUT -m mark --mark 1</code>
Explanation	以包被设置的mark值来匹配包，这个值是由下面将要介绍的 MARK target来设置的，它是一个无符号的整数。所有通过 <i>Netfilter</i> 的包都会被分配一个相关联的 <i>mark field</i> 。但要注意mark值可不是在任何情况下都能使用的，它只能在分配给它值的那台机子里使用，因为它只是由内核在内存里分配的和包相关的几个字节，并不属于包本身，所以我们不能在本机之外的路由器上使用。mark的格式是 <code>--mark value[/mask]</code> ，如上面的例子是没有掩码的，带掩码的例子如 <code>--mark 1/1</code> 。如果指定了掩码，就先把mark值和掩码取逻辑与，然后再和包的mark值比较。

6.4.3.4. Multiport match

多端口匹配扩展使我们能够在一条规则里指定不连续的多个端口，如果没有这个扩展，我们只能按端口来写规则了。其实这只是标准端口匹配的增强版罢了，使我们书写规则更方便而已。



注意：不能在一条规则里同时使用标准端口匹配和多端口匹配，如 `--sport 1024:63353 -m multiport --dport 21,23,80`。这条规则并不能像你想象的那样工作，但也不是不能工作，iptables会使用第一个合法的条件，那么这里多端口匹配就白写了:)

Table 6-11. Multiport match options

Match	<code>--source-port</code>
Example	<code>iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110</code>
Explanation	源端口多端口匹配，最多可以指定15个端口，以英文逗号分隔，注意没有空格。使用时必须有 <code>-p tcp</code> 或 <code>-p udp</code> 为前提条件。
Match	<code>--destination-port</code>
Example	<code>iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110</code>
Explanation	目的端口多端口匹配，使用方法和源端口多端口匹配一样，唯一的区别是它匹配的是目的端口。
Match	<code>--port</code>

Example	<code>iptables -A INPUT -p tcp -m multiport --port 22,53,80,110</code>
Explanation	同端口多端口匹配，意思就是它匹配的是那种源端口和目的端口是同一个端口的包，比如：端口80到端口80的包，110到110的包等。使用方法和源端口多端口匹配一样。

6.4.3.5. Owner match

基于包的生成者（也就是所有者，或称作拥有者，owner）的ID来匹配包，owner可以是启动进程的用户的ID，或用户所在的组的ID，或进程的ID，或会话的ID。这个扩展原本只是为了说明iptables可以做什么，现在发展到实用阶段了。但要注意，此扩展只能用在*OUTPUT*中，原因显而易见：我们几乎不可能得到发送端例程的ID的任何信息，或者在去往真正目的地的路上哪儿有路由。甚至在*OUTPUT*链里，这也不是十分可靠，因为有些包根本没有owner，比如*ICMP responses*，所以它们从不会被这个match抓到：)

Table 6-12. Owner match options

Match	<code>--uid-owner</code>
Example	<code>iptables -A OUTPUT -m owner --uid-owner 500</code>
Explanation	按生成包的用户的ID (UID) 来匹配外出的包。使用这个匹配可以做这样一些事，比如，阻止除root外的用户向防火墙外建立新连接，或阻止除用户http外的任何人使用HTTP端口发送数据。
Match	<code>--gid-owner</code>
Example	<code>iptables -A OUTPUT -m owner --gid-owner 0</code>
Explanation	按生成包的用户所在组的ID (GID) 来匹配外出的包。比如，我们可以只让属于network组的用户上Internet，而其他用户都不行；或者只允许http组的成员能从HTTP端口发送数据。
Match	<code>--pid-owner</code>
Example	<code>iptables -A OUTPUT -m owner --pid-owner 78</code>
Explanation	按生成包的进程的ID (PID) 来匹配外出的包。比如，我们可以只允许PID为94的进程（http进程当然不能是多线程的）使用http端口。这个匹配使用起来有一点难度，因为你要知道进程的ID号。当然，你也可以写一个小小的脚本，先从ps的输出中得到PID，再添加相应的规则，这儿有个例子 Pid-owner.txt 。
Match	<code>--sid-owner</code>
Example	<code>iptables -A OUTPUT -m owner --sid-owner 100</code>
Explanation	按生成包的会话的ID (SID) 来匹配外出的包。一个进程以及它的子进程或它的多个线程都有同一个SID。比如，所有的HTTPD进程的SID和它的父进程一样（最初的 HTTPD进程），即使HTTPD是多线程的（现在大部分都是，比如Apache和Roxen）也一样。这里有个脚本 Sid-owner.txt 可以反映出这一点。

6.4.3.6. State match

状态匹配扩展要有内核里的连接跟踪代码的协助，因为它是从连接跟踪机制中得到包的状态的。这样我们就可以了解连接所处的状态。它几乎适用于所有的协议，包括那些无状态的协议，如ICMP和UDP。针对每个连接都有一个缺省的超时值，如果连接的时间超过了这个值，那么这个连接的记录就会被从连接跟踪的记录数据库中删除，也就是说连接就不再存在了。这个match必须有**-m state**作为前提才能使用。状态机制的详细内容在章节[状态机制](#)中。

Table 6-13. State matches

Match	--state
Example	<code>iptables -A INPUT -m state --state RELATED, ESTABLISHED</code>
Explanation	指定要匹配包的状态，当前有4种状态可用： INVALID ， ESTABLISHED ， NEW 和 RELATED 。 INVALID 意味着这个包没有已知的流或连接与之关联，也可能是它包含的数据或包头有问题。 ESTABLISHED 意思是包是完全有效的，而且属于一个已建立的连接，这个连接的两端都已经有了数据发送。 NEW 表示包将要或已经开始建立一个新的连接，或者是这个包和一个还没有在两端都有数据发送的连接有关。 RELATED 说明包正在建立一个新的连接，这个连接是和一个已建立的连接相关的。比如， <i>FTP data transfer</i> ， <i>ICMP error</i> 和一个TCP或UDP连接相关。注意 NEW 状态并不在试图建立新连接的TCP包里寻找SYN标记，因此它不应该不加修改地用在只有一个防火墙或在不同的防火墙之间没有启用负载平衡的地方。具体如何使用，你就再看看章节 状态机制 吧：)

6.4.3.7. TOS match

根据 *TOS* 字段匹配包，必须使用 `-m tos` 才能装入。*TOS* 是 IP 头的一部分，其含义是 *Type Of Service*，由 8 个二进制位组成，包括一个 3 bit 的优先级子字段（现在已被忽略），4 bit 的 TOS 子字段和 1 bit 未用位（必须置 0）。它一般用来把当前流的优先权和需要的服务（比如，最小延时、最大吞吐量等）通知路由器。但路由器和管理员对这个值的处理相差很大，有的根本就不理会，而有的就会尽量满足要求。

Table 6-14. TOS matches

Match	--tos
Example	<code>iptables -A INPUT -p tcp -m tos --tos 0x16</code>
Explanation	<p>根据 <i>TOS</i> 字段匹配包。这个 match 常被用来 mark 包，以便后用，除此之外，它还常和 <code>iproute2</code> 或高级路由功能一起使用。它的参数可以是 16 进制数，也可以是十进制数，还可以是相应的名字（用 <code>iptables -m tos -h</code> 能查到）。到写这篇文章时，有以下参数可用：</p> <p>Minimize-Delay 16 (0x10)，要求找一条路径使延时最小，一些标准服务如 telnet、SSH、FTP-control 就需要这个选项。</p> <p>Maximize-Throughput 8 (0x08)，要求找一条路径能使吞吐量最大，标准服务 FTP-data 能用到这个。</p> <p>Maximize-Reliability 4 (0x04)，要求找一条路径能使可靠性最高，使用它的有 BOOTP 和 TFTP。</p> <p>Minimize-Cost 2 (0x02)，要求找一条路径能使费用最低，一般情况下使用这个选项的是一些视频音频流协议，如 RTSP (Real Time Stream Control Protocol)。</p> <p>Normal-Service 0 (0x00)，一般服务，没有什么特殊要求。</p>

6.4.3.8. TTL match

根据IP头里的 *TTL* (Time To Live, 即生存期) 字段来匹配包, 此必须由 **-m ttl** 装入。 *TTL field* 是一个字节 (8个二进制位), 一旦经过一个处理它的路由器, 它的值就减去1它的值。当该字段的值减为0时, 报文就被认为是不可转发的, 数据报就被丢弃, 并发送ICMP报文通知源主机, 不可转发的报文被丢弃。这也有两种情况, 一是传输期间生存时间为0, 使用类型为11代码是0的ICMP报文; 二是在数据报重组期间生存时间为0, 使用类型为11代码是1的ICMP报文。这个match只是根据 *TTL* 匹配包, 而对其不做任何更改, 所以在它之后可使用任何类型的match。

Table 6-15. TTL matches

Match	--ttl
Example	iptables -A OUTPUT -m ttl --ttl 60
Explanation	根据TTL的值来匹配包, 参数的形式只有一种, 就是十进制数值。它可以被用来调试你的局域网, 比如解决LAN内的主机到Internet上的主机的连接问题, 或找出 Trojan (Trojan) 可能的入口。这个match的用处相对有限, 但它其实是很有用的, 这就看你的想象力如何了。比如可以用它来发现那些TTL具有错误缺省值的机子 (这可能是实现TCP/IP栈功能的那个程序本身的错误, 或者是配置有问题)。

6.4.4. 针对非正常包的匹配

这个匹配没有任何参数, 也不需要显式地装载。注意这应该被看作是一个实验性的匹配, 它不总是能正常工作的, 对有些不正常的包 (unclean package, 就是所谓的脏包) 或问题, 它是视而不见的。这个match 试图匹配那些好象畸形或不正常的包, 比如包头错或校验和错, 等等。它可能常用来DROP错误的连接、检查有错的流, 但要知道这样做也可能会中断合法的连接。

6.5. Targets/Jumps

target/jump决定符合条件的包到何处去, 语法是 **--jump target** 或 **-j target**。(译者注: 本文中, 原作者把target细分为两类, 即Target和Jump。它们唯一的区别是jump的目标是在同一个表内的链, 而target的目标是具体的操作。) 我们会先接触到两个基本的target, 就是ACCEPT和DROP。

前面提到过用户自定义链要用到 **-N** 命令。下面我们在filter表中建一个名为 **tcp_packets** 的链:

```
iptables -N tcp_packets
```

然后再把它作为jump的目标:

```
iptables -A INPUT -p tcp -j tcp_packets
```

这样我们就会从INPUT链跳入 **tcp_packets** 链, 开始在 **tcp_packets** 中的旅行。如果到达了 **tcp_packets** 链的结尾 (也就是未被链中的任何规则匹配), 则会退到INPUT链的下一条规则继续它的旅行。如果在子链中被ACCEPT了, 也就相当于在父链中被ACCEPT了, 那么它不会再经过父链中的其他规则。但要注意这个包能被其他表的链匹配, 过程可查看章节 [表和链](#)。

target指定我们要对包做的操作, 比如DROP和ACCEPT, 还有很多, 我们后面会介绍。不同的target有不同的结果。一些target会使包停止前景, 也就是不再继续比较当前链中的其他规则或父链中的其他规则, 最好的例子就是DROP和ACCEPT。而另外一些target在对包做完操作之后, 包还会继续和其他的规则比较, 如LOG, ULOG和TOS。它们会对包进行记录、mangle, 然后让包通过, 以便匹配这条链中的其他规则。有了这样的target, 我们就可以对同一个包既改变它的TTL又改变它的TOS。有些target必须要有准确的参数(如TOS需要确定的数值), 有些就不是必须的, 但如果我们想指定也可以(如日志的前缀, 伪装使用的端口, 等等)。本节我们会尽可能全面地介绍每一个target。现在我们就来看看有哪几种target。

6.5.1. ACCEPT target

这个target没有任何选项和参数, 使用也很简单, 指定-j ACCEPT即可。一旦包满足了指定的匹配条件, 就会被ACCEPT, 并且不会再去匹配当前链中的其他规则或同一个表内的其他规则, 但它还要通过其他表中的链, 而且在那儿可能会百DROP也说不准哦。

6.5.2. DNAT target

这个target是用来做目的网络地址转换的, 就是重写包的目的IP地址。如果一个包被匹配了, 那么和它属于同一个流的所有的包都会被自动转换, 然后就可以被路由到正确的主机或网络。DNAT target是非常有用的。比如, 你的Web服务器在LAN内部, 而且没有可在Internet上使用的真实IP地址, 那就可以使用这个 target让防火墙把所有到它自己HTTP端口的包转发给LAN内部真正的Web服务器。目的地址也可以是一个范围, 这样的话, DNAT会为每一个流随机分配一个地址。因此, 我们可以用这个target做某种类型地负载平衡。

注意, DANT target只能用在nat表的PREROUTING和OUTPUT链中, 或者是被这两条链调用的链里。但还要注意的, 包含DANT target的链不能被除此之外的其他链调用, 如POSTROUTING。

Table 6-16. DNAT target

Option	--to-destination
Example	<code>iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10</code>
Explanation	指定要写入IP头的地址, 这也是包要被转发到的地方。上面的例子就是把所有发往地址15.45.23.67的包都转发到一段LAN使用的私有地址中, 即192.168.1.1到192.168.1.10。如前所述, 在这种情况下, 每个流都会被随机分配一个要转发到的地址, 但同一个流总是使用同一个地址。我们也可以只指定一个IP地址作为参数, 这样所有的包都被转发到同一台机子。我们还可以在地址后指定一个或一个范围的端口。比如: <code>--to-destination 192.168.1.1:80</code> 或 <code>--to-destination 192.168.1.1:80-100</code> 。SNAT的语法和这个target的一样, 只是目的不同罢了。要注意, 只有先用--protocol指定了TCP或UDP协议, 才能使用端口。

因为DNAT要做很多工作, 所以我要再罗嗦一点。我们通过一个例子来大致理解一下它是如何工作的。比如, 我想通过Internet连接发布我们的网站, 但是HTTP server在我们的内网里, 而且我们对外只有一个合法的IP, 就是防火墙那个对外的IP——\$INET_IP。防火墙还有一个内网的IP——\$LAN_IP, HTTP server的IP是\$HTTP_IP (这当然是内网的了)。为了完成我们

的设想, 要做的第一件事就是把下面的这个简单的规则加入到nat表的PREROUTING链中:

```
iptables -t nat -A PREROUTING --dst $INET_IP -p tcp --dport 80 -j DNAT \ --to-destination $HTTP_IP
```

现在, 所有从Internet来的、到防火墙的80端口去的包都会被转发(或称做被**DNAT**)到在内网的HTTP服务器上。如果你在Internet上试验一下, 一切正常吧。再从内网里试验一下, 完全不能用吧。这其实是路由的问题。下面我们来好好分析这个问题。为了容易阅读, 我们把在外网上访问我们服务器的那台机子的IP地址记为**\$EXT_BOX**。

1. 包从地址为**\$EXT_BOX**的机子出发, 去往地址为**\$INET_IP** 的机子。
2. 包到达防火墙。
3. 防火墙**DNAT**(也就是转发)这个包, 而且包会经过很多其他的链检验及处理。
4. 包离开防火墙向**\$HTTP_IP**前进。
5. 包到达HTTP服务器, 服务器就会通过防火墙给以回应, 当然, 这要求把防火墙作为HTTP到达**\$EXT_BOX**的网关。一般情况下, 防火墙就是HTTP服务器的缺省网关。
6. 防火墙再对返回包做**Un-DNAT**(就是照着DNAT的步骤反过来做一遍), 这样就好像是防火墙自己回复了那个来自外网的请求包。
7. 返回包好象没经过这么复杂的处理、没事一样回到**\$EXT_BOX**。

现在, 我们来考虑和HTTP服务器在同一个内网(这里是指所有机子不需要经过路由器而可以直接互相访问的网络, 不是那种把服务器和客户机又分在不同子网的情况)的客户访问它时会发生什么。我们假设客户机的IP为**\$LAN_BOX**, 其他设置同上。

1. 包离开**\$LAN_BOX**, 去往**\$INET_IP**。
2. 包到达防火墙。
3. 包被**DNAT**, 而且还会经过其他的处理。但是包没有经过**SNAT** 的处理, 所以包还是使用它自己的源地址, 就是**\$LAN_BOX**(译者注: 这就是IP 传输包的特点, 只根据目的地的不同改变目的地址, 但不因传输过程中要经过很多路由器而随着路由器改变其源地址, 除非你单独进行源地址的改变。其实这一步的处理和对外来包的处理是一样的, 只不过内网包的问题就在于此, 所以这里交待一下原因)。
4. 包离开防火墙, 到达HTTP服务器。
5. HTTP服务器试图回复这个包。它在路由数据库中看到包是来自同一个网络的一台机子, 因此它会把回复包直接发送到请求包的源地址(现在是回复包的目地地址), 也就是**\$LAN_BOX**。
6. 回复包到达客户机, 但它会很困惑, 因为这个包不是来自它访问的那台机子。这样, 它就会把这个包扔掉而去等待“真正”的回复包。

针对这个问题有个简单的解决办法, 因为这些包都要进入防火墙, 而且它们都去往需要做DNAT才能到达的那个地址, 所以我们只要对这些包做**SNAT**操作即可。比如, 我们来考虑上面的例子, 如果对那些进入防火墙而且是去往地址为**\$HTTP_IP**、端口为80的包做**SNAT**操作, 那

么这些包就好像是从\$LAN_IP来的了, 也就是说, 这些包的源地址被改为\$LAN_IP了。这样, HTTP服务器就会把回复包发给防火墙, 而防火墙会再对包做 Un-DNAT操作, 并把包发送到客户机。解决问题的规则如下:

```
iptables -t nat -A POSTROUTING -p tcp --dst $HTTP_IP --dport 80 -j SNAT \
--to-source $LAN_IP
```

要记住, 按运行的顺序`POSTROUTING`链是所有链中最后一个, 因此包到达这条链时, 已经被做过DNAT操作了, 所以我们在规则里要基于内网的地址\$HTTP_IP (包的目的地) 来匹配包。



警告: 我们刚才写的这条规则会对日志产生很大影响, 这种影响应该说是很不好。因为来自 Internet包在防火墙内先后经过了DNAT和SNAT处理, 才能到达HTTP服务器 (上面的例子), 所以HTTP服务器就认为包是防火墙发来的, 而不知道真正的源头是其他的IP。这样, 当它记录服务情况时, 所有访问记录的源地址都是防火墙的IP而不是真正的访问源。我们如果想根据这些记录来了解访问情况就不可能了。因此上面提供的“简单办法”并不是一个明智的选择, 但它确实可以解决“能够访问”的问题, 只是没有考虑到日志而已。

其他的服务也有类似的问题。比如, 你在LAN内建立了SMTP服务器, 那你就要设置防火墙以便能转发SMTP的数据流。这样你就创建了一个开放的SMTP中继服务器, 随之而来的就是日志的问题了。

一定要注意, 这里所说的问题只是针对没有建立DMZ或类似结构的网络, 并且内网的用户访问的是服务器的外网地址而言的。(译者注: 因为如果建立了DMZ, 或者服务器和客户机又被分在不同的子网里, 那就不需要这么麻烦了。因为所有访问的源头都不在服务器所在的网里, 所以就没必要做SNAT去改变包的源地址了, 从而记录也就不是问题了。如果内网客户是直接访问服务器的内网地址那就更没事了)

较好的解决办法是为你的LAN在内网建立一台单独的DNS服务器 (译者注: 这样, 内网的客户使用网站名访问HTTP服务器时, DNS就可以把它解析成内网地址。客户机就可以直接去访问HTTP服务器的内网地址了, 从而避免了通过防火墙的操作, 而且包的源地址也可以被HTTP服务器的日志使用, 也就没有上面说的日志问题了。), 或者干脆建立DMZ得了 (这是最好的办法, 但你要有钱哦, 因为用的设备多啊)。

对上面的例子应该考虑再全面些, 现在还有一个问题没解决, 就是防火墙自己要访问HTTP服务器时会发生什么, 能正常访问吗? 你觉得呢:) 很可惜, 现在的配置还是不行, 仔细想想就明白了。我们这里讨论的基础都是假设机子访问的是HTTP服务器的外网地址, 但这个外网地址其实就是防火墙对外的地址, 所以当防火墙访问这个外网地址时, 就是访问它自己。防火墙上如果有HTTP服务, 那客户机就会看到页面内容, 不过这不是它想看到的 (它想要的在DNAT上了), 如果没有HTTP服务, 客户就只能收到错误信息了。前面给出的规则之所以不起作用是因为从防火墙发出的请求包不会经过那两条链。还记得防火墙自己发出的包经过哪些链吧:) 我们要在nat表的OUTPUT链中添加下面的规则:

```
iptables -t nat -A OUTPUT --dst $INET_IP -p tcp --dport 80 -j DNAT \
--to-destination $HTTP_IP
```

有了最后这条规则, 一切都正常了。和HTTP服务器不在同一个网的机子能正常访问服务了, 和它在一个网内的机子也可以正常访问服务了, 防火墙本身也能正常访问服务了, 没有什么问题了。这种心情, 套用《大话西游》里的一句话, 就是“世界又清净了”。(不要说你不知道什么是《大话西游》)

我想大家应该能明白这些规则只说明了数据包是如何恰当地被DNAT和SNAT的。除此



之外, 在 filter表中还需要其他的规则 (在FORWARD链里), 以允许特定的包也能经过前面写的 (在POSTROUTING链和 OUTPUT链里的) 规则。千万不要忘了, 那些包在到达FORWARD链之前已经在PREROUTING链里被DNAT过了, 也就是说它们的目的地址已被改写, 在写规则时要注意这一点。

6.5.3. DROP target

顾名思义, 如果包符合条件, 这个target就会把它丢掉, 也就是说包的生命到此结束, 不会再向前走一步, 效果就是包被阻塞了。在某些情况下, 这个target会引起意外的结果, 因为它不会向发送者返回任何信息, 也不会向路由器返回信息, 这就可能会使连接的另一方的sockets因苦等回音而亡:) 解决这个问题的较好的办法是使用REJECT target, (译者注: 因为它在丢弃包的同时还会向发送者返回一个错误信息, 这样另一方就能正常结束), 尤其是在阻止端口扫描工具获得更多的信息时, 可以隐蔽被过滤掉的端口等等 (译者注: 因为扫描工具扫描一个端口时, 如果没有返回信息, 一般会认为端口未打开或被防火墙等设备过滤掉了)。还要注意如果包在子链中被DROP了, 那么它在主链里也不会再继续前进, 不管是在当前的表还是在其他表里。总之, 包死翘翘了。

6.5.4. LOG target

这个target是专门用来记录包地有关信息的。这些信息可能是非法的, 那就可以用来除错。LOG会返回包的有关细节, 如IP头的大部分和其他有趣的信息。这个功能是通过内核的日志工具完成的, 一般是syslogd。返回的信息可用dmesg阅读, 或者可以直接查看syslogd的日志文件, 也可以用其他的什么程序来看。LOG对调试规则有很大的帮助, 你可以看到包去了哪里、经过了什么规则的处理, 什么样的规则处理什么样的包, 等等。当你在生产服务器上调试一个不敢保证100%正常的规则集时, 用LOG代替DROP是比较好的 (有详细的信息可看, 错误就容易定位、解决了), 因为一个小小的语法错误就可能引起严重的连接问题, 用户可不喜欢这样哦。如果你想使用真正地扩展日志地话, 可能会对ULOG target有些兴趣, 因为它可以把日志直接记录到MySQL databases或类似的数据库中。



注意, 如果在控制台得到的信息不是你想要的, 那不是iptables或Netfilter的问题, 而是 syslogd 配置文件的事, 这个文件一般都是/etc/syslog.conf。有关这个问题的更多信息请查通过man syslog.conf查看。

LOG现在有5个选项, 你可以用它们指定需要的信息类型或针对不同的信息设定一些值以便在记录中使用。选项如下:

Table 6-17. LOG target options

Option	--log-level
Example	<code>iptables -A FORWARD -p tcp -j LOG --log-level debug</code>
Explanation	告诉iptables和 syslog使用哪个记录等级。记录等级的详细信息可以查看文件syslog.conf, 一般来说有以下几种, 它们的级别依次是: debug, info, notice, warning, warn, err, error, crit, alert, emerg, panic 。其中, error和err、warn和warning、panic和emerg分别是同义词, 也就是说作用完全一样的。注意这三种级别是不被赞成使用的, 换句话说, 就是不要使用它们 (因为信息量太大)。信息级别说明了被记录信息所反映的问题的严重程度。所有信息都是通过内核的功能被记录的, 也就是说, 先在文件

	syslog.conf里设置 kern.=info /var/log/iptables , 然后再让所有关于iptables的LOG信息使用级别info, 就可以把所有的信息存入文件/var/log/iptables内。注意, 其中也可能会有其他的信息, 它们是内核中使用info 这个等级的其他部分产生的。有关日志的详细信息, 我建议你看看 syslog 和 syslog.conf 的帮助, 还有HOWTO, 等等。
Option	--log-prefix
Example	iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"
Explanation	告诉 iptables 在记录的信息之前加上指定的前缀。这样和 grep 或其他工具一起使用时就容易追踪特定的问题, 而且也方便从不同的规则输出。前缀最多能有29个英文字符, 这已经是包括空白字符和其他特殊符号的总长度了。
Option	--log-tcp-sequence
Example	iptables -A INPUT -p tcp -j LOG --log-tcp-sequence
Explanation	把包的TCP序列号和其他日志信息一起记录下来。TCP序列号可以唯一标识一个包, 在重组时也是用它来确定每个分组在包里的位置。注意, 这个选项可能会带来危险, 因为这些记录被未授权的用户看到的话, 可能会使他们更容易地破坏系统。其实, 任何iptables的输出信息都增加了这种危险。(译者注: 现在, 我深刻理解了什么是“言多必失”, 什么是“沉默是金”)
Option	--log-tcp-options
Example	iptables -A FORWARD -p tcp -j LOG --log-tcp-options
Explanation	记录TCP包头中的字段大小不变的选项。这对一些除错是很有价值的, 通过它提供的信息, 可以知道哪里可能出错, 或者哪里已经出了错。
Option	--log-ip-options
Example	iptables -A FORWARD -p tcp -j LOG --log-ip-options
Explanation	记录IP包头中的字段大小不变的选项。这对一些除错是很有价值的, 还可以用来跟踪特定地址的包。

6.5.5. MARK target

用来设置mark值, 这个值只能在本地的mangle表里使用, 不能用在其他任何地方, 就更不用说路由器或另一台机子了。因为mark比较特殊, 它不是包本身的一部分, 而是在包穿越计算机的过程中由内核分配的和它相关联的一个字段。它可以和本地的高级路由功能联用, 以使不同的包能使用不同的队列要求, 等等。如果你想在传输过程中也有这种功能, 还是用TOS target吧。有关高级路由的更多信息, 可以查看[Linux Advanced Routing and Traffic Control HOW-TO](#)。

Table 6-18. MARK target options

Option	--set-mark
Example	iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2
Explanation	设置mark值, 这个值是一个无符号的整数。比如, 我们对一个流或从某台机子发出的所有的包设置了mark值, 就可以利用高级路由功能来对它们进行流量控制等操作了。

6.5.6. MASQUERADE target

这个target和SNAT target的作用是一样的，区别就是它不需要指定`--to-source`。MASQUERADE是被专门设计用于那些动态获取IP地址的连接，比如，拨号上网、DHCP连接等。如果你有固定的IP地址，还是用SNAT target吧。

伪装一个连接意味着，我们自动获取网络接口的IP地址，而不使用`--to-source`。当接口停用时，MASQUERADE不会记住任何连接，这在我们kill掉接口时是有很大大好处的。如果我们使用SNAT target，连接跟踪的数据是被保留下来的，而且时间要好几天哦，这可是要占用很多连接跟踪的内存的。一般情况下，这种处理方式对于拨号上网来说是较好的（这有利于已有那连接继续使用）。如果我们被分配给了一个不同于前一次的IP，不管怎样已有的连接都要丢失，但或多或少地还是有一些连接记录被保留了（真是白痴，是吧）。

即使你有静态的IP，也可以使用MASQUERADE，而不用SNAT。不过，这不是被赞成的，因为它会带来额外的开销，而且以后还可能引起矛盾，比如它也许会影响你的脚本，使它们不能用。

注意，MASQUERADE和SNAT一样，只能用于nat表的 POSTROUTING链，而且它只有一个选项（不是必需的）：

Table 6-19. MASQUERADE target

Option	<code>--to-ports</code>
Example	<code>iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000</code>
Explanation	在指定TCP或UDP的前提下，设置外出包能使用的端口，方式是单个端口，如 <code>--to-ports 1025</code> ，或者是端口范围，如 <code>--to-ports 1024-3000</code> 。注意，在指定范围时要使用连字号。这改变了SNAT中缺省的端口选择，详情请查看 SNAT target 。

6.5.7. MIRROR target

这个target是实验性的，它只是一个演示而已，不建议你使用它，因为它可能引起循环，除此之外，还可能引起严重的DoS。这个target的作用是颠倒IP头中的源目地址，然后再转发包。这会引起很有趣的事，一个骇客最后攻破的可能就是他自己的机子。看来，使用这个target至少可以使我们的机子更强壮:) 我们如果对机子A的80端口使用了MIRROR，会发生什么呢？假设有来自yahoo.com的机子B 想要访问A的HTTP服务，那他得到的将是yahoo的主页，因为请求是来自yahoo的。

注意，MIRROR只能用在INPUT、FORWARD、PREROUTING链和被它们调用的自定义链中。还要注意，如果外出的包是因 MIRROR target发出的，则它们是不会被filter、nat或mangle表内的链处理的，这可能引起循环或其他问题。比如，一台机子向另一台配置了MIRROR且TTL值为255的机子发送一个会被认为是欺骗的数据包，同时这台机子也欺骗自己的数据包，以使它被认为好像是来自第三个使用了MIRROR 的机子。这样，那个包就会不间断地往来很多次，直到TTL为0。如果两台机子之间只有一个路由器，这个包就会往返240-255次。对骇客来说，这是不坏的情况，因为他只要发送一个1500字节的数据（也就是一个包），就可以消耗你的连接的380K字节。对于骇客或者叫做脚本小子（不管我们把他们称作什么）来说，这可是很理想的情况。

6.5.8. QUEUE target

这个target为用户空间的程序或应用软件管理包队列。它是和iptables之外的程序或工具协同使用的, 包括网络计数工具, 高级的数据包代理或过滤应用, 等等。讨论程序的编码已超出了本文的范围。即使讨论, 也要花很多时间, 而且在这样一篇文章之内也无法说清有关Netfilter和iptables的编程。具体的信息请查看[Netfilter Hacking HOW-TO](#)。

6.5.9. REDIRECT target

在防火墙所在的机子内部转发包或流到另一个端口。比如, 我们可以把所有去往端口HTTP的包REDIRECT到HTTP proxy (例如squid), 当然这都发生在我们自己的主机内部。本地生成的包都会被映射到127.0.0.1。换句话说, 这个target把要转发的包的地址改写为我们自己机子的IP。我们在做透明代理 (LAN内的机子根本不需要知道代理的存在就可以正常上网) 时, 这个target可是起了很大作用的。

注意, 它只能用在nat表的PREROUTING、OUTPUT链和被它们调用的自定义链里。 REDIRECT只有一个选项:

Table 6-20. REDIRECT target

Option	--to-ports
Example	<code>iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080</code>
Explanation	<p>在指定TCP或UDP协议的前提下, 定义目的端口, 方式如下:</p> <ol style="list-style-type: none">1、不使用这个选项, 目的端口不会被改变。2、指定一个端口, 如--to-ports 80803、指定端口范围, 如--to-ports 8080-8090

6.5.10. REJECT target

REJECT和DROP基本一样, 区别在于它除了阻塞包之外, 还向发送者返回错误信息。现在, 此target还只能用在INPUT、FORWARD、OUTPUT和它们的子链里, 而且包含 REJECT的链也只能被它们调用, 否则不能发挥作用。它只有一个选项, 是用来控制返回的错误信息的种类的。虽然有很多种类, 但如果你有TCP/IP方面的基础知识, 就很容易理解它们。

Table 6-21. REJECT target

Option	--reject-with
Example	<code>iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset</code>
Explanation	<p>告诉REJECT target应向发送者返回什么样的信息。一旦包满足了设定的条件, 就要发送相应的信息, 然后再象DROP一样无情地抛弃那些包。可用的信息类型有: 1、icmp-net-unreachable 2、icmp-host-unreachable 3、icmp-</p>

port-unreachable 4、icmp-proto-unreachable 5、icmp-net-prohibited 6、icmp-host-prohibited。其中缺省的是**port-unreachable**。你可以在附录 [ICMP类型](#) 中看到更多的信息。还有一个类型——**echo-reply**，它只能和匹配 ICMP ping包的规则联用。最后一个类型是**tcp-reset**，（显然，只能用于TCP协议）它的作用是告诉REJECT返回一个TCP RST包（这个包以文雅的方式关闭TCP连接，有关它的详细信息在[RFC 793 - Transmission Control Protocol](#)里）给发送者。正如iptables的 man page中说的，tcp-reset主要用来阻塞身份识别探针（即113/tcp，当向被破坏的邮件主机发送邮件时，探针常被用到，否则它不会接受你的信）。

6.5.11. RETURN target

顾名思义，它使包返回上一层，顺序是：子链——>父链——>缺省的策略。具体地说，就是若包在子链中遇到了RETURN，则返回父链的下一条规则继续进行条件的比较，若是在父链（或称主链，比如INPUT）中遇到了RETURN，就要被缺省的策略（一般是ACCEPT或DROP）操作了。（译者注：这很象C语言中函数返回值的情况）

我们来举个例子说明一下，假设一个包进入了INPUT链，匹配了某条target为**--jump EXAMPLE_CHAIN**规则，然后进入了子链**EXAMPLE_CHAIN**。在子链中又匹配了某条规则，恰巧target是**--jump RETURN**，那包就返回INPUT链了。如果在INPUT链里又遇到了**--jump RETURN**，那这个包就要交由缺省的策略来处理了。

6.5.12. SNAT target

这个target是用来做源网络地址转换的，就是重写包的源IP地址。当我们有几个机子共享一个Internet连接时，就能用到它了。先在内核里打开ip转发功能，然后再写一个**SNAT**规则，就可以把所有从本地网络出去的包的源地址改为Internet连接的地址了。如果我们不这样做而是直接转发本地网的包的话，Internet上的机子就不知道往哪儿发送应答了，因为在本地网里我们一般使用的是IANA组织专门指定的一段地址，它们是不能在Internet上使用的。**SNAT target**的作用就是让所有从本地网出发的包看起来都是从一台机子发出的，这台机子一般就是防火墙。

SNAT只能用在nat表的POSTROUTING链里。只要连接的第一个符合条件的包被**SNAT**了，那么这个连接的其他所有的包都会自动地被**SNAT**，而且这个规则还会应用于这个连接所在流的所有数据包。

Table 6-22. SNAT target

Option	--to-source
Example	iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000
Explanation	指定源地址和端口，有以下几种方式： 1、单独的地址。 2、一段连续的地址，用连字符分隔，如194.236.50.155-194.236.50.160，这样可以实现负载均衡。每个流会被随机分配一个IP，但对于同一个流使用的是

同一个IP。

3、在指定 `-p tcp` 或 `-p udp` 的前提下，可以指定源端口的范围，如 194.236.50.155:1024-32000，这样包的源端口就被限制在1024-32000了。

注意，如果可能，iptables总是想避免任何的端口变更，换句话说，它总是尽力使用建立连接时所用的端口。但是如果两台机子使用相同的源端口，iptables 将会把他们的其中之一映射到另外的一个端口。如果没有指定端口范围，所有的在512以内的源端口会被映射到512以内的另一个端口，512和1023之间的将会被映射到 1024内，其他的将会被映射到大于或对于1024的端口，也就是说同范围映射。还要注意，这种映射和目的端口无关。因此，如果客户想和防火墙外的HTTP服务器联系，它是不会被映射到FTP control所用的端口的。

6.5.13. TOS target

TOS是用来设置IP头中的Type of Service字段的。这个字段长一个字节，可以控制包的路由情况。它也是iproute2及其子系统可以直接使用的字段之一。值得注意的是，如果你有几个独立的防火墙和路由器，而且还想在它们之间利用包的头部来传递路由信息，TOS是唯一的办法。前面说过，MARK是不能用来传递这种信息的。如果你需要为某个包或流传递路由信息，就要使用TOS字段，它也正是为这个而被开发的。

Internet上有很多路由器在这一方面并没有做好工作，因此，在发送包之前改变其TOS没有什么大用处。最好的情况是路由器根本不理它，最坏的情况是路由器会根据TOS处理，但都是错误的。然而，如果你是在一个很大的WAN或LAN里，而且有很多路由器，TOS还是能有很好的作为的。总的来说，基于TOS的值给包以不同的路由和参数还是可能的，即使在网络里是受限制的（译者注：大不了不起作用就是了）。



TOS只能用来设置具体的或者说是特定的值（这些预定义的值在内核源码的include文件——Linux/ip.h中），原因是很多的，但不管怎么说，你不要使用其他的值就是了。当然，我们也有办法突破这个限制，就是使用一个名为FTOS的patch，你可在由Matthew G. Marsh维护的站点 [Paksecured Linux Kernel patches](#)得到它，但要小心使用哦。除了非常特殊、极端的情况，我们是不应该使用预定义以外的值的。



注意，这个target只能在mangle表内使用。



还要注意，在一些老版（1.2.2或更低）的iptables中包含的这个target在设置了TOS之后，不会调整包的校验和，这样包会被认为是错误的并要求重发。而且，这很可能会导致更多的mangle操作，从而使整个连接无法工作。（译者注：好在我们现在不会再用这么老的版本了：））

TOS target只有一个选项：

Table 6-23. TOS target

Option	<code>--set-tos</code>
Example	<code>iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos</code>

	0x10
Explanation	<p>设置TOS的值，值的形式可以是名字或者相应的数值（十进制或16进制的）。一般情况下，建议你使用名字而不使用数值形式，因为以后这些数值可能会有所改变，而名字一般是固定的。TOS字段有8个二进制位，所以可能的值是0-255（十进制）或0x00-0xFF（16进制）。如前所述，你最好使用预定义的值，它们是：</p> <ol style="list-style-type: none"> 1、Minimize-Delay 16 (0x10)，要求找一条路径使延时最小，一些标准服务如telnet、SSH、FTP- control 就需要这个选项。 2、Maximize-Throughput 8 (0x08)，要求找一条路径能使吞吐量最大，标准服务FTP-data能用到这个。 3、Maximize-Reliability 4 (0x04)，要求找一条路径能使可靠性最高，使用它的有BOOTP和TFTP。 4、Minimize-Cost 2 (0x02)，要求找一条路径能使费用最低，一般情况下使用这个选项的是一些视频音频流协议，如RTSP (Real Time Stream Control Protocol) 。 5、Normal-Service 0 (0x00)，一般服务，没有什么特殊要求。这个值也是大部分包的缺省值。 <p>完整的列表可以通过命令iptables -j TOS -h得到。在1.2.5版时，这个列表就已经是完整的了，而且会保持很长一段时间。</p>

6.5.14. TTL target



这个target需要patch-o-matic里的名为TTL的patch, 可从 <http://www.netfilter.org> 获得。此站点的FAQ是开始学习iptables和Netfilter的好地方。

TTL可以修改IP头中Time To Live字段的值。它有很大的作用，我们可以把所有外出包的Time To Live值都改为一样的，比如64，这是Linux的默认值。有些ISP不允许我们共享连接（他们可以通过TTL的值来区分是不是有多个本子使用同一个连接），如果我们把TTL都改为一样的值，他们就不能再根据TTL来判断了。

关于任何设置Linux的TTL默认值，请参阅附录 [其他资源和链接](#) 内的 [ip-sysctl.txt](#)。

TTL只能在mangle表内使用，它有3个选项：

Table 6-24. TTL target

Option	--ttl-set
Example	iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64
Explanation	设置TTL的值。这个值不要太大，也不要太小，大约64就很好。值太大会影响网络，而且有点不道德，为什么这样说呢？如果有些路由器的配置不太正确，

	包的TTL又非常大，那它们就会在这些路由器之间往返很多次，值越大，占用的带宽越多。这个target就可以被用来限制包能走多远，一个比较恰当的距离是刚好能到达DNS服务器。
Option	<code>--ttl-dec</code>
Example	<code>iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-dec 1</code>
Explanation	设定TTL要被减掉的值，比如 <code>--ttl-dec 3</code> 。假设一个进来的包的TTL是53，那么当它离开我们这台机器时，TTL就变为49了。为什么不是50呢？因为经过我们这台机器，TTL本身就要减1，还要被TTL target再减3，当然总共就是减去4了。使用这个 target可以限制“使用我们的服务的用户”离我们有多远。比如，用户总是使用比较近的DNS，那我们就可以对我们的DNS服务器发出的包进行几个 <code>--ttl-dec</code> 。（译者注：意思是，我们只想让距离DNS服务器近一些的用户访问我们的服务）当然，用 <code>--set-ttl</code> 控制更方便些。
Option	<code>--ttl-inc</code>
Example	<code>iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1</code>
Explanation	设定TTL要被增加的值，比如 <code>--ttl-inc 4</code> 。假设一个进来的包的TTL是53，那么当它离开我们这台机器时，TTL应是多少呢？答案是56，原因同 <code>--ttl-dec</code> 。使用这个选项可以使我们的防火墙更加隐蔽，而不被trace-routes发现，方法就是设置 <code>--ttl-inc 1</code> 。原因应该很简单了，包每经过一个设备，TTL就要自动减1，但在我们的防火墙里这个1又被补上了，也就是说，TTL的值没变，那么trace-routes就会认为我们的防火墙是不存在的。Trace-routes让人又爱又恨，爱它是因为在连接出问题，它可以给我们提供极有用的信息，告诉我们哪里有毛病；恨它是由于它也可以被黑客或骇客用来收集目标机器的资料。怎么使用它呢？这里有个很好的例子，请看脚本 Ttl-inc.txt 。

6.5.15. ULOG target

ULOG可以在用户空间记录被匹配的包的信息，这些信息和整个包都会通过netlink socket被多播。然后，一个或多个用户空间的进程就会接受它们。换句话说，ULOG是至今iptables和Netfilter下最成熟、最完善的日志工具，它包含了很多更好的工具用于包的记录。这个target可以是我们将信息记录到MySQL或其他数据库中。这样，搜索特定的包或把记录分组就很方便了。你可以在[ULOGD project page](#)里找到ULOGD用户空间的软件。


Table 6-25. ULOG target

Option	<code>--ulog-nlgroup</code>
Example	<code>iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-nlgroup 2</code>
Explanation	指定向哪个netlink组发送包，比如 <code>--ulog-nlgroup 5</code> 。一个有32个netlink组，它们被简单地编号位1-32。默认值是1。
Option	<code>--ulog-prefix</code>
Example	<code>iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-prefix "SSH connection attempt: "</code>
Explanation	指定记录信息的前缀，以便于区分不同的信息。使用方法和 LOG的prefix一样，只是长度可以达到32个字符。
Option	<code>--ulog-cprange</code>
Example	<code>iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-cprange 100</code>
Explanation	指定每个包要向“ULOG在用户空间的代理”发送的字节数，如 <code>--ulog-cprange</code>

	100, 表示把整个包的前100个字节拷贝到用户空间记录下来, 其中包含了这个包头, 还有一些包的引导数据。默认值是0, 表示拷贝整个包, 不管它有多大。
Option	--ulog-qthreshold
Example	iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-qthreshold 10
Explanation	告诉ULOG在向用户空间发送数据以供记录之前, 要在内核里收集的包的数量 (译者注: 就像集装箱), 如--ulog-qthreshold 10。这表示先在内核里积聚10个包, 再把它们发送到用户空间里, 它们会被看作同一个netlink的信息, 只是由好几部分组成罢了。默认值是1, 这是为了向后兼容, 因为以前的版本不能处理分段的信息。

Chapter 7. 防火墙配置实例 rc.firewall

在本章里, 我们将要建立一个防火墙, 并且详细地说明了如何去阅读、理解它。在这个例子中, 我们使用的是最基本的配置, 对其工作方式和我们在里面做了些什么都有深入的解释。这个例子应该能在某些方面给你提供基本的思路, 比如, 如何解决不同的问题 (当然是网络方面的), 再如, 在真正把脚本应用于工作之前应该考虑些什么, 等等。对本例中的变量值做些修改就可能在实际的网络中使用, 但不建议你这样做, 因为你的网络配置和我在例子中使用的情況可能不一样哦。但只要你有这个基本的防火墙规则集, 很可能只需要少量的调整就可以把它用于实际了。

可能有效率更高的方法来建立规则集, 但这个脚本就是为易读而写的, 所以每个人都能理解它, 即使没有多少BASH脚本编程的知识。

7.1. 关于rc.firewall

好, 既然你能从头看到这儿, 就说明你已经做好一切准备来检查这个脚本了。例子 [rc.firewall.txt](#) (代码在附录 [示例脚本的代码](#)里) 很大, 但没有多少注释。我建议你先大致看看它的内容, 留个印象, 再来仔细地阅读本章 (要有耐心哦)。

7.2. rc.firewall详解

7.2.1. 参数配置

本小节要对照着[rc.firewall脚本代码](#)来看。

[rc.firewall.txt](#)的第一小节是配置选项, 包含的都是一些至关重要的信息, 它们是随着你的网络的不同而改变的。比如, 每个网络的IP地址都不一样, 所有要把它放在这儿。`$INET_IP`的值应该是在Internet上能使用的才可以, 如果你有`$INET_IP`的话。如果没有, 你就要看看[rc.DHCP.firewall.txt](#)这种配置方法了, 里面有很多有趣的东西。变量 `$INET_IFACE`应该指向连接Internet的真实设备, 比如eth0、eth1、ppp0、tr0等等。

这个脚本里没有包含任何DHCP或PPPoE的选项, 所以这两节是空白的。其他空白的部分, 也是

这样的原因。之所以保留这些空白,是为了更容易区分这些结构相同而内容不同的脚本。如果你需要这些部分,可以从其他脚本拷贝过来,或者你自己写了:)

*Local Area Network*小节包含的是LAN必须的信息,如连接到LAN的网卡的IP、LAN所用的地址段等。

Localhost configuration小节里的信息在99%的情况下都不要改变,因为我们总是使用127.0.0.1作为地址,也总是把接口命名为lo。紧随其后的是IPTables Configuration,里面只有一个变量,即`$IPTABLES`。它指定的是iptables程序的准确位置,如果是自己编译安装的话,一般都是`/usr/local/sbin/iptables`。但更多的发行版都把程序放在另外的地方,如`/usr/sbin/iptables`,等等。

7.2.2. 外部模块的装载

首先,我们要使用命令`/sbin/depmod -a`使module dependencies files保持最新,然后,再装载脚本需要的模块。我们应该始终避免装入不需要的模块,如果可能,还要尽力避免装入无所事事的模块,除非你确实需要它们。这样做主要是为了安全,因为每增加一个模块都要花费额外的努力以增加新的规则(这样就容易出漏洞哦)。比如,如果你想支持LOG、REJECT和MASQUERADE target,不要把相应的功能静态地编译进内核,我们使用以下模块来完成:

```
/sbin/insmod ipt_LOG
/sbin/insmod ipt_REJECT
/sbin/insmod ipt_MASQUERADE
```



注意,本文使用的脚本都是用类似命令装入模块,这可能会引起装载失败(有错误信息显示)。原因是多方面的,但如果较基本的模块也失败的话,那最大的可能是哪个模块或相应的功能已被静态地编译进内核了。进一步的信息可以看看附录[常见问题与解答](#)中的[模块装载问题](#)。

接下来的一行是装载ipt_owner模块,它的作用是“只允许特定的用户创建特定的连接”。在这个例子中,我没有使用到它,但你可能会用到。比如,你可能只允许root建立FTP和HTTP连接访问redhat.com,而其他用户都不可以。你也可以只允许你自己使用的用户名和root才能访问Internet,这样别人会很烦的,但你的安全性在某些方面会有所提高哦,比如,把你当作发起攻击的跳板的情况。关于ipt_owner的更多信息,可以看看章节[规则是如何练成的](#)里的[Owner match](#)。

在这儿我们也可以为状态匹配安装扩展模块。状态匹配和连接跟踪的所有扩展模块的名字都是这样的: `ip_conntrack_*`和`ip_nat_*`。连接跟踪的helper是一些特殊的模块,正是它们告诉了内核怎样恰当地跟踪特殊的连接。没有这些helper,内核在处理特殊连接的时候,就不知道该查看些什么东西。NAT helper就是连接跟踪helper的扩展,它会告诉内核在包里找什么、如何转换它们,这样连接才能真正工作起来。比如,FTP是一个复杂的协议,它利用包的有效数据部分来发送连接信息。如果一台需要被NAT的机器(译者注:也就是说,机器在一个内网里)连接Internet上的FTP服务器,它就会把自己的内网IP地址放在包的数据区内发送出去,以使FTP服务器能连接到那个地址。但私有地址不能在LAN外使用,所以FTP服务器不知道用它做什么,连接就会断掉了。FTP NAT helper能完成这些连接中所有的地址转换工作,因此FTP服务器就知道该往哪儿连了。同样的事情也发生在DCC的文件传输(这里指的是发送)和聊天上,为了建立连接,IP地址和端口都需要利用IRC协议的数据区发送,而且还要做一些转换工作。没有这些helper的话,FTP和IRC只有一部分工作是正常的,但另一部分根本就无法工作。例如,你可以通过DCC接收文件,但就是不能发送。这个问题的原因在于DCC是如何建立连接的。当DCC想发送文件时,会告诉接收者你要发送文件,并让它知道要连接到

什么地方。如果没有helper, 这个DCC连接最终会断开, 因为接收者收到的是内网的地址。这样, 当它按那个地址连接时, 其实就连到和它在同一内网的机子了。那为什么可以接收呢? 因为发送者给你的是可在 Internet上使用的IP地址 (大部分情况下, IRC服务器都有真实的IP地址)。



如果你在通过防火墙使用mIRC DCC时遇到了问题, 但和其他IRC客户沟通很正常, 看看附录[常见问题与解答](#)里的 [关于mIRC DCC的问题](#) 吧。

在这个例子中, 我们在这儿装载支持FTP和IRC协议的模块。有关连接跟踪和nat的详细信息, 请查看附录 [常见问题与解答](#)。在patch-o-matic中, 还有H.323 conntrack helper等其他象NAT helper的模块。但为了使用它们, 你需要使用patch-o-matic提供的补丁, 还需要编译内核。详细的操作信息可以查看章节[准备阶段](#)。



注意, 为了能对FTP和IRC协议做网络地址转换, 需要装载ip_nat_ftp和ip_nat_irc。在装载NAT模块之前, 你还要载入ip_conntrack_ftp和ip_conntrack_irc模块。NAT模块和conntrack模块以相同的方式被使用, 但NAT模块使我们能对这两个协议做NAT。

7.2.3. proc的设置

我们可以使用下面的语句打开IP转发功能 (IP forwarding) :

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```



注意, 何时何地打开这个功能才算合适是值得好好考虑的一个问题。在本文所用的脚本中, 我都是在创建IP过滤器 (在本文里就是指iptables的过滤规则) 之前打开它的。这可能引起这样一种情况, 就是在一小段时间内 (时间的长短随脚本的复杂程度和机子的性能高低而变化, 可能只有一毫秒, 也可能会长达一分钟), 防火墙可以转发任何包 (译者注: 因为这时防火墙的过滤规则还没有被装入)。这种情况又会导致安全方面的问题, 不怀好意的人可能会趁此通过防火墙破坏我们的网络。也就是说, 我们应该在创建所有防火墙的规则之后再打开IP转发功能, 我这样做只是为了保证所有脚本的兼容性。 (译者注: 我们在实际应用中一定要注意这一点, 尽量不要先开IP转发功能)

万一你使用的是SLIP、PPP或DHCP, 也就是说你是动态获取IP的, 那还要用下面的命令打开ip_dynaddr:

```
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
```

如果你还要打开其他的proc选项, 也是用类似的方法, 但有关那些选项的具体介绍以不是本文的内容, 你可以看看其他相关的文章。在附录[其他资源和链接](#)里就有一些介绍了内核proc系统的短小精干的文章。如果你在本文中找不到想要的资料, 就可以到附录[其他资源和链接](#)去看看, 你会有所收获的。

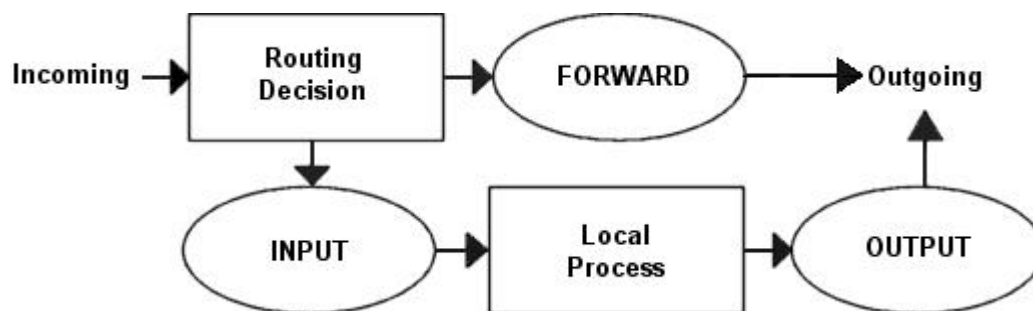


在本文所用的脚本中还包含了一个名为Non-Required proc configuration (非必需的proc设置) 的小节。当有什么工作不象你所想象的那么正常时, 可以来这儿看看, 它能提供给你最基本的一些信息, 但在你真正弄明白它们的含义之前不要进行改动。

7.2.4. 规则位置的优化

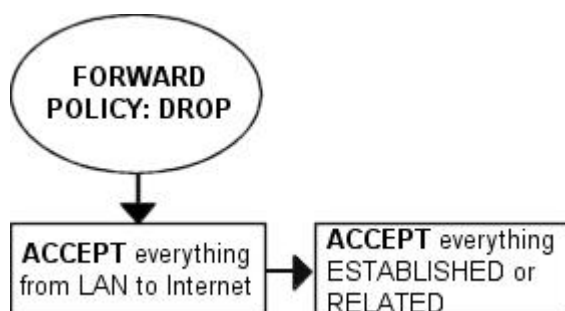
本节简要地描述了针对脚本`rc.firewall.txt`，我将如何选择、使用内建的链链和自定义的链。我选择过的一些路径从这个或那个角度看可能是错误的，我会指出这些情况和问题发生在何时何地。这里还对章节 [表和链](#) 做了简要的回顾，希望能给你一点儿提醒，以使你能想起在实际应用中包是如何表和链的。

为了尽可能地少占用CPU，我们已经替换了所有不同的自定义链，与此同时，我把主要的精力放在了安全性和易读性上。我不让TCP包去经历ICMP、UDP和TCP规则的洗礼，而是简单地匹配所有的TCP包，然后让它去一个自定义链中旅行。这种方法并不比让它经历所有的规则开销大。下图可以解释在Netfilter中，外来的包是如何被处理的（相对于章节 [表和链](#) 的深入讨论，这个图形太粗糙了）。我希望通过上面的解释和下面的图形能让大家明白写这个脚本的目的，详细的注释在后面几节。

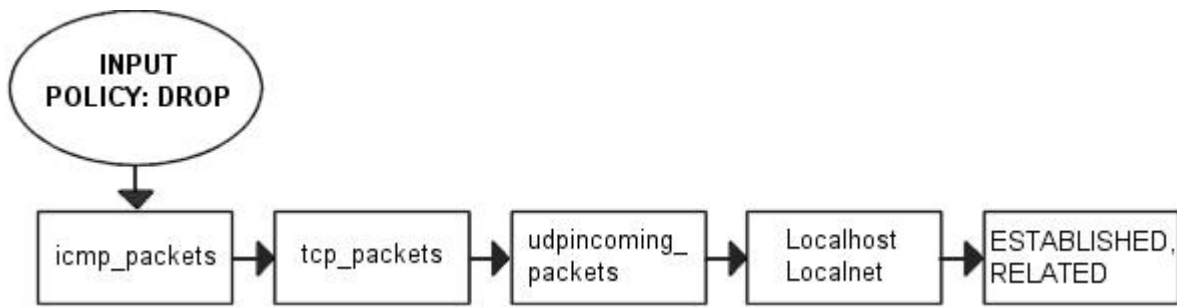


利用这个图形，我们可以弄清楚脚本的目的。整个脚本基于这样一种假设，我们有一个局域网，一个防火墙及一个Internet连接，且有一个静态IP地址（相对的是动态地址，它们使用的连接是DHCP、PPP、SLIP，等等），还想把防火墙作为Internet上的一台服务器来运行某些服务。我们完全信任局域网，因此不能阻塞任何从局域网发出的数据传输。还有一个要优先考虑的事，我们只允许那些被明确说明为可以接受的数据通过。为了做到这一点，我们就要把缺省策略设为DROP。这样，那些没有被明确标识为允许进入的数据就都被阻塞了。

在上面的假设里，我们想让局域网能够访问Internet。因为局域网是被完全信任的，所以我们应该允许所有来自局域网的数据通过。但Internet是不被信任的，所以我们想阻塞从Internet向我们的局域网发起的连接。根据上面的所有假设，我们来考虑考虑需要做什么、不需要做什么以及我们想做什么。



首先，我们解决的是局域网要能连接到Internet的问题。那我们就要对所有数据包做NAT操作，因为局域网内的机器都没有真实的IP地址。NAT是在PREROUTING链中完成的，这也是脚本最后创建的那个规则所在的链。这意味着我们必须要在FORWARD链中做过滤工作，否则我们就是允许所有外部的机器都能完全访问局域网了。因为我们完全信任局域网，所以允许所有由内向外的数据通过。由于我们假设Internet上的机器都不可以访问局域网内的机器，所以要阻塞所有由外向内的连接，但已经建立的或相关的连接除外，因为它们只是用来回应内网对外网的访问，而不是建立对内网的新连接。

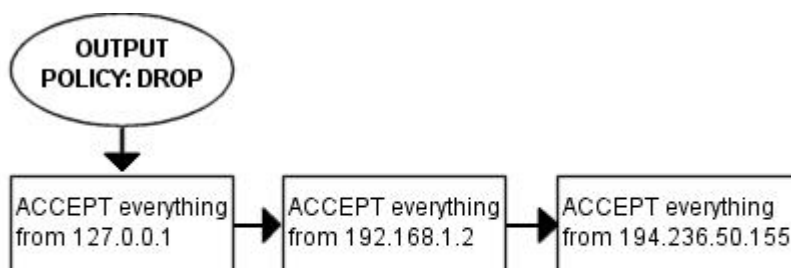


由于资金有限，我们的防火墙只提供了有限的几个服务：*HTTP*、*FTP*、*SSH*和*IDENTD*。因此，我们要在 *INPUT* 链里允许这些协议通过，还要在 *OUTPUT* 链里允许返回的数据通过。我们除了完全信任局域网，也信任loopback和它的IP地址，因此我们要有相应的规则来允许所有来自局域网和loopback的数据通过。但是我们不会允许一些特殊的包或包头通过，也不会接受Internet上某一段IP的访问。比如，网段 *10.0.0.0/8* 是为局域网而保留的，一般来说，我们不允许来自它们的包进入，因为这样的包90%都是用来进行欺骗的。不过，在实现这条标准之前，还要注意一个问题，就是有一些ISP在他们的网络里使用的恰恰就是这些地址。在附录 [常见问题与解答](#) 里有这个问题的进一步说明。

因为我们在防火墙上运行FTP服务，而且想让包经历最少的规则，所以要把处理established和related状态的规则放到 *INPUT* 链的顶部。基于同样的原因，我们把这些规则分到子链中。这样，包就可以尽量少地穿越规则，从而节省时间，也可以降低网络的冗余。

在这个脚本里，我们依据不同的协议（如*TCP*、*UDP*或*ICMP*）把包分到子链中。用来匹配 *TCP* 包的链叫做 *tcp_packets*，它可以匹配所有我们允许通过的*TCP*端口和子协议（如FTP、HTTP等）。我们还要建立一个名为 *allowed* 的子链，以便在真正接受“那些使用有效端口来访问防火墙的*TCP*包”之前，对它们进行附加的检查。至于*ICMP*包，自有称作 *icmp_packets* 的链来处理。在决定如何建立这个链时，我考虑到如果我们同意接受*ICMP*包的类型和代码，就没有必要对它们做附加的检查，所以直接接受它们就行了。最后，UDP包由谁处理呢？当然就是 *udp_packets* 了。如果包是那种允许被接收的类型，就直接放行了。

因为我们的网络很小，所以防火墙也要作为工作站来用。这就要求我们要允许一些特殊的协议能和它通信，比如 *speak freely* 和 *ICQ*。



现在，我们来考虑考虑OUTPUT链。因为很信任防火墙，所以我们允许几乎所有离开它的包通过，而没有阻塞任何用户和协议。但我们也不想让人利用这台机子进行IP欺骗，因此我们只放行那些从防火墙本身的IP发出的包。为了实现这一点，我们很可能在ACCEPT链中加入这样一条规则：如果包是由防火墙的IP发出的，就放行，否则，它们就会被OUTPUT链的缺省策略DROP掉。

7.2.5. 缺省策略的设置

在开始写其他规则之前，我们先要用下面的语句建立缺省的策略：

```
iptables [-P {chain} {policy}]
```

每一条链的策略都是用来处理那些在相应的链里没被规则匹配的包。也就是说，如果有一个包没有被规则集中的任何规则匹配，那策略就有用武之地了。



要谨慎地设置其他表里的链的策略，因为它们不是用来过滤包的，这就可能引起很怪异的行为发生。

7.2.6. 自定义链的设置

现在，你对我们的防火墙应该已经有了一个很清晰的印象，心动了吧。心动不如行动，让我们把它变为现实吧。这一节我们就要小心仔细地创建所有自定义链和链内的规则。

如前所述，我们要建立这几条自定义链：*icmp_packets*、*tcp_packets*、*udp_packets* 和 *allowed*，其中*allowed*链是由*tcp_packets*链调用的。所有进入`$INET_IFACE`的ICMP包都会被重定向到*icmp_packets*链，TCP包是到 *tcp_packets*链，那UDP包自然就是*udp_packets*链了，详细的解释都在 [INPUT chain](#)里。创建自定义链的命令还记得吗？很简单哦，只要使用选项-N，再指定链的名字即可（不要忘了，新建的链都是空的），如下：

```
iptables [-N chain]
```

在下面的几节里，我们会详尽地介绍上面创建的每一条链，以使你了解它们包含哪些规则、有什么作用。

7.2.6.1. bad_tcp_packets链

这条链包含的规则检查进入包（incoming packet）的包头是否不正常或有没有其他问题，并进行相应地处理。但事实上，我们使用它只是为了过滤掉一些特殊的包：没有设置SYN位但又是NEW状态的TCP包，还有那些设置了SYN/ACK但也被认为是NEW状态的TCP包。这条链可以用来检查所有可能的不一致的东西，比如上面的包或者*XMAS* port-scans等。我们还可以为INVALID状态的包增加一条规则的。

如果你想完全了解无SYN位的NEW状态（NEW not SYN），可以去附录[常见问题与解答](#)里看看[未设置SYN的NEW状态包](#)一节，它介绍了未设置SYN的NEW状态包通过其他规则的情况。在某些情况下可以允许这种包通过，但99%的情况是我们不想让它们通过。因此，我们会先记录这种包，然后再扔掉它们。

我们拒绝SYN/ACK包以NEW状态进入的原因也是非常简单的，深入的说明在附录[常见问题与解答](#)的[NEW状态的SYN/ACK包](#)里。基本上，我们这样做是出于对其他主机的好意，因为我们为他们预防了序列号预测攻击（sequence number prediction）。

7.2.6.2. allowed链

如果包是从`$INET_IFACE`进入的, 而且是TCP包, 那它就要经过 `tcp_packets` 链的检验。如果这个连接就是冲着被允许通过的端口来的, 我们还要对它进行一些检查, 以确定是否真的要接受它。这些“最后的审判”都是在`allowed`链里进行的。

首先, 我们看看这个包是否是SYN包, 如果是, 它很可能是新连接的第一个包, 我们当然接受了。如果不是, 那就看看包是否来自某个`ESTABLISHED`或`RELATED`状态的连接, 是的话, 就接受。`ESTABLISHED`状态的连接是那种在两个方向上都有流量存在的连接。依据状态机制的观点, 这个连接一定处于是`ESTABLISHED`状态的, 因为我们现在能看到这个包, 说明以前肯定收到了相应的SYN包。最后一条规则将`DROP`所有其他的包。当包到达最后这条规则, 就几乎意味着所有连接都不会有双向的交流, 也就是说, 我们不会回应 `SYN`包。当试图用非`SYN`包开始新的连接时, 包也会走到这条规则。不用`SYN`包建立新连接没有什么实际的用处, 当然, 端口扫描要排除在外。就我知道的而言, 现在没有什么有用的 `TCP/ IP`程序会使用`SYN`包以外的包来打开一个 `TCP`连接。因此, 我们要把这样的包`DROP`掉, 我有99%的把握说它们是端口扫描用的。

7.2.6.3. 处理TCP的链

`tcp_packets`链指定了哪些端口可从Internet访问。但我们还要对进入的包做更多的检查, 因此, 每个包都会被发送到上面提到的`allowed`链。

`-A tcp_packets`告诉iptables要在哪条链里增加规则, 规则被放在指定链的末尾。`-p TCP`指定要匹配的是 `TCP`包, `-s 0/0`说明要匹配的源地址是从网络掩码为0.0.0.0的地址0.0.0.0开始的, 换句话说, 就是所有的地址。这实际上是默认值, 我写出来只是尽可能使你更明白。`--dport 21`指定目的端口, 也就是说如果包是发往端口21的, 就会被匹配。如果所有的标准都匹配了, 包就要被送往`allowed`链。

`TCP`的21号端口也是允许访问的, 也就是`FTP`的控制端口, 它可以控制`FTP`连接, 前面提到过, 我还允许所有`RELATED`状态的连接通过。这样, 我们也就可以使用`PASSIVE`（主动）和`ACTIVE`（被动）的连接了, 当然, 要事先装载`ip_conntrack_ftp`模块。如果我们不想再提供 `FTP`服务, 就卸载`ip_conntrack_ftp`模块, 并把`$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed` 这一行从文件 `rc.firewall.txt`里删掉。

22号端口是`SSH`使用的。如果你允许外网的任何人都能通过`telnet`（使用23号端口）访问你的机器, 那你还是使用`SSH`吧, 它的安全性要好很多。注意, 你操作的是防火墙, 把任何访问权分配给除你自己之外的人都不是什么好主意。防火墙总是应该尽量少地暴露自己。

80是`HTTP`端口, 也就是说你在防火墙上运行了网页服务。如果你不提供网页服务, 就删掉这条规则吧。

最后我们还提供了`IDENTD`服务, 端口是113。这个服务对某些协议是必须的, 如`IRC`。注意, 如果你`NAT`一些在内网里的主机的话, 软件`oidentd`也值得一用, 它会把 `IDENTD`请求中继给内网里正确的机器。

如果没有匹配上面任何一条规则, 包就会被送回`tcp_packets`链的父链, 也就是把它发到`tcp_packets`链的那条规则所在的链。如果你想打开更多的端口, 只要对`tcp_packets`链里的任何一行使用“复制、粘贴大法”, 再修改一下端口号即可。

7.2.6.4. 处理UDP的链

如果我们在 *INPUT* 链中遇到了 *UDP* 包, 就把它发送到 *udp_packets* 链。在那里, 我们只处理 *UDP* 包, 所以要用 `-p UDP` 来指定相应的协议。我们接受来自任何地址的包, 故有 `-s 0/0`, 这其实就是源地址选项的默认值, 但为了更明确, 我们还是把它写出来了。此外, 我们只接受发往特定端口的包, 这些端口是我们想对 Internet 开放的。注意, 我们不需要依据发送端的源端口来决定是否打开某个端口, 这个工作是由状态机制完成的。如果我们想运行某个使用 *UDP* 端口的服务 (如 *DNS*), 只要开放相应的端口, 其他的端口不需要打开。那些处于 *ESTABLISHED* 状态、正在进入防火墙的包在到达包含 `--state ESTABLISHED, RELATED` 的规则 (这是 *INPUT* 链里那些“处理来自 Internet 的包的规则”中的第一条规则) 之后就会被接受了。

我们不接受外来的以 53 号端口为目的的 *UDP* 包, 也就是说, 我们不想接受外来的 *DNS* 查询。其实, 规则已经写好了, 我只是把它给注释掉了。如果你想把防火墙作为一台允许 Internet 访问的 *DNS* 服务器, 那就把注释符号去掉。

就我个人而言, 我会打开 123 号端口, 它对应的协议是 *network time protocol*, 简称 *NTP*。我们可以利用这个协议与某台具有精确时间的时间服务器联系, 以设置本机的时间。你们中的大部分可能用不到此协议, 所以我也把它注释掉了, 虽然我已经写出了规则。

我打开了 2074 号端口, 它是某些实时的多媒体应用程序使用的。比如 *speak freely*, 你可以用这个程序通过音箱、麦克风或耳麦与其他人进行实时交谈。如果你不需要, 就把这条规则注释掉吧。

端口 4000 相应的协议是 *ICQ* 协议, 由 *ICQ* 使用, 世界上使用最广泛的聊天程序之一, “地球人都知道”。Linux 上至少有 2-3 种不同的 *ICQ* 克隆。我想不必解释为什么要开放这个端口了吧。(译者注: 国产的聊天程序, 常见的是 *QQ* (端口 8000), 现在又有了 *UC* (端口 3001) 等。)

如果你正在经历日志满天飞的苦恼, 这里有两个额外的规则可以使用, 当然, 这要看是因为什么引起的了。我们这里的第一条规则会阻塞目的端口是 135 到 139 的广播包。大部分 Microsoft 使用者会用到 *NetBIOS* 或 *SMB*, 而它们就使用这些广播包。这条规则可以阻塞所有位于外网的那些 Microsoft Network 产生的广播包, 我们的日志也就可以简洁一些了。第二条规则也是解决日志问题, 不过问题的产生者变了, 这回是外网的 *DHCP* 查询。如果你的外网是由非交换的以太网组成的, 在那里客户机可以通过 *DHCP* 得到 IP 地址, 也就是说, 如果外网里会有很多 *DHCP* 查询广播包, 那就启用这条规则吧。



注意, 我把最后这两条规则也注释掉了, 因为有些人可能想看看相关的记录。如果你正经历“合法日志过多”的痛苦, 就试试丢弃那些包吧。其实, 在 *INPUT* 链中, 就在这两条有关日志的规则之前, 还有更多这种类型的规则。

7.2.6.5. 处理ICMP的链

现在, 我们该决定可以接受哪些 *ICMP* 类型了。在 *INPUT* 链中, 如果 *ICMP* 包是从 *eth0* (即本例的 Internet 接口) 进入的, 我们就要把它重定向到 *icmp_packets* 链 (前面提到过), 以检查是否是可以接受的类型。目前, 我只接受三种 *ICMP* 包: *ICMP Echo requests*, *TTL equals 0 during transit* 和 *TTL equals 0 during reassembly*。默认不接受其他任何 *ICMP* 类型的原因是, 几乎所有其他类型的 *ICMP* 包都是 *RELATED* 状态的, 也就是说它们都会被处理 *RELATED* 状态的规则放行。

如果一个 *ICMP* 包是用来回应“已经存在的包或流”的, 那它就是与那些流相关的, 也就是说, 它的状态是 *RELATED*。更多的信息在章节 [状态机制](#) 里。



现在来解释一下我为什么只接受上面提到的三种ICMP包。Echo Request用来请求echo reply, 这个操作主要被用来ping其他的机器, 以确定那些机器是否可用。如果没有这一条规则, 其他机器将不能通过ping来确定我们是否可用。注意, 有些人倾向于删掉此规则, 只是因为他们不想被Internet看到。删掉这个规则将会使任何来自Internet的、对我们防火墙的ping都无效, 因为防火墙对他们完全没有回应。

允许超时 (Time Exceeded, 如 *TTL equals 0 during transit* 传输期间生存时间为0和 *TTL equals 0 during reassembly* 在数据报组装期间生存时间为0) 信息进入, 我们就可以追踪从本地到某台主机的路径, 或者在包的TTL为0时, 我们能得到回应信息。比如, 在我们追踪到某台主机的路径时, 会以 *TTL = 1* 的包开始。当它得到第一个路由时, TTL减为0, 我们也会得到第一个路由返回的超时信息。然后是 *TTL = 2* 的包, 我们就会得到第二个路由器返回的超时信息。如此, 直到得到我们的目的主机返回的信息。这样, 我们就可以从路径上的每一台主机得到一个回应, 从而我们可以看到路径上的每一台主机, 也就可以知道路径是断在哪台机器了。

完整的 *ICMP* 类型列表在附录 [ICMP类型](#) 里。关于ICMP类型的更多信息与用法, 我建议你看看下面的文章:

- Ralph Walden的 [The Internet Control Message Protocol](#)。
- J. Postel的 [RFC 792 - Internet Control Message Protocol](#)。



注意, 我阻塞了所有我不想接受的 *ICMP* 包, 这对你的网络来说可能会有问题, 但在我这里, 一切工作正常。

7.2.7. INPUT链

我写的 *INPUT* 链大部分是使用其他链来完成这个艰难的工作的。这样做, 我们就不需要从 iptables 装载太多的规则 (译者注: 这是针对装载INPUT链的规则说的, 因为这时其他规则已经装载好了), 而且它在较慢的机器上也可以工作得很好, 但另一方面, 这样的机器在高负载时还是会丢弃很多包 (译者注: 机器慢, 就是不行)。之所以能做到这一点, 是因为对于大量不同的包, 我们通过检查特定的细节来确定它们的类别, 再把这些包发送到相应的自定义链去处理。这样, 我们可以分割规则集使之包含很少的规则, 每个包要经历的规则也就少了。从而, 在过滤包时, 防火墙的开销也就小多了。

首先, 我们要检查进入的tcp包的形态是否是不正常的或我们不想要的。这个工作是这样完成的, 我们把所有的tcp包送到 *bad_tcp_packets* 链, 由其中的规则进行检查, 具体的描述在小节 [bad_tcp_packets链](#) 里。

然后, 我们开始处理被信任的网络的数据传输。这包括所有来自“连接内网的网卡”的流量, 所有来自和发往 *loopback* 的流量 (要注意, 和 *loopback* 相对应的IP地址包括了所有分配给防火墙的地址, 其中也包括连接Internet的地址)。我们把处理 *LAN* 的流量的规则放在防火墙的上部, 因为我们的局域网产生的流量要远远多于Internet连接。这样, 规则会更有效率, 防火墙就能以较小的开销去匹配包, 从而网络阻塞的可能性也就减小了, 而且也便于我们查看经过防火墙的包主要是什么类型。

下面的一些规则会处理来自Internet的信息，在接触这些规则之前，有一个相关的规则，我们可用它来减少一些开销。这是一个处理状态的规则，它允许所有处于状态`ESTABLISHED`或`RELATED`且发往 Internet接口的包进入。在`allowed`链中有一个与此类似的规则（译者注：实在是多余，建议大家把它拿掉吧）。顺序上，当然是`INPUT`链里的规则先处理包了。然而，在`allowed`链里保留一个`state ESTABLISHED, RELATED`规则还是有一些原因的，比如，方便某些人想剪切此功能，粘贴到其他地方。

在`INPUT`链里，我们会把从`$INET_IFACE`进入的所有`TCP`包发往`tcp_packets`链，类似地，把`UDP`包发往`udp_packets`链，把`ICMP`包发往`icmp_packets`链。一般说来，防火墙遇到的最多的包是`TCP`包，其次是`UDP`包，最后是`ICMP`包。但要注意，这只是一般情况，对你可能不适用。一样的规则因为顺序不同，或者说逻辑不同，效率会有很大的差别。如果规则集写得不好，即使只有100条规则，而且有100mbit的网卡，就算是`Pentium III`的机子也会吃不消的。所以你自己写规则集时一定要注意这一点。

这里有一条被注释掉了规则，万一在我们的防火墙外部有一些Microsoft网络，我们可以启用它来解除日志过多的烦恼。Microsoft的客户机有个坏习惯，就是向地址224.0.0.0/8发送大量的多播包。因此我们要有条规则来阻塞那些包，以免我们的日志被它们填满。还记得吗？`udp_packets`链里也有两条类似的规则。忘了的话，就到[处理UDP的链](#)看看吧。

在其他的包被`INPUT`链的策略处理之前，我们会把它们记录下来，以便查找可能的问题或bug：它可能就是我们不想允许它进入的那种包，也可能是对我们做了什么坏事的用户，还可能是防火墙的问题，如我们阻塞了应该被放行的包。我们要了解所有的情况，这样问题才能得以解决。我们每分钟最多记录3个包，因为我们可不想让日志里记载的都是废话。为了容易辨别包的来源，我们还对所有的记录设置了前缀。

所有没被上面的规则处理的包都会被策略`DROP`掉。策略的设置在本章的小节[缺省策略的设置](#)里，距离我们已经很远喽。

7.2.8. FORWARD链

在本例中，`FORWARD`链包含的规则很少。首先，我们会把所有的包发往`bad_tcp_packets`链。此链我们前面提到过多次，它可以被多条链调用，其实它也就是为这个目的而设计的。

之后就是`FORWARD`链的主要规则了。第一个允许所有来自`$LAN_IFACE`的数据通过，没有任何限制，也就是说，我们的`LAN`可自由地访问Internet。第二个允许`ESTABLISHED`和`RELATED`状态的包能通过防火墙。换句话说，就是所有对我们的内网发出的连接的回应都可以返回局域网。为了使我们的内网能访问Internet，这些规则是必须的，因为我们在前面已经把`FORWARD`链的策略设为`DROP`了。这样设置规则也是很聪明的，因为它在保证局域网可以访问Internet的同时阻止了Internet对局域网的访问。

最后我们也有一个处理日志的规则，用来记录没被上面任何规则匹配的包。这样的包很可能是形态不正常的或者是其他问题，比如可能是黑客攻击。这个规则与`INPUT`链中的那个类似，只是前缀不同，这里用的是：“`IPT FORWARD packet died:`”。前缀主要用来分离日志的记录，便于我们查找包的来源和包头的一些信息。

7.2.9. OUTPUT链

除了我几乎没有人把防火墙还当作工作站来使用，但正因为这样，我允许几乎所有从防火墙

的IP（包括 `LOCALHOST_IP`, `$LAN_IP`或`$STATIC_IP` ）出发的数据，而阻塞其他情况。因为其他任何情况都可能被人以某种方式欺骗。最后的规则还是用来记录那些要被策略DROP掉的包。这样，我们就可以了解它们，继而可以对产生的问题（可能是具有威胁性的错误，或者是用来进行欺骗的包）采取行动。

7.2.10. PREROUTING链

顾名思义，*PREROUTING*链（*nat*表的）是在路由之前做网络地址转换工作的。然后，包再经过路由，就会被送到*filter*表的*INPUT*或*FORWARD*链。我们在这里讨论这个链的唯一原因是，我们觉得有责任再次指出你不应该在此链中做任何过滤。*PREROUTING*链只会匹配流的第一个包，也就是说，这个流的所有其他的包都不会被此链检查。事实上，在这个脚本中，我们根本没有用到*PREROUTING*链。如果你想对一些包做*DNAT*操作，例如，你把*web server*放在了局域网内，这里就是你放置规则的地方。有关*PREROUTING*链的详细信息在章节[表和链](#)中。



千万注意，*PREROUTING*链只能做网络地址转换，不能被用来做任何过滤，因为每个流只有第一个包才会经过此链。

7.2.11. POSTROUTING链

我们最后的任务应该是构造网络地址转换，对吧？至少对我来说是的。我们在*nat*表的*POSTROUTING*里只加入了一条规则，它会对所有从Internet接口（对我来说，这是*eth0*）发出的包进行*NAT*操作。在所有的例子脚本里，都有一些变量，它们要给以正确的配置。选项*-t*指定要在那个表里插入规则，这里是*nat*表。命令*-A*说明我们要把规则添加到 *POSTROUTING*链末尾。*-o \$INET_IFACE*指定要匹配所有从接口*INET_IFACE*出去的包，这里我们使用的是*eth0*。最后，我们把*target*设置为*SNAT*。这样，所有匹配此规则的包都会由*SNAT target*处理，之后，它们的源地址就是Internet接口的地址了。不要忘了*SNAT*可是一定要有IP地址的，用*-to-source* 来设置哦。

在这个脚本中，我们选择*SNAT*而不用*MASQUERADE*是有原因的。主要的原因是我们的防火墙有静态IP地址，使用*SNAT*会更快更有效。还有一个原因是我们要在这个例子中展示它的作用以及怎样使用它。如果你没有静态的IP地址，要想实现*SNAT*，还是使用*MASQUERADE*为好，因为它简单易用，而且它可以自动获得IP地址。当然，计算机的消耗会多一点，但如果你使用*DHCP*，这样做是很值得的。如果你想了解*MASQUERADE target*的表现，应该看看脚本[rc.DHCP.firewall.txt](#)。

Chapter 8. 例子简介

本章的目的是对指南提到的每个脚本都给以简单明了的说明，以及提供一个关于这些脚本的框架，描述它们提供的服务。这些脚本不是任何情况都能用的，它们可能并不符合你的意图。也就是说，为了能满足你的需要，还是要取决于你自己。在这方面，下面的内容可能会给你很大的帮助。第一小节介绍了这些脚本的结构，你会发现我们在这些脚本里使用的处理方式还是比较容易的。

8.1. rc.firewall.txt脚本的结构

本指南所有的脚本都是依据一个特定的结构来写的。理由嘛，就是这样可以使它们彼此相似，便于我们查找不同之处。本章将要对这个结构做一个很好的说明，而且还会简单地阐述这些脚本为什么会按照现在这种样子来写，以及我为什么选择一直使用这种结构。



注意，即使我选择了这种结构，你也不一定非要用它，对你来说，它可能并不是最好的。我选择它只是因为它易读，而且能很好地符合我的逻辑。

8.1.1. 脚本结构

这就是本指南所有脚本遵循的脚本结构。如果有不同于此的地方，可能就是我在出错了，除非我特意说明为什么要打破这种结构。

1. *Configuration* —— 首先是一个配置选项区，里面的变量在脚本中会用到。几乎任何脚本（shell-script）的第一部分都是配置选项区。
 1. *Internet* —— 有关Internet连接的配置。如果我们没有任何Internet连接，这一部分就可以跳过去。注意，相比我们列出来的，这一部分可能会包含更多小节，虽然我们这里只有了几个，但足够应对我们已有的各种Internet连接了。
 1. *DHCP* —— 如果脚本用到了DHCP，我们就要在此添加相应的配置。
 2. *PPPoE* —— 如果想把脚本用于PPPoE连接，就要在此添加相应的配置。
 2. *LAN* —— 如果防火墙后有局域网，就要使用这里的配置了。大部分情况下都会用到这儿，因为局域网几乎总是存在的。
 3. *DMZ* —— 对非军事区（DMZ zone）的配置。大部分脚本不会用到这个设置，因为这些脚本针对的主要是一些普通的家庭网络，或小企业的网络。
 4. *Localhost* —— 本地（local-host）的有关设置。虽然我把它写成变量的形式，但一般不会被改变，也不应该有什么理由要改变它们。
 5. *iptables* —— 有关iptables的设置。大部分情况下，这里只设置一个变量，用来指向iptables程序的位置。
 6. *Other* —— 如果还有什么信息，首先应该把它们放在相应的小节里，实在没有相应的小节，就放这儿吧。
2. *Module loading* —— 脚本应该维护一个模块列表。它分为两部分，第一部分包含必需的模块，同时第二部分要包含不必要的模块的列表。



注意，这些模块可能会提高安全性，或为管理者、客户添加某些服务，还有一些模块不是必需的，但它们可能也被加入了列表。不过，在本例中，我已经注意了这个问题。

1. *Required modules* —— 这里装载的是必要的模块，它们可能会提高安全性或为管理者、客户增加某些服务。

2. *Non-required modules* —— 这里列出的是不必要的模块, 所以它们都被注释掉了。如果你用到了它们提供的功能, 就可以启用它们。
3. *proc configuration* —— 这儿关心的是有关proc系统的设置。如果一些选项是必须的, 我们就启用它, 如果不是, 就把它注释掉。大部分有用的proc配置都列在这儿了, 但远远不是全部的。
 1. *Required proc configuration* —— 包含了使脚本能正常工作的所有必需的proc配置, 它也可以包含一些能提高安全性或为管理者、客户增加特定服务的选项。
 2. *Non-required proc configuration* —— 这里提到的选项不是必需的, 虽然它们可能很有用。因此, 我把它们都注释掉了。当然, 这里并没有包括所有这样的选项。
4. *rules set up* —— 现在, 应该添加规则了。我把所有的规则都明确地分配到了与表、链相应的小节里。所有自定义的规则都写在系统内建的链之前 (译者注: 当然要写在前面的了, 因为后面要调用它们哦)。另外, 我是按照命令 `iptables -L` 输出的顺序来安排此脚本里表与链的出现顺序的 (译者注: 这样, 便于我们查看哦)。

1. *Filter table* —— 首先是filter表, 而且我们先要设置策略。

1. *Set policies* —— 为所有系统内建的链设置策略。通常, 我会设置DROP, 对于允许使用的服务或流会在后面明确地给以ACCEPT。这样, 我们就可以方便地排除所有我们不想让人们使用的端口。
2. *Create user specified chains* —— 在这里创建所有以后会用到的自定义链。如果没有事先建立好, 后面是不能使用它们的, 所以我们要尽早地建立这些链。
3. *Create content in user specified chains* —— 建立自定义链里使用的规则。其实你也可以在后面的某个地方写这些规则, 之所以写在这儿, 唯一的原因是这样做规则和链会离得近些, 便于我们查看。
4. *INPUT chain* —— 创建INPUT链的规则。



从这里开始, 我就是遵循 `iptables -L` 的输出格式来创建规则的, 这的唯一原因就是为了便于阅读, 避免混淆。

5. *FORWARD chain* —— 为FORWARD链创建规则。
6. *OUTPUT chain* —— 为OUTPUT链创建规则。其实, 在这里要建的规则很少。
2. *nat table* —— 在处理完filter表之后, 该设置nat表了。我们这样做是有一定原因的。首先, 我们不想太早地打开转发机制 (译者注: 注意, filter表设定的是过滤机制, 而不是转发) 和NAT功能, 因为它们可能导致数据会在错误的时间 (也就是这样的时刻: 我们打开了NAT, 但过滤规则还没有运行) 通过防火墙。还有, 我把nat表看作是围绕filter表的一个层。也就是说, filter表是核心, nat表是它外部的一个层, mangle表是第二层。从某些观点来看, 这可能有点不对, 但也八九不离十了。
 1. *Set policies* —— 与filter一样, 我们先来设置策略。一般说来, 缺省的策略, 即ACCEPT, 就很好。这个表不应该被用来做任何过滤, 而且我们也不

应该在这儿丢弃任何包, 因为对我们假设的网络情况来说, 可能会发生一些难以应付的事情。我把策略设为ACCEPT, 因为没有什么原因不这样做。

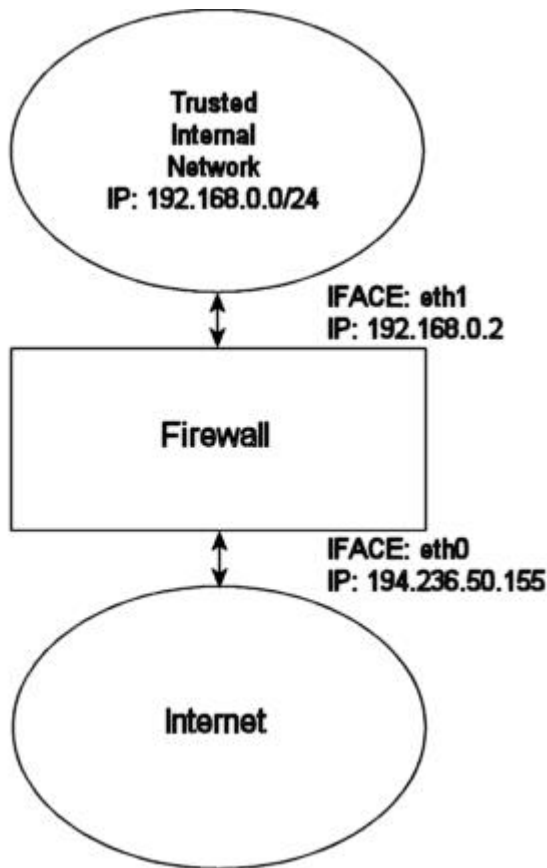
2. *Create user specified chains* —— 在这儿创建nat表会用到的自定义链。一般情况下, 我没有任何规则要在这儿建立, 但我还是保留了这个小节, 以防万一罢了。注意, 在被系统内建链调用之前一定要建好相应的自定义链。
 3. *Create content in user specified chains* —— 建立自定义链的规则。
 4. *PREROUTING chain* —— 要对包做DNAT操作的话, 就要用到此链了。大部分脚本不会用到这条链, 或者是把里面的规则注释掉了, 因为我们不想在不了解它的情况下就在防火墙上撕开一个大口子, 这会对我们的局域网造成威胁。当然, 也有一些脚本默认使用了这条链, 因为那些脚本的目的就是提供这样的服务。
 5. *POSTROUTING chain* —— 如果使用SNAT操作, 就要在此建立规则。你可能有一个或多个局域网需要防火墙的保护, 而我就是依据这样的情况来写此脚本的, 所以这个脚本中使用的 POSTROUTING链是相当实用的。大部分情况下, 我们会使用SNAT target, 但有些情况, 如PPPoE, 我们不得不使用MASQUERADE target。
 6. *OUTPUT chain* —— 不管什么脚本都几乎不会用到这个链。迄今为止, 我还没有任何好的理由使用它, 如果你有什么理由用它了的话, 麻烦你把相应的规则也给我一份, 我会把它加到本指南里的。
3. *mangle table* —— 最后要做的就是处理mangle表了。通常, 我不会使用这个表, 因为一般情况下, 它不会被任何人要到, 除非他们有什么特殊的需要, 比如为了隐藏一条连接后的多台机子, 我们要统一设置TTL或TOS等。在这个脚本里, 此表是空白的。但在此指南中还是有个小小的例子说明了mangle表的用处。
1. *Set policies* —— 设置策略。这里的情形和nat表几乎完全相同。这里不应该做过滤, 也不应该丢弃任何包。在任何脚本里我都不会把mangle表的策略设为其他的值, 也不鼓励你这样做。
 2. *Create user specified chains* —— 建立自定义链。我几乎不会用到这个链, 所以没有建立任何规则。保留此小节, 只是已备后用。
 3. *Create content in user specified chains* —— 建立自定义链的规则。
 4. *PREROUTING* —— 本指南的所有脚本都未在此链建立规则。
 5. *INPUT chain* —— 本指南的所有脚本都未在此链建立规则。
 6. *FORWARD chain* —— 本指南的所有脚本都未在此链建立规则。
 7. *OUTPUT chain* —— 本指南的所有脚本都未在此链建立规则。
 8. *POSTROUTING chain* —— 本指南的所有脚本都未在此链建立规则。

这应该可以较详细地解释每个脚本的结构是怎样的以及它们为什么要使用这个结构了。



注意, 上面的描述其实还是非常简单的, 应该被看作一个摘要, 它简要地解释了脚本为什么要按照这种松散的结构来写。千万注意, 我可没有说过这种结构是唯一的、最好的。

8.2. rc.firewall.txt



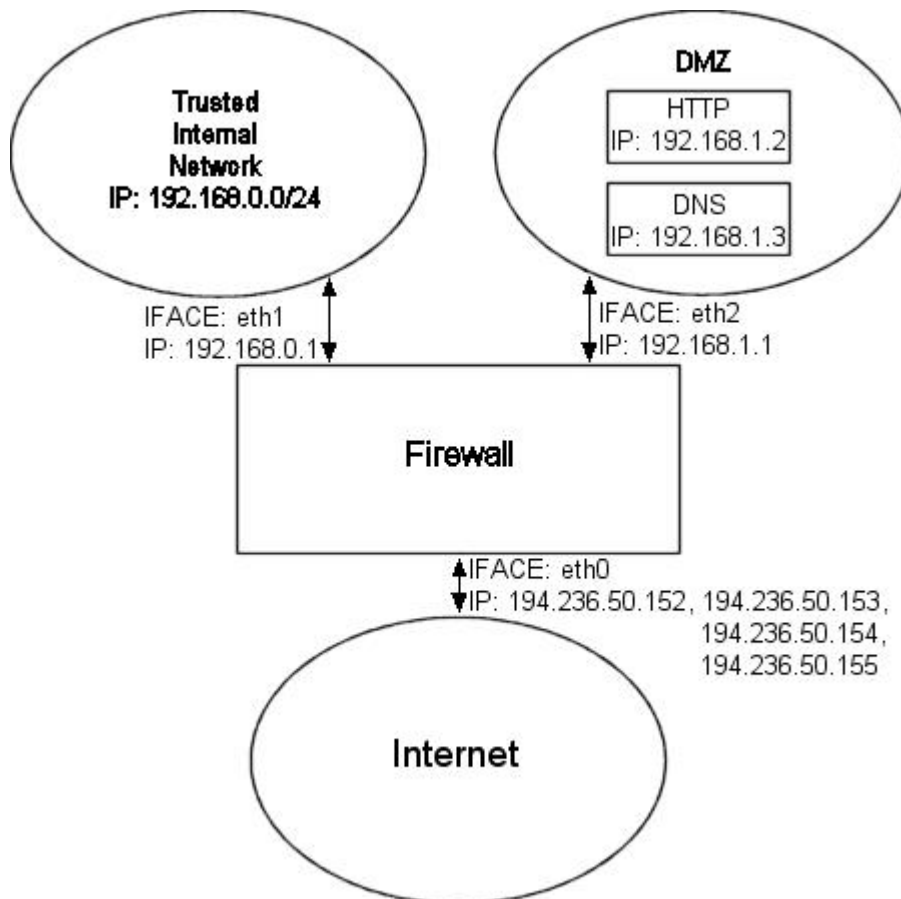
脚本[rc.firewall.txt](#)是核心, 第七章[防火墙配置实例 rc.firewall](#)对它已经做了很详细的解释, 其他的脚本都是以它为基础得到的。这个脚本主要是针对具有两个连接的家庭网络而设计的, 如一个局域网连接, 一个Internet连接。我们假设的情况是你有一个静态IP地址, 不需要DHCP, PPP, SLIP或其他什么协议为你动态分配IP。如果你想要的恰恰是使用这些协议的脚本, 就到[rc.DHCP.firewall.txt](#)看看吧。

脚本rc.firewall.txt要完全发挥作用, 系统必需要有下面列出的功能, 你可以把它们编译进内核, 也可以编译成模块。如果你改变了脚本, 就要加入相应的功能模块或把它们编进内核。

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER

- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

8.3. rc.DMZ.firewall.txt



脚本[rc.DMZ.firewall.txt](#)所针对的情况是这样的：有一个可信任的内网，一个DMZ，还有一个Internet连接。这里的DMZ是通过设置一对一的*NAT* 操作得到的，它需要IP别名（就是在同一块网卡上设置多个IP地址）的支持。我们还有别的方法来实现 DMZ：如果你有一个整个的网段，可划分子网，然后把某个子网分给DMZ，再为防火墙配置相应的内网与外网 IP地址（译者注：第一种方法是针对有多个网段的情况，即内网一个网段，DMZ一个网段，第二种方法是把一个网段划分成几个子网，这样就和第一种情况一样了）。注意，这种方法会多消耗两个IP，一个是网络地址，一个是广播地址（译者注：具体细节请上网搜索子网划分的相关信息，这个指南并不包含此类信息）。以上两种方法用哪一个就要你自己决定了。本指南会给你实现防火墙与NAT的手段或叫做技术，但具体如何去做，没有完全的说明，因为这已经超出本文的范围了。

这个脚本需要以下模块，也可能它们已被编译进内核了。

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONTRACK
- CONFIG_IP_NF_IPTABLES

- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

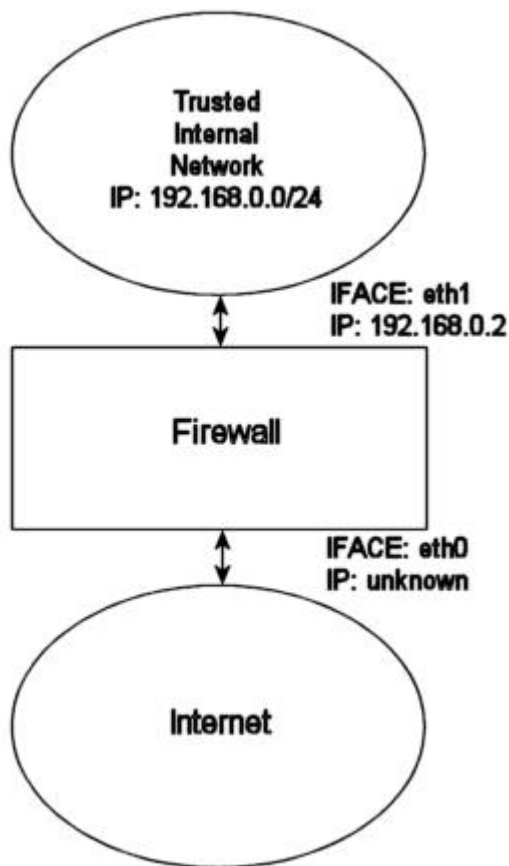
从图中可以看出, 此脚本假设你有两个内网, 一个是可信任的内网, 使用地址 192.168.0.0/24, 另一个是 DMZ (我们正是对它做一对一的NAT), 使用地址 192.168.1.0/24。如果有人从Internet向我们的DNS_IP发送一个包, 我们就要对它使用*DNAT*, 之后, 此包的目的地地址就指向DMZ 里的DNS服务器了, 它也就可以到达真正的DNS服务器。否则, DNS服务器不会看到这个包, 也就没有应答之说了。下面是实现上述*DNAT*功能的语句:

```
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d  
$DNS_IP \  
--dport 53 -j DNAT --to-destination $DMZ_DNS_IP
```

我们可以看出, 这个规则要放在*nat*表的*PREROUTING*链中, 包要满足的条件是: 使用 *TCP*协议且使用53号端口, 从接口*\$INET_IFACE*进入, 而且要以*\$DNS_IP*为目的。被匹配的包要交给*DNAT* target来处理, 它会把包的目的地地址改为由**--to-destination**指定的地址*\$DMZ_DNS_IP*。这就是 *DNAT*的工作流程。当相应的应答包被发送到防火墙时, 会自动地被un-DNAT。

现在, 你应该完全可以读懂这个脚本了。如果有什么你不明白的东西在脚本的其他部分没有被用到, 那可能就是我的错误了, 要告诉我哦。

8.4. rc.DHCP.firewall.txt



脚本[rc.DHCP.firewall.txt](#)适用于那些使用*DHCP*、*PPP*或*SLIP*连接Internet的情况，它和原始脚本[rc.firewall.txt](#)几乎一样，主要的区别在于这里不再使用变量**STATIC_IP**。原因很简单了，就是它不能和动态的IP一起使用。此脚本相对于原始脚本的改变是很少的，但还是有一些人发信问我做了什么改变。经过大家的考验，这个脚本应该是一个很好的解决方案了。

它需要如下功能模块。

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_MASQUERADE
- CONFIG_IP_NF_TARGET_LOG

我做的改变主要是删除了变量**STATIC_IP**以及和它相关的所有东西。以前，主要的过滤工作是基于变量**STATIC_IP**的，现在是**INET_IFACE**了。也就是说，在这个脚本里，我们不再把**-d \$STATIC_IP**作为过滤的条件，而是用 **-i \$INET_IFACE**。这几乎是唯一的改变，也是必需的改

变。

可还是有一些问题要考虑。现在, 我们不能再在 *INPUT* 链依据某些条件, 比如 `--in-interface $LAN_IFACE --dst $INET_IP` 来进行过滤 (译者注: 因为这时已无固定的 *INET_IP*)。这强迫我们只能基于 Internet 接口进行包的过滤, 在这种情况下, 内网必须访问那个可变的 Internet 的 IP。这会出现一些问题, 有个例子可以说明这一点, 就是我们在防火墙上运行 HTTP 服务。如果我们访问这个网站 (其中, 主页包含了一个指向 HTTP 服务器的静态连接, 这可能是某个动态的 DNS 解决方案), 问题就暴露了。经过 NAT 操作的机子会向 DNS 查询 HTTP 服务器的 IP, 然后再试着访问这个 IP。万一我们是基于接口和 IP 做的过滤, 这台机子就不能访问到 HTTP 了, 因为 *INPUT* 链会 DROP 掉这个包 (译者注: 还是因为 Internet 接口的 IP 不固定)。在某种情况下, 这也会发生在你有静态 IP 的时候, 但在那种情况下, 我们可以增加一条规则, 以检查 LAN 接口的包是否是发往 *INET_IP* 的, 若是, 则 **ACCEPT**。

如果你看过以前的内容, 得到或写一个可以获取动态 IP 的脚本可能会是个解决问题的好办法。比如, 我们可以写一个脚本, 它紧随着 Internet 连接的启动而运行, 而且它可以从命令 **ifconfig** 的输出中提取 IP, 再把这个 IP 赋给某个变量。较好的办法是使用一些程序自带的脚本, 如 **pppd** 带的脚本 `ip-up`。也有一些网站, 如 linuxguruz.org, 提供了很多有用的脚本, 你可以在附录 [其他资源和链接](#) 找到它的链接。



这个脚本比 `rc.firewall.txt` 的安全性要差一点。我明确地建议你尽可能使用后者, 因为前者的开放性大了点, 所以外部攻击的威胁就大了。

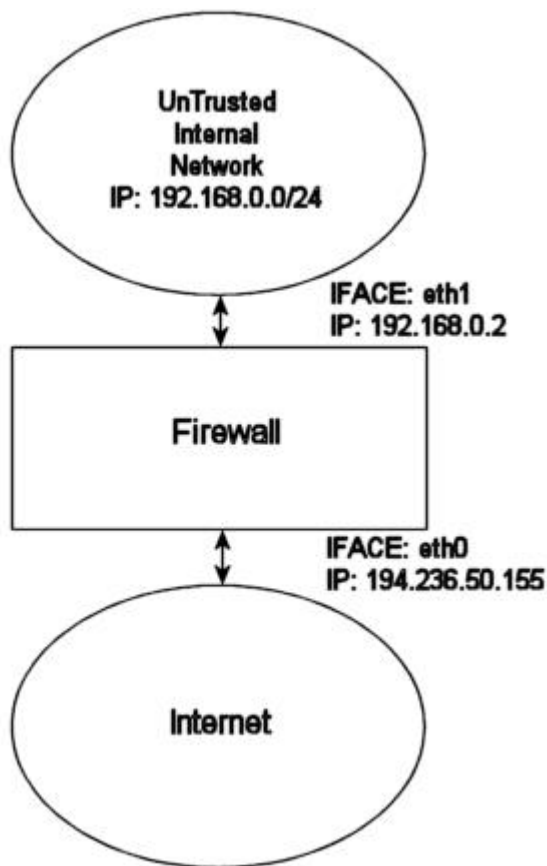
还有一种方法可获得 IP, 就是在脚本里加上类似这样的语句:

```
INET_IP=`ifconfig $INET_IFACE | grep inet | cut -d : -f 2 |
\
cut -d ' ' -f 1`
```

上面这句话的作用是从 **ifconfig** 的输出里提取接口 *\$INET_IFACE* 的 IP, 再赋给 *\$INET_IP*。更好的办法是使用脚本 [retreiveip.txt](#)。但要注意, 这个方法可能会引起一些不正常的情况, 比如使防火墙和内网之间已有的连接停止。下面就说明一下最常见的问题。

1. 如果这个脚本的代码是在另一个脚本内运行的, 而那个脚本又是由 *PPP daemon* 启动的, 就会因为 NEW not SYN rules (具体信息查看 [未设置 SYN 的 NEW 状态包](#)) 的原因而挂起所以当前活动的连接。如果你删掉那个规则, 可能会没有事, 但还是不保险。
2. 如果你不想改动已有的规则, 而又要添加或删除规则, 还要不损害已有的规则, 这就没法做了。比如, 你又想阻塞所有局域网里的机子访问防火墙, 又想让它们能控制防火墙上 *PPP daemon*, 如果不删除那个用来阻塞的规则, 怎么能完成这样的事?
3. 事情可能也不必这么复杂, 就像上面说的, 这会导致一些安全问题。但如果这个脚本能保持简单, 维持规则的顺序与发现问题都是很容易的。

8.5. rc.UTIN.firewall.txt



脚本[rc.UTIN.firewall.txt](#)适用于这样的情况：我们不信任任何与防火墙连接的网络，包括内网。我们只允许内网使用*POP3*、*HTTP* 和*FTP*。至于从Internet来的连接，权限和其他脚本一样。

此脚本需要以下功能模块。

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

这个脚本遵循的原则是不要相信任何人，包括我们自己的员工。这是个令人悲痛的现实，大部分破坏和攻击确实是来自我们内部的。这个脚本只是在加强防火墙方面给了你一个例子。它和 `rc.firewall.txt` 并没有太多的不同，只是少了一些允许通行的规则。

8.6. rc.test-iptables.txt

这个脚本用来测试iptables里所有的链。当然, 这要根据你的配置情况做些操作, 如打开 `ip_forwarding` 或是设置 `masquerading`, 等等。只要你的安装了基本的 iptables, 就可以使用它。其实这个脚本只使用了LOG, 以便能记录所有的ping请求与应答。通过这种方式, 我们就可以了解哪些链被穿越了以及被穿越的顺序。使用方法如下, 先运行这个脚本, 再发布一个ping命令, 如:

```
ping -c 1 host.on.the.internet
```

然后用命令 `tail -n 0 -f /var/log/messages` 就可看到用了哪些链以及是什么顺序, 除非记录因某些原因被替换了。



此脚本仅仅是为测试而写的。也就是说, 不要使用类似这样的规则, 它记录某一类包的所有信息, 这会很快地占满你的日志分区, 而且它会成为一个有效的DoS攻击。它还可能导致在最初的DoS攻击之后, 无法记录真正的攻击信息。

8.7. rc.flush-iptables.txt

[rc.flush-iptables.txt](#) 不应该被称作脚本, 它只是重置并清空所有的表、链。它先把 *filter* 表的 *INPUT*、*OUTPUT* 和 *FORWARD* 链的策略设为缺省的ACCEPT, 然后是 *nat* 表的 *PREROUTING*、*POSTROUTING* 和 *OUTPUT* 链。我们这样做就不必为被关闭的连接和没有通过的包而操心。这个脚本就是为防火墙设置和除错用的, 因此我们只关心打开所有的东西并恢复它们的缺省值就行了。

之后, 我们清空 *filter* 表里所有的链, 紧接着是 *NAT* 表的。这样, 就不会有什么不应该存在的规则了。这个做完后就该删除 *filter* 表和 *NAT* 表里的自定义链了。这时, 脚本的工作就应该完成了。当然, 如果你用到了 *mangle* 表, 可以在这个脚本里添加相应的清空规则。(译者注: 其实作者已经这样做了)



最后再说明一下, 有些人写信建议我把这个脚本放到 `rc.firewall` 脚本里面, 而且是用 Red Hat Linux 脚本的语法, 这样当 `rc.firewall` 启动时, 这个脚本也可以启动。但我不会这样做的, 因为这是一个指南, 主要是用来学习 iptables 的使用方法的, 不应该有过多的 shell 脚本特有的语法。加入 shell 脚本特有的语法会使阅读的难度大大增加, 这就远离了我的初衷。这个指南是按照易读的标准来写的, 以后我会继续这样做。

8.8. Limit-match.txt

这个脚本是用来测试 `limit match` 的, 也会让你明白 `limit match` 是如何工作的。装入这个脚本, 再用不同的时间间隔发送ping数据包, 可以看出哪个包可以通过, 这些包又是以什么频率通过的。你应该可以看出, 在 `limit` 的 `burst` 值再次到达之前, 所有的 *echo replies* 都会被阻塞。

8.9. Pid-owner.txt

这个脚本说明了如何使用PID owner match。它其实什么都没做，但你可以运行一下，命令 `iptables -L -v` 的输出会说明它确实匹配了些东西。

8.10. Sid-owner.txt

说明SID owner match如何使用的一个例子。同样，它也是什么都没做，但你可以运行一下，命令 `iptables -L -v` 的输出会说明它确实匹配了些东西。

8.11. Ttl-inc.txt

一个小小的例子，说明了如何隐藏我们的防火墙或路由器，以使跟踪路由程序看不到，这样就可以对可能的攻击者隐藏很多信息。

8.12. Iptables-save ruleset

这只是一个输出的例子，它在[规则的保存与恢复](#)里被用来说明 `iptables-save` 命令是如何使用的。所以，它没有任何用处，只是一个参考而已。

附录 A. 常用命令详解

A.1. 查看当前规则集的命令

查看当前正在使用的规则集是一个十分常用的操作，使用 `iptables` 的什么命令还记得吗？我们可是在[规则是如何练成的](#)这一章里介绍过啊，虽然到时说得简单了点。再复习一下吧，命令语法如下：

```
iptables -L
```

这个命令会尽可能地以易读的形式显示当前正在使用的规则集。比如，它会尽量用文件 `/etc/services` 里相应的名字表示端口号，用相应的DNS记录表示IP地址。但后者可能会导致一些问题，例如，它想尽力把LAN的IP地址（如192.168.1.1）解析成相应的名字。但192.168.0.0/16这个网段是私有的，也就是说，它只能用在局域网里，而不能在Internet里使用，所以它不会被Internet上的DNS服务器解析。因此，当解析这个地址时，命令就好像停在那儿了。为了避免这种情况的发生，我们就要使用选项：

```
iptables -L -n
```

如果你想看看每个策略或每条规则、每条链的简单流量统计，可以在上面的命令后再加一个 `verbose` 标志，如下：

```
iptables -L -n -v
```

不要忘了，`iptables -L` 命令还可以查看 `nat` 表和 `mangle` 表的内容哦（更不要忘了，默认的表是 `filter`），只需要使用 `-t` 选项，比如我们只想看 `nat` 表的规则，就用下面的命令：

```
iptables -L -t nat
```

在 `/proc` 里，可能还有一些文件你会感兴趣。比如，你可以在连接跟踪记录表里看到当前有哪些连接。这个表包含了当前的所有连接，你还可以通过它了解到每个连接处于什么状态。要注意，这个表是不能编辑的，即使可以，也不应该更改它。可以用下面的命令查看这个表：

```
cat /proc/net/ip_conntrack | less
```

此命令会显示当前所有被跟踪的连接，但要读懂那些记录可是有些难度哦。

A. 2. 修正和清空iptables的命令

即使你把 `iptables` 弄的一塌糊涂，我们也有非常有效的命令来处理，而不必重新启动计算机。我接到过很多关于这个问题的询问，所以我想最好在这儿回答一下。如果你增加的规则有问题，要想删掉它，只要把命令中的 `-A` 改为 `-D` 即可。这样，`iptables` 就会找到那个错误的规则并删掉它，但如果在你的规则里有好几条同样的规则，它只能删掉找到的第一条。如果你不想这样的事情发生，那就试试用序号来删除。如，你想删除 `INPUT` 链的第10条规则，可以使用 `iptables -D INPUT 10`。

还有一种情况，就是要清空整个链，这就要使用选项 `-F`。比如，我们要清空整个 `INPUT` 链，使用的命令就是 `iptables -F INPUT`。但是要注意，选项 `-F` 并不改变链的缺省策略。所以，如果被我们清空的那条 `INPUT` 链的策略是 `DROP`，它还是会阻塞所有的包。那怎么才能重置策略呢？还记得策略 `DROP` 是如何设置的吧，还是用那个方法啊。比如，我们把 `INPUT` 链的策略改为 `ACCEPT`，就用 `iptables -P INPUT ACCEPT`。

我已经写了一个用来清空并重置 `iptables` 的脚本，叫做 [*rc.flush-iptables.txt*](#)（附录里有它的代码），在你写自己的防火墙脚本时，很可能会用到。但如果你在 `mangle` 表里乱试乱改而导致问题的话，这个脚本就帮不上忙了。因为在脚本 `rc.firewall.txt` 里，我没有用到 `mangle` 表，所以在 `rc.flush-iptables.txt` 里也就没有添加相应的恢复功能。

附录 B. 常见问题与解答

B. 1. 模块装载问题

装载模块时，你可能会遇到几个问题，比如，有错误提示说明没有你指定名字的那种模块：

```
insmod: iptable_filter: no module by  
that name found
```

这个提示是无关紧要的, 因为那些模块很有可能已经被静态地编译进内核了。当你遇到这个信息时, 这是你应该首先想到的。至于是否真的如我们所想, 最简单的测试方法就是敲一个用到那个模块功能的命令试试。对于上面的情况, 可能是filter表没有装入, 从而就没有相应的功能, 当然不能使用filter表了。为了检查 filter表是否装入, 可以用下面的命令来试试:

```
iptables -t filter -L
```

这个命令会输出filter表里所有的链, 或者是运行失败, 给出错误提示信息。如果一切正常, 输出结果类似下面的情况, 当然, 这还要看你是否已经在filter表里加入了规则 (译者注: 在这个例子里, 表是空的)。

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

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

如果你确实没有装载filter表, 得到的就是如下信息:

```
iptables
v1.2.5: can't initialize iptables table `filter': Table \
      does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
```

这个问题就有些严重了, 从此提示中我们能得到两个信息: 第一, 我们确实没有把相应的功能编译进内核里; 第二, 在模块一般应在的目录中没有找到这个模块。这意味着问题是, 你或者忘记了装载想用的模块, 或者没有用**depmod -a**命令更新模块数据库, 或者没有把相应的功能编译进内核 (不论是静态的还是作为模块)。当然还可能是其他原因, 但这些是主要的, 不管怎样, 大部分原因是很容易解决的。比如, 第一个问题可以简单地通过在源码目录里运行**make modules_install** 命令来解决, 这当然是有前提的, 就是源码已经编译 (compile) 而且模块已经构建 (build)。第二个问题的解决办法也很简单, 只要运行一下**depmod -a**命令, 之后再看看能否正常工作即可。第三个问题有点超出我们的范围了, 而且这个问题或多或少会让你感到发晕。更多的信息可以在[Linux文档计划](#)里找到。

在运行iptables时, 你还可能得到另外一个错误信息:

```
iptables: No chain/target/match by that name
```

这说明你要用的链或target、或match不存在, 原因有很多, 但最普遍的是你拼错了名字。当你想使用一个不可用的模块时也会产生这种错误。模块之所以不可用, 可能是因为你没有装载正确的模块, 或者内核里不包含那个模块, 或者是iptables自动装载模块时失败了。通常, 你不止应该考虑上面提到的所有解决办法, 还要考虑规则中target的拼写错误, 或者其他的原因。

B. 2. 未设置SYN的NEW状态包

iptables有个“特点”没有被很好地给以说明,所以很多人(当然,也包括我)都忽视了它。这个“特点”就是:如果你使用状态**NEW**,那么未设置SYN的包也会通过防火墙。之所以有这个特点,是因为在某些情况下,我们想把那样的包看作某个(比如是和另一个防火墙有关的)已处于**ESTABLISHED**状态的连接的一部分。这个特点使拥有两个或更多的防火墙协同工作成为可能,而且可使数据在服务器间无丢失的传输,如辅助防火墙可以接受子网的防火墙的操作。但它也会导致这样的事情:状态 **NEW**会允许几乎所有的TCP连接进入,而不管是否有3次握手。为了处理这个问题,我们需要在防火墙的 INPUT链、OUTPUT链和FORWARD链加入如下规则(译者注:此规则作者称为“NEW not SYN rules”,下一小节还会提到):

```
$IPTABLES -A
INPUT -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```



警告,在Netfilter/iptables项目中,这个特点所拥有的行为缺少文档说明,更明确的说,在你的防火墙上,它是一个很不安全的因素。

注意,这个规则用于microsoft的TCP/IP(微软实现的TCP/IP就是不行,至少现在不行)产生的包时还是有些问题。如果包是由microsoft的产品生成的,且被标为状态**NEW**,那么就会被此规则记录然后丢弃。看起来规则工作很正常啊,是吧。但问题就出在这儿了,因为连接无法中断了。这个问题出现在关闭连接时,在最后一个包即FIN/ACK包发出后,Netfilter的状态机制就会关闭连接、删除连接跟踪表里的相应记录。但就在这时,Microsoft那不完美的程序会发送另外一个包,这个包就是那种未设置SYN且被认为是NEW状态的包,因此它就会被上面的规则匹配。换句话说,就是对这个规则不需要过于关注,如果你很在意它,就在规则里加入选项--log-headers吧。这样,你就可以把包头记录下来,从而可以更好地了解相应的包。

对于这个规则,还有一些已知的问题。比如,某个连接(比如是从LAN发出的)已经连接到防火墙,而且有个脚本要在启动PPP时激活。当你启动PPP连接时,刚才提到的那个连接可能就会被干掉(be killed)。当然,这只会特定的情况下才能发生,就是你把conntrack和nat作为模块运行,并且每次运行那个脚本时这两个模块都要被装入和卸载。如果你在防火墙之外的机子上运行telnet,而且又通过这个telnet连接运行脚本rc.firewall.txt,也会导致上面的问题。为了能简单地表达这个问题,你先准备一个telnet连接,或其他的流连接,再运行连接跟踪模块,然后装入上面的规则,最后,试着用telnet client或daemon发送一些数据。效果应该出来了,连接跟踪代码会认为这个连接是非法的,因为在此之前,它没有看到任何方向有包发出,更为严重的是现在连接上有了未设置SYN的包,因为刚才由telnet client或daemon发出的包肯定不是这个连接的第一个包。因此,上面的规则就起作用了,也就是说,这个包会被记录下来,然后被无情地扔掉,从而连接就会中断。

B. 3. NEW状态的SYN/ACK包

某些,TCP欺骗攻击所用的技术叫做序列号预测(Sequence Number Prediction)。在这类攻击中,攻击者利用另一台机子的IP访问攻击对象(译者注:这就是为什么叫欺骗的原因了,攻击者是想假冒另一台被攻击对象信任的机器,以达到欺骗攻击对象的目的),然后再试着预测攻击对象使用什么序列号。

我们来看看典型的使用序列号预测技术的欺骗是如何实现的,参与者:攻击者[A]

(attacker)试图假装另一台机器[O](other host)向受害者[V](victim)发送数据。

1. [A]以[0]的IP为源地址向[V]发SYN。
2. [V]向[0]回应SYN/ACK。
3. 现在, 若[0]以RST回应这个未知的SYN/ACK, 攻击就失败了, 但如果[0]已经没有这个能力了呢? 比如它早已被另外的攻击(如SYN flood)降服, 或者被关闭, 或者它的RST包被防火墙拒绝。
4. 如果[0]没能破坏这条连接, 而且[A]猜对了序列号, 那它就能以[0]的身份和[V]交谈了。

只要我们没能在第三步以RST回应那个未知的SYN/ACK包, [V]就会被攻击, 而且我们还会被连累(译者注: 因为我们本身也被攻击了, 而且还可能会成为攻击者的替罪羊被起诉, 呜呜, 好惨)。所以, 为安全起见, 我们应该以正确的方式向[V]发送一个RST包。如果我们使用类似“NEW not SYN rules”(译者注: 在上一小节中)的规则, SYN/ACK包就可以被丢弃了。因此, 我们在bad_tcp_packets链中加入了如下规则:

```
iptables -A
bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
```

这样, 你想成为上面那个[0]的机会就很少了(译者注: 作者好幽默啊, 我们可不想成为被别人利用的对象), 而且这条规则在绝大部分情况下是安全的, 不会有什么副作用, 但多个防火墙要协同工作的情况要除外。那种情况下, 防火墙之间会经常传递、接受包或流, 有了这条规则, 有些连接可能会被阻塞, 即使是合法的连接。这条规则的存在还产生了另外一问题, 就是有几个portscan(端口扫描器)会看到我们的防火墙, 但好在仅此而已。

B. 4. 使用私有IP地址的ISP

我的一位朋友告诉我说有些事我完全忘记了, 从那时起, 我就把这一节加上了。你刚上网时连接的网络是ISP提供的, 但某些愚蠢的ISP在那个网络里使用的是私有地址, 而那是IANA专门分配给局域网使用的。Swedish Internet Service Provider和电话垄断企业Telia就是这样做的, 例如在DNS服务器上, 他们使用的IP地址段就是10. x. x. x。我们最容易遇到的问题是, 在这个脚本里, 为了防止被欺骗, 不允许从10. x. x. x发出的连接来访问我们。不幸的是, 对于上面的例子, 为了DNS能正常地被访问, 我们不得不把规则的放宽松一些。也就是说, 我们或者在刚才提到的那条防止欺骗的规则上面增加一条规则(如下), 或者是把那条规则注释掉:

```
/usr/local/sbin/iptables -t nat -I PREROUTING -i
eth1 -s \
10.0.0.1/32 -j ACCEPT
```

我愿意对这些ISP再多费些唇舌。这些IP地址不是为了让你象这样愚蠢的使用而分配给你的, 至少我知道不是这样的。对于一个大集团的站点或者是我们自己的家庭网络来说, 这样用是很合适的, 但你不能只因为你们的一些原因就强迫我们把自己公示于天下。

B. 5. 放行DHCP数据

一旦你了解DHCP是如何工作的, 就会知道这其实是一个很简单的任务。但你必须小心处理到底让谁进入、不让谁进入。首先, 我们要明白DHCP是工作在UDP协议之上的, 所以, UDP协议是我们期望的第一个条件。其次, 我们应该检查是从那个接口接收和发送请求的。例如, 如果我们设置了DHCP使用接口eth0, 那就要阻塞 eth1上的DHCP请求。为了让规则再详细些, 我们只需打开 (allow) DHCP实际使用的UDP端口, 一般都是67和 68。这两个端口是标准定义, 我们就用它们来匹配被允许的包。现在, 规则应该是这个样子的:

```
$IPTABLES -I INPUT -i $LAN_IFACE -p udp --dport 67:68 --sport \
67:68 -j ACCEPT
```

注意, 现在我们能够接受所有来自和发往UDP端口67、68的数据, 好像不太安全, 但这并不是多大的问题, 因为这条规则只允许从67或68端口连接的主机才能访问。当然, 此规则还可以更严谨一些, 但也应该足够接受所有的DHCP请求和更新, 而不至于需要在防火墙上开一个大洞。如果你很在意现在的规则是否很宽松, 你当然可以写一个限制条件更紧的。

B. 6. 关于mIRC DCC的问题

mIRC使用一个特殊的设定, 它可以使mIRC连接穿过防火墙, 也可以使DCC连接能在防火墙不了解它的情况下正常工作。如果此选项和iptables还有ip_conntrack_irc模块与ip_nat_irc模块一起使用, 那mIRC就不能工作了。问题在于mIRC会自动对包进行NAT操作, 这样当包到达防火墙后, 防火墙就完全不知道该对包做什么了, 也不知道该怎么做。如果是防火墙来处理, 它只是简单地用自己的IP去询问IRC服务器, 然后用那个地址发送DCC请求。mIRC不希望防火墙自作聪明地以这种方式代替自己来处理这个包。

打开 “I am behind a firewall” (我在防火墙后) 这个配置选项并且使用 ip_conntrack_irc和 ip_nat_irc模块, 会导致Netfilter建立包含 “Forged DCC send packet” 的记录。

最简单的解决办法是不要选中mIRC的那个选项而让iptables来做这些工作。意思就是要明确地告诉mIRC, 它不是在防火墙后面的。

附录 C. ICMP类型

这是一个完整的ICMP类型的列表:

Table C-1. ICMP类型

TYPE	CODE	Description	Query	Error
0	0	Echo Reply——回显应答 (Ping应答)	x	
3	0	Network Unreachable——网络不可达		x
3	1	Host Unreachable——主机不可达		x

3	2	Protocol Unreachable——协议不可达		x
3	3	Port Unreachable——端口不可达		x
3	4	Fragmentation needed but no frag. bit set——需要进行分片但设置不分片比特		x
3	5	Source routing failed——源站选路失败		x
3	6	Destination network unknown——目的网络未知		x
3	7	Destination host unknown——目的主机未知		x
3	8	Source host isolated (obsolete)——源主机被隔离（作废不用）		x
3	9	Destination network administratively prohibited——目的网络被强制禁止		x
3	10	Destination host administratively prohibited——目的主机被强制禁止		x
3	11	Network unreachable for TOS——由于服务类型 TOS, 网络不可达		x
3	12	Host unreachable for TOS——由于服务类型 TOS, 主机不可达		x
3	13	Communication administratively prohibited by filtering——由于过滤, 通信被强制禁止		x
3	14	Host precedence violation——主机越权		x
3	15	Precedence cutoff in effect——优先中止生效		x
4	0	Source quench——源端被关闭（基本流控制）		
5	0	Redirect for network——对网络重定向		
5	1	Redirect for host——对主机重定向		
5	2	Redirect for TOS and network——对服务类型和网络重定向		
5	3	Redirect for TOS and host——对服务类型和主机重定向		
8	0	Echo request——回显请求（Ping请求）	x	
9	0	Router advertisement——路由器通告		
10	0	Route solicitation——路由器请求		
11	0	TTL equals 0 during transit——传输期间生存时间为0		x
11	1	TTL equals 0 during reassembly——在数据报组装期间生存时间为0		x
12	0	IP header bad (catchall error)——坏的IP首部（包括各种差错）		x
12	1	Required options missing——缺少必需的选项		x
13	0	Timestamp request (obsolete)——时间戳请求（作废不用）	x	
14		Timestamp reply (obsolete)——时间戳应答（作废不用）	x	
15	0	Information request (obsolete)——信息请求	x	

		(作废不用)		
16	0	Information reply (obsolete)——信息应答 (作废不用)	x	
17	0	Address mask request——地址掩码请求	x	
18	0	Address mask reply——地址掩码应答	x	

附录 D. 其他资源和链接

这里有一些资源的链接，我从这些地方获得了不少信息，相信对你应该也很有帮助：

- [ip-sysctl.txt](#) ——来自内核2.4.14，一篇关于IP网络控制参数的短小精干的参考文章。
- [The Internet Control Message Protocol](#) ——一篇很好的详细介绍ICMP协议的文章，作者是Ralph Walden。
- [RFC 792 - Internet Control Message Protocol](#) ——ICMP的权威文件，如果你想找关于ICMP协议的信息，这是你应该首先想到的地方。作者：J. Postel。
- [RFC 793 - Transmission Control Protocol](#) ——TCP的权威文件，从1981年开始，它就成为TCP的规范了。只要你想学习TCP，就一定要读读这篇技术性很强的文章。作者：J. Postel
- [ip_dynaddr.txt](#) ——来自内核2.4.14，关于通过sysctl和proc文件系统设置ip_dynaddr 的参考文章。
- [iptables.8](#) ——iptables 1.2.4的帮助，这是HTML版本的。在你读写iptables规则时，这是一个很好的参考，你应该把它带在身边。
- [Firewall rules table](#) ——由Stuart Clark给出的一个小小的PDF文件，里面是防火墙配置的参考样式，对你书写自己的防火墙规则很有帮助。
- <http://www.netfilter.org/> ——Netfilter和iptables的官方网站，是每一个打算在linux里配置iptables和 Netfilter的人必到之处。
- <http://www.netfilter.org/documentation/index.html#FAQ> ——官方的Netfilter *Frequently Asked Questions*，是开始了解iptables和Netfilter的好去处。
- <http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/index.html> ——非常好的包过滤基础指南，介绍了如何使用iptables进行包过滤。作者是iptables和Netfilter的核心开发者之一Rusty Russell。
- <http://www.netfilter.org/unreliable-guides/NAT-HOWTO/index.html> ——介绍网络地址转换的很好的指南。作者是iptables和Netfilter的核心开发者之一Rusty Russell。
- <http://www.netfilter.org/unreliable-guides/netfilter-hacking-HOWTO/index.html> ——只有很少的文章介绍如何在Netfilter和iptables 的用户空间、内核空间里编写代码，这是其中一篇。作者还是Rusty Russell。

- <http://www.linuxguruz.org/iptables/> ——很好的资源链接网页，里包含了Internet上大部分关于iptables的链接，尤其是它还包含了很多为不同用处而写的iptables脚本的链接。
- <http://www.islandsoft.net/veerapen.html> ——这篇文章讨论了iptables自动增强坚固性的可能，以及如何通过很少的改动使你的计算机能自动地把敌对站点加入iptables的一个特殊的“禁止列表”。
- </etc/protocols> ——此文件是从Slackware发行版中抽取的。你可以利用此文件找到协议所对应的协议号，如IP、ICMP或TCP对应的号码。
- </etc/services> ——此文件也是从Slackware发行版中抽取的。它非常值得一读，你可以大致了解什么协议使用什么端口。
- [Internet Engineering Task Force](#) ——IETF是制定和维护互联网标准的最大的组织之一，很多大企业集团和个人都是它的成员，他们共同工作是为了确保Internet的互操作性。
- [Linux Advanced Routing and Traffic Control HOW-TO](#) ——此站点主要讨论Linux高级路由和流量控制，这个HOW-TO是关于Linux高级路由的最大的也是最好的一篇文章。作者是Bert Hubert。
- [Paksecured Linux Kernel patches](#) ——此站点包含了Matthew G. Marsh写的所有内核补丁，FTOS patch就在这儿。
- [ULOGD project page](#) ——ULOGD的站点。
- [The Linux Documentation Project](#) ——有关Linux的文档的极好（可以说是最好）的站点。有关Linux的很多较大的文档这儿都有，如果TLDP里没有，你就要好好地在网络上搜索一下了。如果你想了解多一些，就去看看吧。
- <http://kalamazoolinux.org/presentations/20010417/conntrack.html> ——这篇文章里有一个极其精彩的例子，它是用来展示conntrack模块以及它在Netfilter里的工作的。如果你想多看一些有关conntrack的文章，这一篇应该是必读的。
- <http://www.docum.org/> ——此站点包含了全部有关CBQ（Class Based Queue）、tc和ip命令的资料，这是很少的几个这样的站点中的一个。此站点由Stef Coene维护。
- <http://lists.samba.org/mailman/listinfo/netfilter> ——Netfilter的官方邮件列表，非常有用哦。万一你遇到了一些问题，而这篇文章或这里提到的一些链接解决不了，它就是你的救世主了。

当然，资源不止我上面提到的这些，还有iptables的源码和文档，及很多可以帮助你的朋友。

附录 E. 鸣谢

很多朋友在我写这篇文章时给了我热心的帮助，我要感谢他们：

- [Fabrice Marie](#)，对我糟糕的语法和拼写做了大量的订正，还用make文件等工具把这篇

指南转换成了DocBook。

- [*Marc Boucher*](#), 在状态匹配代码的使用方面给了我很多帮助。
- [*Frode E. Nyboe*](#), 大幅度改善了rc.firewall的规则, 当我要重写这个规则集、把多个表的遍历(the multiple table traversing)引入同一份文件时, 给了我很多灵感。
- [*Chapman Brad*](#), [*Alexander W. Janssen*](#), 开始时, 我对包如何穿越nat和filter 表的理解是错误的, 是他们使我了解到这一点的, 而且他们还给了我正确的顺序。
- [*Michiel Brandenburg*](#), [*Myles Uyema*](#), 帮我解决了一些状态匹配代码, 并让它正常工作。
- [*Kent 'Artech' Stahre*](#), 帮我绘制图形, 还帮我查错。
- *Anders 'DeZENT' Johansson*, 提示我有些古怪的ISP在Internet上使用保留的网址, 至少对他来说遇到了这样的情况。
- [*Jeremy 'Spliffy' Smith*](#), 提示我有些内容容易使大家糊涂, 还帮我进行了测试和查错。

还有很多人, 我和他们进行过讨论, 也请教过他们, 这里不能一一提及了。

Appendix F. History

Version 1.1.19 (21 May 2003)

.

By: Oskar Andreasson

Contributors: Peter van Kampen, Xavier Bartol, Jon Anderson, Thorsten Bremer and Spanish Translation Team.

Version 1.1.18 (24 Apr 2003)

.

By: Oskar Andreasson

Contributors: Stuart Clark, Robert P. J. Day, Mark Orenstein and Edmond Shwayri.

Version 1.1.17 (6 Apr 2003)

.

By: Oskar Andreasson

Contributors: Geraldo Amaral Filho, Ondrej Suchy, Dino Conti, Robert P. J. Day, Velez Dimo, Spencer Rouser, Daveonos, Amanda Hickman, Olle Jonsson and Bengt Aspvall.

Version 1.1.16 (16 Dec 2002)

.

By: Oskar Andreasson

Contributors: Clemens Schwaighower, Uwe Dippel and Dave Wreski.

Version 1.1.15 (13 Nov 2002)

.

By: Oskar Andreasson

Contributors: Mark Sonarte, A. Lester Buck, Robert P. J. Day, Togan Muftuoglu, Antony Stone, Matthew F. Barnes and Otto Matejka.

Version 1.1.14 (14 Oct 2002)

.

By: Oskar Andreasson

Contributors: Carol Anne, Manuel Minzoni, Yves Soun, Miernik, Uwe Dippel, Dave Klipec and Eddy L O Jansson.

Version 1.1.13 (22 Aug 2002)

<http://iptables-tutorial.haringstad.com>

By: Oskar Andreasson

Contributors: Tons of people reporting bad HTML version.

Version 1.1.12 (19 Aug 2002)

<http://www.netfilter.org/tutorial/>

By: Oskar Andreasson

Contributors: Peter Schubnell, Stephen J. Lawrence, Uwe Dippel, Bradley Dilger, Vegard Engen, Clifford Kite, Alessandro Oliveira, Tony Earnshaw, Harald Welte, Nick Andrew and Stepan Kasal.

Version 1.1.11 (27 May 2002)

<http://www.netfilter.org/tutorial/>

By: Oskar Andreasson

Contributors: Steve Hnizdur, Lonni Friedman, Jelle Kalf, Harald Welte, Valentina Barrios and Tony Earnshaw.

Version 1.1.10 (12 April 2002)

<http://www.boingworld.com/workshops/linux/iptables-tutorial/>

By: Oskar Andreasson

Contributors: Jelle Kalf, Theodore Alexandrov, Paul Corbett, Rodrigo Rubira Branco, Alistair Tonner, Matthew G. Marsh, Uwe Dippel, Evan Nemerson and Marcel J.E. Mol.

Version 1.1.9 (21 March 2002)

<http://www.boingworld.com/workshops/linux/iptables-tutorial/>

By: Oskar Andreasson

Contributors: Vince Herried, Togan Muftuoglu, Galen Johnson, Kelly Ashe, Janne Johansson, Thomas Smets, Peter Horst, Mitch Landers, Neil Jolly, Jelle Kalf, Jason Lam and Evan Nemerson.

Version 1.1.8 (5 March 2002)

<http://www.boingworld.com/workshops/linux/iptables-tutorial/>

By: Oskar Andreasson

Version 1.1.7 (4 February 2002)

<http://www.boingworld.com/workshops/linux/iptables-tutorial/>

By: Oskar Andreasson

Contributors: Parimi Ravi, Phil Schultz, Steven McClintoc, Bill Dossett, Dave Wreski, Erik Sjöstrand, Adam Mansbridge, Vasoo Veerapen, Aladdin and

Rusty Russell.

Version 1.1.6 (7 December 2001)

<http://people.unix-fu.org/andreasson/>

By: Oskar Andreasson

Contributors: Jim Ramsey, Phil Schultz, G鰈an B轟e, Doug Monroe, Jasper Aikema, Kurt Lieber, Chris Tallon, Chris Martin, Jonas Pasche, Jan Labanowski, Rodrigo R. Branco, Jacco van Koll and Dave Wreski.

Version 1.1.5 (14 November 2001)

<http://people.unix-fu.org/andreasson/>

By: Oskar Andreasson

Contributors: Fabrice Marie, Merijn Schering and Kurt Lieber.

Version 1.1.4 (6 November 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Contributors: Stig W. Jensen, Steve Hnizdur, Chris Pluta and Kurt Lieber.

Version 1.1.3 (9 October 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Contributors: Joni Chu, N.Emile Akabi- Davis and Jelle Kalf.

Version 1.1.2 (29 September 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Version 1.1.1 (26 September 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Contributors: Dave Richardson.

Version 1.1.0 (15 September 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Version 1.0.9 (9 September 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Version 1.0.8 (7 September 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Version 1.0.7 (23 August 2001)

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Contributors: Fabrice Marie.

Version 1.0.6

http://people.unix-fu.org/andreasson

By: Oskar Andreasson

Version 1.0.5
<http://people.unix-fu.org/andreasson>
By: Oskar Andreasson
Contributors: Fabrice Marie.

Appendix G. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all

copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role

of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled

“Dedications”. You must delete all sections entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute
and/or modify this document under the terms of the GNU Free
Documentation License, Version 1.1 or any later version published by
the Free Software Foundation; with the Invariant Sections being LIST
THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-
Cover Texts being LIST. A copy of the license is included in the
section entitled "GNU Free Documentation License".
```

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend

releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix H. GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to

know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

1. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - A. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - B. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - C. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in

object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

11. NO WARRANTY

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND

FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

2. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief  
idea of what it does.>  
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify it  
under the terms of the GNU General Public License as published by the  
Free Software Foundation; either version 2 of the License, or (at your  
option) any later version.
```

```
This program is distributed in the hope that it will be useful, but  
WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
General Public License for more details.
```

```
You should have received a copy of the GNU General Public License along  
with this program; if not, write to the Free Software Foundation, Inc.,  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author Gnomovision
comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is
free software, and you are welcome to redistribute it under certain
conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the progra
m
`Gnomovision' (which makes passes at compilers)
written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

附录 I. 示例脚本的代码

I.1. rc.firewall脚本代码

```
#!/bin/sh
#
# rc.firewall - Initial SIMPLE IP Firewall script for Linux 2.4.x and iptables
#
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeiND0Tnet>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
```

```
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#####
#
# 1. Configuration options.
#

#
# 1.1 Internet Configuration.
#

INET_IP="194.236.50.155"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Local Area Network configuration.
#
# your LAN's IP range and localhost IP. /24 means to only use the first 24
# bits of the 32 bit IP address. the same as netmask 255.255.255.0
#

LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"

#
# 1.3 DMZ Configuration.
#

#
# 1.4 Local host Configuration.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 IPTables Configuration.
#

IPTABLES="/usr/sbin/iptables"

#
# 1.6 Other Configuration.
#

#####
#
# 2. Module loading.
#
```

```
#
# Needed to initially load modules
#

/sbin/depmod -a

#
# 2.1 Required modules
#

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state

#
# 2.2 Non-Required modules
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

#####
#
# 3. /proc set up.
#

#
# 3.1 Required proc configuration
#

echo "1" > /proc/sys/net/ipv4/ip_forward

#
# 3.2 Non-Required proc configuration
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#####
#
# 4. rules set up.
#

#####
# 4.1 Filter table
#

#
# 4.1.1 Set policies
#
```

```
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Create userspecified chains
#

#
# Create chain for bad tcp packets
#

$IPTABLES -N bad_tcp_packets

#
# Create separate chains for ICMP, TCP and UDP to traverse
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Create content in userspecified chains
#

#
# bad_tcp_packets chain
#

$IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# allowed chain
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# TCP rules
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

#
# UDP ports
#

#$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 4000 -j ACCEPT

#
```

```
# In Microsoft Networks you will be swamped by broadcasts. These lines
# will prevent them from showing up in the logs.
#

$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP

#
# If we get DHCP requests from the Outside of our network, our logs will
# be swamped as well. This rule will block them from getting logged.
#

$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP

#
# ICMP rules
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 INPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Rules for special networks not part of the Internet
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT

#
# Special rule for DHCP requests from LAN, which are not caught properly
# otherwise.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# Rules for incoming packets from the internet.
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_packets
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
# If you have a Microsoft Network on the outside of your firewall, you may
# also get flooded by Multicasts. We drop them so we do not get flooded by
# logs
#
```



```
#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Log weird packets that don't match the above.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 FORWARD chain
#

#
# Bad TCP packets we don't want
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Accept the packets we actually want to forward
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 OUTPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Special OUTPUT rules to decide which IP's to allow.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

#####
# 4.2 nat table
#

#
# 4.2.1 Set policies
```

```
#

#
# 4.2.2 Create user specified chains
#

#
# 4.2.3 Create content in user specified chains
#

#
# 4.2.4 PREROUTING chain
#

#
# 4.2.5 POSTROUTING chain
#

#
# Enable simple IP Forwarding and Network Address Translation
#

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 OUTPUT chain
#

#####
# 4.3 mangle table
#

#
# 4.3.1 Set policies
#

#
# 4.3.2 Create user specified chains
#

#
# 4.3.3 Create content in user specified chains
#

#
# 4.3.4 PREROUTING chain
#

#
# 4.3.5 INPUT chain
#

#
# 4.3.6 FORWARD chain
#

#
# 4.3.7 OUTPUT chain
#

#
# 4.3.8 POSTROUTING chain
#
```

I. 2. rc.DMZ.firewall脚本代码

```
#!/bin/sh
#
# rc.DMZ.firewall - DMZ IP Firewall script for Linux 2.4.x and iptables
#
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeindOTnet>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#####
#
# 1. Configuration options.
#

#
# 1.1 Internet Configuration.
#

INET_IP="194.236.50.152"
HTTP_IP="194.236.50.153"
DNS_IP="194.236.50.154"
INET_IFACE="eth0"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Local Area Network configuration.
#
# your LAN's IP range and localhost IP. /24 means to only use the first 24
# bits of the 32 bit IP address. the same as netmask 255.255.255.0
#

LAN_IP="192.168.0.1"
LAN_IFACE="eth1"

#
# 1.3 DMZ Configuration.
#
```

```
DMZ_HTTP_IP="192.168.1.2"
DMZ_DNS_IP="192.168.1.3"
DMZ_IP="192.168.1.1"
DMZ_IFACE="eth2"
```

```
#
# 1.4 Local host Configuration.
#
```

```
LO_IFACE="lo"
LO_IP="127.0.0.1"
```

```
#
# 1.5 IPTables Configuration.
#
```

```
IPTABLES="/usr/sbin/iptables"
```

```
#
# 1.6 Other Configuration.
#
```

```
#####
#
# 2. Module loading.
#
```

```
#
# Needed to initially load modules
#
/sbin/depmod -a
```

```
#
# 2.1 Required modules
#
```

```
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state
```

```
#
# 2.2 Non-Required modules
#
```

```
#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc
```

```
#####
#
# 3. /proc set up.
#
```

```

#
# 3.1 Required proc configuration
#

echo "1" > /proc/sys/net/ipv4/ip_forward

#
# 3.2 Non-Required proc configuration
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#####
#
# 4. rules set up.
#

#####
# 4.1 Filter table
#

#
# 4.1.1 Set policies
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Create userspecified chains
#

#
# Create chain for bad tcp packets
#

$IPTABLES -N bad_tcp_packets

#
# Create separate chains for ICMP, TCP and UDP to traverse
#

$IPTABLES -N allowed
$IPTABLES -N icmp_packets

#
# 4.1.3 Create content in userspecified chains
#

#
# bad_tcp_packets chain
#

$IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#

```



```
# allowed chain
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# ICMP rules
#

# Changed rules totally
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 INPUT chain
#

#
# Bad TCP packets we don't want
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Packets from the Internet to this box
#

$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
# Packets from LAN, DMZ or LOCALHOST
#

#
# From DMZ Interface to DMZ firewall IP
#

$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j ACCEPT

#
# From LAN Interface to LAN firewall IP
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j ACCEPT

#
# From Localhost interface to Localhost IP's
#

$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT

#
# Special rule for DHCP requests from LAN, which are not caught properly
# otherwise.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# All established and related packets incoming from the internet to the
```

```
# firewall
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT

#
# In Microsoft Networks you will be swamped by broadcasts. These lines
# will prevent them from showing up in the logs.
#

#$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP

#
# If we get DHCP requests from the Outside of our network, our logs will
# be swamped as well. This rule will block them from getting logged.
#

#$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP

#
# If you have a Microsoft Network on the outside of your firewall, you may
# also get flooded by Multicasts. We drop them so we do not get flooded by
# logs
#

#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Log weird packets that don't match the above.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 FORWARD chain
#

#
# Bad TCP packets we don't want
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# DMZ section
#
# General rules
#

$IPTABLES -A FORWARD -i $DMZ_IFACE -o $INET_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $INET_IFACE -o $DMZ_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $LAN_IFACE -o $DMZ_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $DMZ_IFACE -o $LAN_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT

#
# HTTP server
#
```

```
$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
--dport 80 -j allowed
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
-j icmp_packets

#
# DNS server
#

$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
-j icmp_packets

#
# LAN section
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 OUTPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Special OUTPUT rules to decide which IP's to allow.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

#####
# 4.2 nat table
#

#
# 4.2.1 Set policies
#
```

```
#
# 4.2.2 Create user specified chains
#

#
# 4.2.3 Create content in user specified chains
#

#
# 4.2.4 PREROUTING chain
#

$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --dport 80 \
-j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP

#
# 4.2.5 POSTROUTING chain
#

#
# Enable simple IP Forwarding and Network Address Translation
#

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 OUTPUT chain
#

#####
# 4.3 mangle table
#

#
# 4.3.1 Set policies
#

#
# 4.3.2 Create user specified chains
#

#
# 4.3.3 Create content in user specified chains
#

#
# 4.3.4 PREROUTING chain
#

#
# 4.3.5 INPUT chain
#

#
# 4.3.6 FORWARD chain
#

#
# 4.3.7 OUTPUT chain
#
```

```
#
# 4.3.8 POSTROUTING chain
#
```

I. 3. rc.UTIN.firewall脚本代码

```
#!/bin/sh
#
# rc.firewall - UTIN Firewall script for Linux 2.4.x and iptables
#
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#####
#
# 1. Configuration options.
#

#
# 1.1 Internet Configuration.
#

INET_IP="194.236.50.155"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Local Area Network configuration.
#
# your LAN's IP range and localhost IP. /24 means to only use the first 24
# bits of the 32 bit IP address. the same as netmask 255.255.255.0
#

LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
```



```
LAN_IFACE="eth1"

#
# 1.3 DMZ Configuration.
#

#
# 1.4 Local host Configuration.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 IPTables Configuration.
#

IPTABLES="/usr/sbin/iptables"

#
# 1.6 Other Configuration.
#

#####
#
# 2. Module loading.
#

#
# Needed to initially load modules
#

/sbin/depmod -a

#
# 2.1 Required modules
#

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state

#
# 2.2 Non-Required modules
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

#####
#
# 3. /proc set up.
#
```

```

#
# 3.1 Required proc configuration
#

echo "1" > /proc/sys/net/ipv4/ip_forward

#
# 3.2 Non-Required proc configuration
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#####
#
# 4. rules set up.
#

#####
# 4.1 Filter table
#

#
# 4.1.1 Set policies
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Create userspecified chains
#

#
# Create chain for bad tcp packets
#

$IPTABLES -N bad_tcp_packets

#
# Create separate chains for ICMP, TCP and UDP to traverse
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Create content in userspecified chains
#

#
# bad_tcp_packets chain
#

$IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

```

```
#
# allowed chain
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# TCP rules
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

#
# UDP ports
#

#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 4000 -j ACCEPT

#
# In Microsoft Networks you will be swamped by broadcasts. These lines
# will prevent them from showing up in the logs.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d $INET_BROADCAST \
--destination-port 135:139 -j DROP

#
# If we get DHCP requests from the Outside of our network, our logs will
# be swamped as well. This rule will block them from getting logged.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP

#
# ICMP rules
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 INPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Rules for special networks not part of the Internet
#

$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
```

```
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT

#
# Rules for incoming packets from anywhere.
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -j tcp_packets
$IPTABLES -A INPUT -p UDP -j udp_packets
$IPTABLES -A INPUT -p ICMP -j icmp_packets

#
# If you have a Microsoft Network on the outside of your firewall, you may
# also get flooded by Multicasts. We drop them so we do not get flooded by
# logs
#

#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Log weird packets that don't match the above.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 FORWARD chain
#

#
# Bad TCP packets we don't want
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Accept the packets we actually want to forward
#

$IPTABLES -A FORWARD -p tcp --dport 21 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 80 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 110 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 OUTPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
```

```
#
# Special OUTPUT rules to decide which IP's to allow.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

#####
# 4.2 nat table
#

#
# 4.2.1 Set policies
#

#
# 4.2.2 Create user specified chains
#

#
# 4.2.3 Create content in user specified chains
#

#
# 4.2.4 PREROUTING chain
#

#
# 4.2.5 POSTROUTING chain
#

#
# Enable simple IP Forwarding and Network Address Translation
#

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 OUTPUT chain
#

#####
# 4.3 mangle table
#

#
# 4.3.1 Set policies
#

#
# 4.3.2 Create user specified chains
#

#
# 4.3.3 Create content in user specified chains
#
```

```
#
# 4.3.4 PREROUTING chain
#

#
# 4.3.5 INPUT chain
#

#
# 4.3.6 FORWARD chain
#

#
# 4.3.7 OUTPUT chain
#

#
# 4.3.8 POSTROUTING chain
#
```

I. 4. rc.DHCP.firewall脚本代码

```
#!/bin/sh
#
# rc.firewall - DHCP IP Firewall script for Linux 2.4.x and iptables
#
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#####
#
# 1. Configuration options.
#

#
# 1.1 Internet Configuration.
#

INET_IFACE="eth0"

#
# 1.1.1 DHCP
#
```



```
#
# Information pertaining to DHCP over the Internet, if needed.
#
# Set DHCP variable to no if you don't get IP from DHCP. If you get DHCP
# over the Internet set this variable to yes, and set up the proper IP
# address for the DHCP server in the DHCP_SERVER variable.
#

DHCP="no"
DHCP_SERVER="195.22.90.65"

#
# 1.1.2 PPPoE
#

# Configuration options pertaining to PPPoE.
#
# If you have problem with your PPPoE connection, such as large mails not
# getting through while small mail get through properly etc, you may set
# this option to "yes" which may fix the problem. This option will set a
# rule in the PREROUTING chain of the mangle table which will clamp
# (resize) all routed packets to PMTU (Path Maximum Transmit Unit).
#
# Note that it is better to set this up in the PPPoE package itself, since
# the PPPoE configuration option will give less overhead.
#

PPPOE_PMTU="no"

#
# 1.2 Local Area Network configuration.
#
# your LAN's IP range and localhost IP. /24 means to only use the first 24
# bits of the 32 bit IP address. the same as netmask 255.255.255.0
#

LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"

#
# 1.3 DMZ Configuration.
#

#
# 1.4 Local host Configuration.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 IPTables Configuration.
#

IPTABLES="/usr/sbin/iptables"

#
# 1.6 Other Configuration.
#

#####
#
```

```

# 2. Module loading.
#

#
# Needed to initially load modules
#

/sbin/depmod -a

#
# 2.1 Required modules
#

/sbin/modprobe ip_conntrack
/sbin/modprobe ip_tables
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_MASQUERADE

#
# 2.2 Non-Required modules
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

#####
#
# 3. /proc set up.
#

#
# 3.1 Required proc configuration
#

echo "1" > /proc/sys/net/ipv4/ip_forward

#
# 3.2 Non-Required proc configuration
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#####
#
# 4. rules set up.
#

#####
# 4.1 Filter table
#

#
# 4.1.1 Set policies
#

```

```

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Create userspecified chains
#

#
# Create chain for bad tcp packets
#

$IPTABLES -N bad_tcp_packets

#
# Create separate chains for ICMP, TCP and UDP to traverse
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Create content in userspecified chains
#

#
# bad_tcp_packets chain
#

$IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# allowed chain
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# TCP rules
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

#
# UDP ports
#

$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
if [ $DHCP == "yes" ] ; then
    $IPTABLES -A udp_packets -p UDP -s $DHCP_SERVER --sport 67 \
    --dport 68 -j ACCEPT
fi

```

```
#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 4000 -j ACCEPT

#
# In Microsoft Networks you will be swamped by broadcasts. These lines
# will prevent them from showing up in the logs.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE \
--destination-port 135:139 -j DROP

#
# If we get DHCP requests from the Outside of our network, our logs will
# be swamped as well. This rule will block them from getting logged.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
--destination-port 67:68 -j DROP

#
# ICMP rules
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 INPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Rules for special networks not part of the Internet
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -j ACCEPT

#
# Special rule for DHCP requests from LAN, which are not caught properly
# otherwise.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# Rules for incoming packets from the internet.
#

$IPTABLES -A INPUT -p ALL -i $INET_IFACE -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_packets
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
```

```
# If you have a Microsoft Network on the outside of your firewall, you may
# also get flooded by Multicasts. We drop them so we do not get flooded by
# logs
#

$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Log weird packets that don't match the above.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 FORWARD chain
#

#
# Bad TCP packets we don't want
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Accept the packets we actually want to forward
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 OUTPUT chain
#

#
# Bad TCP packets we don't want.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Special OUTPUT rules to decide which IP's to allow.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $INET_IFACE -j ACCEPT

#
# Log weird packets that don't match the above.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

#####
```

```
# 4.2 nat table
#
#
# 4.2.1 Set policies
#
#
# 4.2.2 Create user specified chains
#
#
# 4.2.3 Create content in user specified chains
#
#
# 4.2.4 PREROUTING chain
#
#
# 4.2.5 POSTROUTING chain
#
if [ $PPPOE_PMTU == "yes" ] ; then
    $IPTABLES -t nat -A POSTROUTING -p tcp --tcp-flags SYN,RST SYN \
    -j TCPMSS --clamp-mss-to-pmtu
fi
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE
#
# 4.2.6 OUTPUT chain
#
#####
# 4.3 mangle table
#
#
# 4.3.1 Set policies
#
#
# 4.3.2 Create user specified chains
#
#
# 4.3.3 Create content in user specified chains
#
#
# 4.3.4 PREROUTING chain
#
#
# 4.3.5 INPUT chain
#
#
# 4.3.6 FORWARD chain
#
#
# 4.3.7 OUTPUT chain
#
```



```
#  
# 4.3.8 POSTROUTING chain  
#
```

I.5. rc.flush-iptables脚本代码

```
#!/bin/sh  
#  
# rc.flush-iptables - Resets iptables to default values.  
#  
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeinDOTnet>  
#  
# This program is free software; you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as published by  
# the Free Software Foundation; version 2 of the License.  
#  
# This program is distributed in the hope that it will be useful,  
# but WITHOUT ANY WARRANTY; without even the implied warranty of  
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
# GNU General Public License for more details.  
#  
# You should have received a copy of the GNU General Public License  
# along with this program or from the site that you downloaded it  
# from; if not, write to the Free Software Foundation, Inc., 59 Temple  
# Place, Suite 330, Boston, MA 02111-1307 USA  
  
#  
# Configurations  
#  
IPTABLES="/usr/sbin/iptables"  
  
#  
# reset the default policies in the filter table.  
#  
$IPTABLES -F INPUT ACCEPT  
$IPTABLES -F FORWARD ACCEPT  
$IPTABLES -F OUTPUT ACCEPT  
  
#  
# reset the default policies in the nat table.  
#  
$IPTABLES -t nat -F PREROUTING ACCEPT  
$IPTABLES -t nat -F POSTROUTING ACCEPT  
$IPTABLES -t nat -F OUTPUT ACCEPT  
  
#  
# reset the default policies in the mangle table.  
#  
$IPTABLES -t mangle -F PREROUTING ACCEPT  
$IPTABLES -t mangle -F OUTPUT ACCEPT  
  
#  
# flush all the rules in the filter and nat tables.  
#  
$IPTABLES -F  
$IPTABLES -t nat -F  
$IPTABLES -t mangle -F
```

```
#
# erase all chains that's not default in filter and nat table.
#
$IPTABLES -X
$IPTABLES -t nat -X
$IPTABLES -t mangle -X
```

I.6. rc.test-iptables脚本代码

```
#!/bin/bash
#
# rc.test-iptables - test script for iptables chains and tables.
#
# Copyright (C) 2001 Oskar Andreasson <bluefluxATkoffeindOTnet>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#
# Filter table, all chains
#
iptables -t filter -A INPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A INPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter FORWARD:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter FORWARD:"

#
# NAT table, all chains except OUTPUT which don't work.
#
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-reply \
```

```
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat OUTPUT:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat OUTPUT:"

#
# Mangle table, all chains
#
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle POSTROUTING:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle POSTROUTING:"
```